ES_LPC2210

Errata sheet LPC2210

Rev. 3 — 1 April 2011

Errata sheet

Revision history

Rev	Date	Description
3	20110401	• The format of this errata sheet has been redesigned to comply with the new identity guidelines of NXP Semiconductors. • Added ADC.1.
2.2	20080103	• Previous version.

Contact information

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

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

ES_LPC2210

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

© NXP B.V. 2011. All rights reserved.

Errata sheet **Rev. 3 — 1 April 2011** **2 of 22**

1. Product identification

The LPC2210 devices typically have the following top-side marking:

LPC2210xxx

xxxxxxx

xxYYWW R

The last letter in the last line (field 'R') will identify the device revision. This Errata Sheet covers the following revisions of the LPC2210:

Table 1.   Device revision table

Revision identifier (R)	Revision description
'A'	First device revision
'B'	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
ADC.1	First two ADC conversions in burst mode from same channel	'A'	Section 3.1
ADC.2	First conversion from channel specified by previous SEL setting	'A'	Section 3.2
ADC.3	Incorrect scan pattern	'A'	Section 3.3
ADC.4	Global powerdown does not power down the ADC	'A'	Section 3.4
ADC.5	Edge triggered ADC conversion start error	'A'	Section 3.5
ADC.6	Writing to ADCR while conversion in progress	'A'	Section 3.6
SPI.1	Unintentional clearing of SPI interrupt flag	'A', 'B'	Section 3.7
SPI.2	Incorrect shifting of data in slave mode at lower frequencies	'A'	Section 3.8
SSP.1	Initial data bits/clocks corrupted in SSP transmission	'B'	Section 3.9
EXTINT.1	Corruption of VPBDIV via EXTPOLAR or EXTMODE	'A'	Section 3.10
EXTINT.2	Incorrect setting of EXTMODE and/or EXTPOLAR	'A'	Section 3.11
CAP.1	Problem when selecting P0.21 as a capture 1.3 input (timer1)	'A'	Section 3.12
VPBDIV.1	Incorrect read of VPBDIV	'A'	Section 3.13
CORE.1	Incorrect load of the link register	'A', 'B'	Section 3.14
Timer.1	Missed Interrupt Potential	'A'	Section 3.15
PWM.1	Missed Interrupt Potential for Match Functionality	'A'	Section 3.16
Timer.2	Timer Counter reset occurs on incorrect edge in counter mode	'B'	Section 3.17
Timer0.1	Match 0.1 is not connected to P0.5	'A', 'B'	Section 3.18

Table 2. **Functional problems table** *...continued*

Functional problems	Short description	Revision identifier	Detailed description
UART.1	Coinciding VPB read and hardware register update	'A'	Section 3.19
Reset.1	Device does not power up correctly under certain internal conditions	'A'	Section 3.20
ADC.7	External sync inputs not operational	'A', 'B'	Section 3.21

Table 3. **AC/DC deviations table**

AC/DC deviations	Short description	Product version(s)	Detailed description
V_{IH}.1	Incompatibility of actual V_{IH} levels as compared to those specified	'A'	Section 4.1
V_3.1	Leakage current on V_3 due to External Interrupt and/or ADC pins	'A'	Section 4.2
AINx.1	Corruption of an ADC conversion	'A'	Section 4.3

Table 4. **Errata notes table**

Errata notes	Short description	Revision identifier	Detailed description
Note.1	Pin n.c. (pin 22, H2) must not be driven LOW during reset	'A', 'B'	Section 5.1

3. Functional problems detail

3.1 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.

3.2 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.

ES_LPC2210

Errata sheet **Rev. 3 — 1 April 2011** **5 of 22**

3.3 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.

3.4 ADC.4: Global power down 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.

ES_LPC2210

Errata sheet **Rev. 3 — 1 April 2011** **6 of 22**

3.5 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.

3.6 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.

3.7 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.

3.8 SPI.2: 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:

1. SPI is configured as a slave (with CPHA=0).
2. 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.

3.9 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 30 MHz, the first pulse can be expected to be approximately 10 ns shorter and the second pulse around 5 ns shorter. The remaining two pulses are around 2 ns shorter than subsequent pulses.

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

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

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

ES_LPC2210

© NXP B.V. 2011. All rights reserved.

Errata sheet

Rev. 3 — 1 April 2011

8 of 22

Work-around:

None.

3.10 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.

3.11 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 follows:

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 and you would see the EXTMODE and EXTPOLAR registers reflecting the correct values.

3.12 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.

3.13 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.

3.14 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:

ES_LPC2210

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

© NXP B.V. 2011. All rights reserved.

Errata sheet **Rev. 3 — 1 April 2011** **10 of 22**

<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 would have to be done manually.

3.15 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.

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

3.17 Timer.2: 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:

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

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

3. Edge 3- TC=0.

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

5. Edge 5- TC=0.

3.18 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.

ES_LPC2210

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

© NXP B.V. 2011. All rights reserved.

Errata sheet **Rev. 3 — 1 April 2011** **14 of 22**

3.19 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 behavior 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.

3.20 Reset.1: Device does not power up correctly under certain internal conditions

Problem:

If certain rare chip-internal conditions are met, the device will not start up correctly when executing a power-on reset. The crystal oscillator will be running but the device will not execute code.

Work-around:

Apply a second (warm) reset pulse (without power-on cycle). The minimum time requirement between the first (unsuccessful) reset and the second reset is 4105 external oscillator clock cycles, which means that the assertion of the second reset should occur 4105 cycles or more after the deassertion of the first reset. For example, at 10 MHz, this is equal to 411 μs; at 20 MHz, this is equal to 206 μs. This can be achieved by using an external watchdog timer or by any other circuitry in the application that is able to assert a second reset pulse.

The root cause for this problem has been identified and will be fixed in the LPC2210/01 version of this device which will have a dedicated order number (LPC2210FBD144/01).

ES_LPC2210

Errata sheet **Rev. 3 — 1 April 2011** **16 of 22**

3.21 ADC.7: External sync inputs not operational

Introduction:

In software-controlled mode (BURST bit is 0), the 10-bit ADC can start conversion by using the following options in the A/D Control Register:

26:24	START		When the BURST bit is 0, these bits control whether and when an ADC conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/CAP0.2 pin.	
		011	Start conversion when the edge selected by bit 27 occurs on P0.22/CAP0.0/MAT0.0 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1.	

Fig 1.   A/D control register options

Problem:

The external start conversion feature, ADCR:START = 0x2 or 0x3, may not work reliably and ADC external trigger edges on P0.16 or P0.22 may be missed. The occurrence of this problem is peripheral clock (pclk) dependent. The probability of error (missing a ADC trigger from GPIO) is estimated as follows:

- For PCLK_ADC = 60 MHz, probability error = 12 %
- For PCLK_ADC = 50 MHz, probability error = 6 %
- For PCLK_ADC = 12 MHz, probability error = 1.5 %

The probability of error is not affected by the frequency of ADC start conversion edges.

Work-around:

In software-controlled mode (BURST bit is 0), the START conversion options (bits 26:24 set to 0x1 or 0x4 or 0x5 or 0x6 or 0x7) can be used. The user can also start a conversion by connecting an external trigger signal to a capture input pin (CAPx) from a Timer peripheral to generate an interrupt. The timer interrupt routine can then start the ADC conversion by setting the START bits (26:24) to 0x1. The trigger can also be generated from a timer match register.

4. AC/DC deviations detail

4.1 V_{IH}.1: Incompatibility of actual V_{IH} levels as compared to those specified

Introduction:

The specified, minimum, value for V_{IH} is 2.0 V.

Problem:

Any pin associated with either an external interrupt input or an analog to digital converter (ADC) input has a V_{IH} of 2.4 V, not 2.0 V. 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 2.4 V at these pins.

4.2 V3.1: Leakage current on V3 due to External Interrupt and/or Analog to Digital Converter (ADC) pins.

Introduction:

V3 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 2.44 us.

Problem:

If the external interrupt and/or ADC pins are pulled higher than 1.8 V then it will lead to increased current consumption from V3. If V3 is 3.0 V and V1.8 is 1.8 V then the leakage current will increase to a typical number of 200 uA (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.

4.3 AINx.1: Corruption of an ADC conversion if any of the ADC input pins has a voltage higher than V3A

Introduction:

Analog input pins are multiplexed with GPIO pins which are 5 V tolerant.

Problem:

The result of an ADC conversion will be corrupted if any of the GPIO pins which have an analog input as an alternate function is connected to a voltage higher than V3A.

Work-around:

None

5. Errata notes detail

5.1 Note.1

Pin n.c. (pin 22, H2) must not be driven LOW during reset. If LOW on reset the device behavior is undetermined.

ES_LPC2210

Errata sheet **Rev. 3 — 1 April 2011** **19 of 22**

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.

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 accepts 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 national authorities.

6.3 Trademarks

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

7. Contents

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