Errata Sheet

Rel. 1.2, 27.11.2006

Device	**XC866L-4FR**
Marking/Step	**BC**
Package	**PG-TSSOP-38**

This Errata Sheet describes the deviations from the current user documentation.The module oriented classification and numbering system uses an ascending sequence over several derivatives, including already solved deviations. So gaps inside this enumeration can occur.

Table 1 Current Documentation

XC866 User's Manual	V1.2	June 2006
XC866 Data Sheet	V1.0	Feb 2006

Each erratum identifier follows the pattern Module_Arch.TypeNumber:

- **Module**: subsystem or peripheral affected by the erratum
- **Arch**: microcontroller architecture where the erratum was firstly detected.
  - **AI**: Architecture Independent (detected on module level)
  - **CIC**: Companion ICs
  - **TC**: TriCore (32 bit)
  - **X**: XC1xx / XC2xx (16 bit)
  - **XC8**: XC8xx (8 bit)
  - **none**: C16x (16 bit)
- **Type**: none - Functional Deviation; '**P**' - Parametric Deviation; '**H**' - Application Hint; '**D**' - Documentation Update
- **Number**: ascending sequencial number within the three previous fields. As this sequence is used over several derivatives, including already solved deviations, gaps inside this enumeration can occur.

Note: Devices marked with EES or ES are engineering samples which may not be completely tested in all functional and electrical characteristics, therefore they should be used for evaluation only.

The specific test conditions for EES and ES are documented in a separate Status Sheet.

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.1	03.11.2006	

Table 3 Errata fixed in this step

Errata	Short Description	Chg
ADC_XC8.003	Limitation during power-down mode	Fixed

Table 4 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
BRG_XC8.001	FDRES register is reloaded incorrectly		6
BROM_XC8.006	IRAM data is corrupted after any warm reset		6
BROM_XC8.010	SYSCON0.RMAP Switching Error		7
EVR_XC8.005	Reset toggling issue for repeated power up	New	8
INT_XC8.003	Unintended External Interrupt 0/1 Request in Bypass Edge Detection Mode		10
INT_XC8.004	Unable to Detect New Interrupt Request if Any One of Timer 2 Interrupt Flags Is Not Cleared (Unexpectedly)		10

Table 4 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
INT_XC8.005	Write to IRCON0 Blocks Interrupt Request of External Interrupt 0,1		13
INT_XC8.006	Write to IRCON1 Blocks Concurrent Hardware Set of EIR Flag		14
OCDS_XC8.002	Protect Mask Not Implemented		14
OCDS_XC8.006	Last Break Address May Cause Program Flash to Fail		15
OCDS_XC8.007	Watchdog Timer behavior during Debug		15
OCDS_XC8.009	Break while CPU is in NMI service mode		16
OSC_XC8.002	OSC_CON register is not reset following a wake-up reset		16
PLL_XC8.001	PLL N-Divider is reset to default value after a WDT reset		17
WDT_XC8.002	WDT should not be refreshed during Prewarning period		17
WDT_XC8.003	WDTRST bit cannot be set if software writes to PMCON0 register at the same time		18

Table 5 Deviations from Electrical- and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg
AC_XC8.001	Oscillator long term frequency deviation		19
DC_XC8.002	V_{DDP} 3.3V Range	Update	19
ESD_XC8.001	ESD Human Body Model robustness		20

Table 6 Application Hints

Hint	Short Description	Chg	Pg
ADC_XC8.H001	Arbitration mode when using external trigger at the selected input line REQTR		21
BROM_XC8.H001	SYSCON0.RMAP handling in ISR		21
EVR_XC8.H001	Consideration for Application with Fast Toggling Pin(s)		22
INT_XC8.H001	Interrupt Request With No Interrupt Flag Set		22
INT_XC8.H002	Effect of Interrupt Node Enable Bit on Interrupt Behavior		23
INT_XC8.H003	Interrupt Flags of External Interrupt 0 and 1		24
LIN_XC8.H001	LIN BRK field detection logic		25
LIN_XC8.H002	EOFSYN and ERRSYN bits status after exit from Boot ROM		26
OCDS_XC8.H002	Any NMI request is lost on Debug entry and during Debug		26

2 Functional Deviations

BRG_XC8.001 FDRES register is reloaded incorrectly

The reload of FDRES register happens when:

overflow occurs or

FDCON.FDEN bit is changed from 0 to 1

However, an unintended reload can also occur while FDEN bit is set. This happens whenever a write is carried out on FDCON register and as a result, FDEN bit is written again with 1, which reloads FDRES register.

In Normal Divider Mode, this can be caused by the clearing of the overflow flag (FDCON.NDOV) by software, which indirectly sets the FDEN bit. The user should not operate the BRG in Normal Divider Mode. If a timer function is required, the user is recommended to use any of the available Timer modules.

In Fractional Divider Mode, user should only write to FDCON when BRG is not running. If software writes to FDCON register while receiving a byte, the same unintended reload happens. However, the maximum error possible in this case is only $1/f_{MOD}$ per write access to FDCON register and therefore, should not lead to a wrong reception.

Workaround

None.

BROM_XC8.006 IRAM data is corrupted after any warm reset

After any warm reset (i.e. reset without powering off the device), boot up via User Mode affects certain IRAM data.

The affected IRAM address ranges are:

(1) 00_H - 09_H

(2) $3E_H$ - 40_H

(3) 61_H - 68_H

(4) 80_H - $0BF_H$

Workaround

None

BROM_XC8.010 SYSCON0.RMAP Switching Error

When executing from XRAM, if SYSCON0.RMAP is switched using an one-machine-cycle read-modify-write instruction (e.g. ORL dir,A) and the SFR is accessed immediately by an one-machine-cycle instruction (e.g. MOV A,dir) or a PUSH instruction, the SFR from the previous mapping might be accessed instead.

This RMAP switching error does not occur if code is executed from the Flash memory.

Workaround

When executing code from XRAM, use two-machine-cycle instructions to either switch RMAP or access the SFR. Alternatively, add one or more instructions (e.g. NOP) between the one-machine-cycle RMAP switching and SFR accessing instructions.

EVR_XC8.005 Reset toggling issue for repeated power up

Due to the limitation of the EVR, the reset toggling may occur during the power up. To prevent the reset toggling issue for power up, the following conditions must be met :

1. For power up, the rising time must be less than 2ms/V.
2. The power down time must be more than 0.2ms/V.
3. A permanent V_{DDP} residual voltage is less than 500mV.

In case of V_{DDP} residual voltage is more than 500mV, the holding time of residal voltage affects the reset toggling issue. **Figure 1** shows the critical cases for power up/power down when the residual voltages is held for some time. **Figure 2** shows the uncritical case for power up and power down when the power up is triggered immediately after power down. In case of a residual voltage due to immediate power up after power down, the users have to contact IFX to ensure no power up failure.

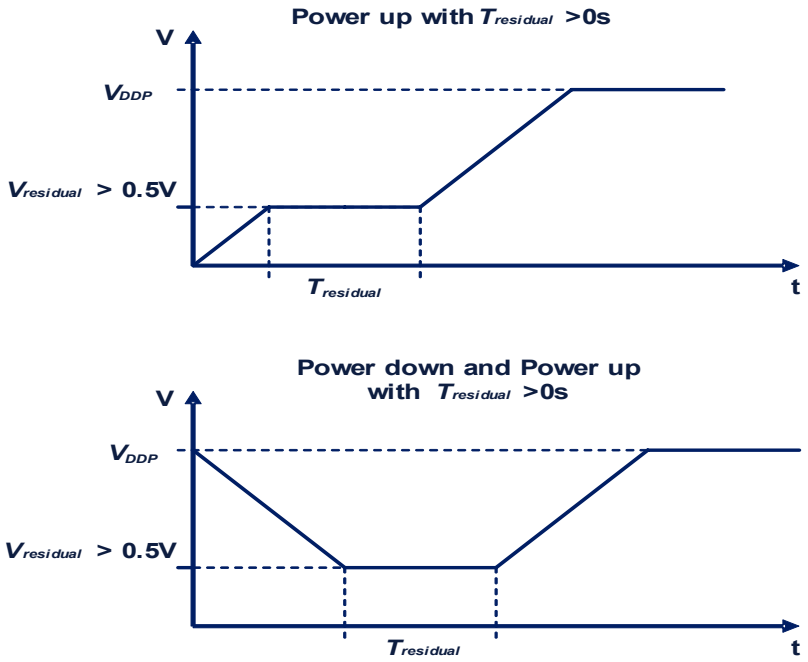

Figure 1 Critical Cases for Power up/ Power down when $V_{residual}$ >500mV

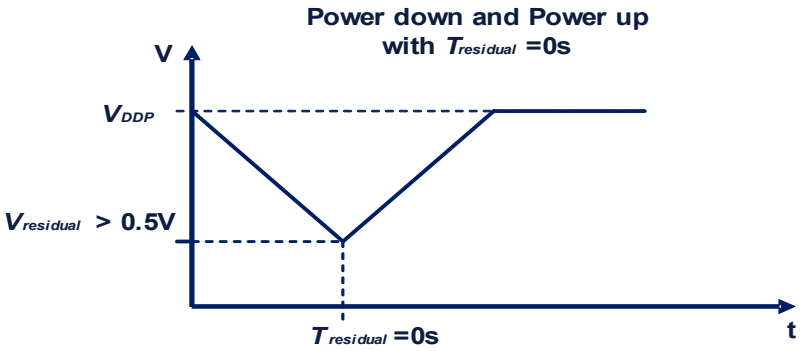

Figure 2 Uncritical Case for Power up/Power down when $V_{residual}$ >500mV

Workaround

None.

INT_XC8.003 Unintended External Interrupt 0/1 Request in Bypass Edge Detection Mode

For external interrupt 0 or 1 in bypass edge detection mode with falling edge as the active event, clearing the IRCON0 EXINT0/1 flag while the pin input is still low will lead to the internal circuitry falsely detecting a falling edge event. This will lead to unintended interrupt request.

Workaround

Never clear the IRCON0.EXINT0/1 flag in bypass edge detection mode. Access the TCON.IE0/1 flag directly instead, if necessary (TCON.IE0/1 flag is cleared automatically by hardware on vectoring to the interrupt routine).

INT_XC8.004 Unable to Detect New Interrupt Request if Any One of Timer 2 Interrupt Flags Is Not Cleared (Unexpectedly)

As illustrated in the simplified figure, the Timer 2 interrupt flags TF2 and EXF2 are combined as one interrupt request output. These flags are located within the Timer 2 kernel, with a single interrupt request line as output from the kernel.

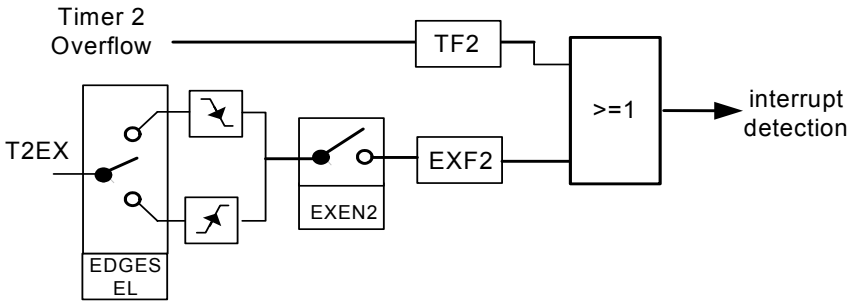

Figure 3

Being of interrupt structure 2, the interrupt request of Timer 2 is detected on the rising edge of a positive pulse.

The problem is that it may occur at some point in the application, that any new Timer 2 interrupt request can no longer be detected after a return-from-interrupt (reti) due to service of earlier event(s). This happens when the following conditions are true:

1. Timer 2 events are serviced by interrupt (i.e. flags are checked and cleared only in the interrupt routine ISR),
2. Either TF2 or EXF2 is set at any one time throughout the ISR (even while the other is cleared) such that at least one of the flags is still set after reti, causing the Timer 2 request line, which is an OR-function of the two flags TF2 and EXF2, to remain set throughout the ISR and after reti.

Two example scenarios are illustrated in the following figure:

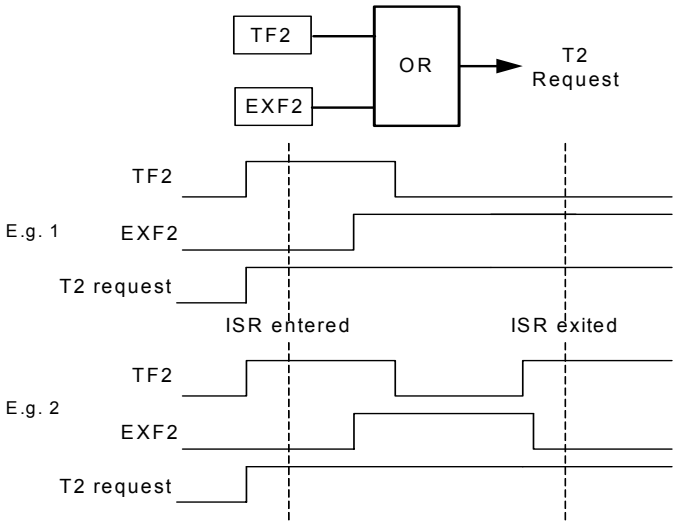

Figure 4

This means, any future Timer 2 TF2 or EXF2 event is not able to cause a rising edge and therefore not able to trigger an interrupt request to the core – as if the Timer 2 interrupts had been disabled, until both EXF2 and TF2 flags are cleared at some time. The clearing of flags would have to be done by user's code additionally outside of the Timer 2 interrupt routine, which is however normally not feasible with an interrupt service scheme.

Note: This condition affects only the detection of Timer 2 interrupt events TF2 and EXF2. It does not block the detection of other interrupt events belonging to the same interrupt node.

Workaround

There is no problem if EXF2 is not enabled by setting bit EXEN2 = 0 i.e. external events are disabled, or if bit DCEN = 1 in auto-reload mode.

Otherwise if EXF2 is enabled, while TF2 is always enabled when Timer 2 is running, there are two suggested workaround:

1. If other events of Timer 2 interrupt node (ET2) need not be enabled for interrupt, disable this interrupt node and use software polling of the flags instead.
2. Before return from interrupt, check again if TF2 or EXF2 is (still) set (due to new request since the last check). If so, jump and execute the ISR routine from start. Exit only when all flags are checked to be cleared. However, dummy interrupt of the node may occur after return from interrupt, and should be ignored. Another drawback is if Timer 2 events are occurring at high rate, the CPU may be 'stuck' in the service routine of the Timer 2 interrupt for a long time.

```
<entry point of interrupt node service routine>
.....
Start:
check flag TF2
.....
clear flag TF2
.....
check flag EXF2
.....
clear flag EXF2
.....
Finish:
if TF2 or EXF2 is set, jump to Start
reti (return from interrupt)
```

Figure 5

INT_XC8.005 Write to IRCON0 Blocks Interrupt Request of External Interrupt 0,1

Any write (read-modify-write or direct MOV) to the SFR IRCON0 will block an incoming interrupt request from external interrupt 0 or 1, even though the respective flag (EXINT0 or EXINT1) is set.

Workaround

After any write to the IRCON0, check (read IRCON0) the flags EXINT0 and EXINT1. If any flag is set, run the service routine; otherwise proceed.

In case of enabled for interrupt, the service routine should be duplicated: one copy as the interrupt service routine (with reti executed); another copy in main code memory for software call (with ret executed).

INT_XC8.006 Write to IRCON1 Blocks Concurrent Hardware Set of EIR Flag

Any write (read-modify-write or direct MOV) to the SFR IRCON1 will block a concurrent hardware set of the SSC error interrupt flag EIR (due to event happening). On the other hand, the interrupt request is triggered by the EIR event nevertheless.

Workaround

In the interrupt service routine, check and clear the kernel error source flags (TE, RE, PE and BE) instead of the EIR flag.

OCDS_XC8.002 Protect Mask Not Implemented

The protect mask feature is not implemented for the OCDS SFRs. This means it is possible for rwh SFR bits to not hold an updated (in a way, become corrupted) value under the following circumstances of an application: The hardware updates any rwh bit of a SFR which is in the midst of a read-modify-write operation.

The consequence is that the SFR bit update by hardware is lost, as the bit value is overwritten by the write-back operation.

Workaround

Software to take care to not include read-modify-write operations to OCDS SFRs which contain rwh bits. Use alternative instruction(s) instead.

OCDS_XC8.006 Last Break Address May Cause Program Flash to Fail

This problem occurs only if the last break address results in a return address in the range 0000_H to $00FF_H$.

In this case, the Flash bank where Flash wordline program is executed will hang in the program state while no Flash NMI is triggered. The contents of the Flash wordline may also be corrupted.

This renders the particular Flash bank unusable for code fetch or data read.

Workaround

When debugging code that calls Flash programming, do not break the CPU that results in the return address (from debug mode) to be in the range 0000_H to $00FF_H$.

Note: The return address is a function of the break address, refer to OCDS chapter of User's Manual.

OCDS_XC8.007 Watchdog Timer behavior during Debug

Firstly, the Watchdog Timer (WDT) cannot be refreshed by writing WDTCON.WDTRS in debug mode.

Secondly, on entering debug mode, the timer base counter which provide the clock to WDT is cleared to zero. The effect is that on exiting Monitor Mode in user mode, the WDT will count a little longer before overflow. This extra time is less than one WDT count.

Note: In debug mode, the WDT is suspended and stops counting.

Workaround

None. These WDT behavior occur only during debug.

OCDS_XC8.009 Break while CPU is in NMI service mode

While the CPU is in NMI service mode, the debug function has the following limitations:

1. External break via JTAG or MBC pin occurring while in NMI service mode, results in debug mode entry being postponed until NMI mode is ended.
2. Hardware or software breakpoints inside the NMI service routine will trigger the debug function. However, it is not possible to return to and execute user code (run or step) at the point of break as the debug cannot be exit.

Workaround

The workaround for the respective scenerios are

1. None. In fact, since NMI is usually a critical situation, any asynchronous break event postponed till after return from NMI mode has insignificant effects.
2. Do not set breakpoints or put the TRAP instruction inside the NMI service routine.

OSC_XC8.002 OSC_CON register is not reset following a wake-up reset

On a wake-up from power-down with reset, the register OSC_CON is not reset to its default value as specified. If an external oscillator is used, it will remain as the selected oscillator source on wake-up reset. On the other hand, the PLL settings including the NDIV value will be reset to the default state.

Depending on the external oscillator's frequency (f_{OSC}), there are three possible scenarios:

1. If $f_{OSC} > 12.5$ MHz, the system might freeze due to a system frequency that is too high.
2. If $9.375 \text{ MHz} \leq f_{OSC} \leq 12.5$ MHz, the system starts executing the user code correctly, even though it might run at a reduced frequency.
3. If $f_{OSC} < 9.375$ MHz, the system might freeze due to the inability of the PLL to lock.

Workaround

The wake-up from power-down with reset feature should not be used with an external oscillator outside the range of 9.375 to 12.5 MHz (scenarios 1 and 3). There is no impact on the user if the system is using the on-chip oscillator.

PLL_XC8.001 PLL N-Divider is reset to default value after a WDT reset

The user configured PLL N-divider value should be retained following a WDT reset. However in the current implementation, it is reset to its default value instead.

A problem arises when an external oscillator is used to generate the system clock. Depending on the external oscillator's frequency (f_{OSC}), there are three possible scenarios:

1. If $f_{OSC} > 12.5$ MHz, the system might freeze due to a system frequency that is too high.
2. If $9.375 \text{ MHz} \leq f_{OSC} \leq 12.5$ MHz, the system starts executing the user code correctly, even though it might run at a reduced frequency.
3. If $f_{OSC} < 9.375$ MHz, the system might freeze due to the inability of the PLL to lock.

Workaround

The WDT should not be used with an external oscillator outside the range of 9.375 to 12.5 MHz (scenarios 1 and 3). If the external oscillator falls within 9.375 to 12.5 MHz (scenario 2), the PLL N-divider must always be re-initialized in the user code with the required value after a WDT reset. There is no impact on the user if the system is using the on-chip oscillator.

WDT_XC8.002 WDT should not be refreshed during Prewarning period

If the WDT is not serviced before the timer overflows, a WDT NMI request is asserted and Prewarning is entered. During the Prewarning period, which last for 30_H

However in the current implementation, the WDT reset can be prevented by constantly writing to WDTCON.WDTRS during Prewarning.

Workaround

The user should not refresh the WDT upon entering the Prewarning period.

WDT_XC8.003 WDTRST bit cannot be set if software writes to PMCON0 register at the same time

When a software write access to register PMCON0 happens at the same time as a WDT reset, the WDT reset indication bit, PMCON0.WDTRST, may not be updated correctly.

Workaround

The Prewarning feature should always be used with the WDT NMI enabled (bit NMICON.NMIWDT is set). In the WDT NMI routine, care must also be taken not to write to the PMCON0 register.

3 Deviations from Electrical- and Timing Specification

AC_XC8.001 Oscillator long term frequency deviation

Table 7

Parameter	Symbol	Limit Values			Unit	Test Condition
		min.	typ.	max.		
Long term frequency deviation[1)]	Δf_{LT}	-5.0	-	5.0	%	with respect to f_{NOM}+ Δf_{CC}, over lifetime and temperature (-10 °C to 125 °C), for one device after trimming.
		-6.0	-	0	%	with respect to f_{NOM}+Δf_{CC}, over lifetime and temperature (-40 °C to -10 °C), for one device after trimming.

1) Not all parameters are 100% tested, but are verified by design/characterisation and test correlation.

Workaround

None.

DC_XC8.002 V_{DDP} 3.3V Range

When ambient temperature is below 0°C, and when V_{DDP} is 3.6 V or below, V_{DDC} may drop below 2.3 V. As this V_{DDC} voltage level is out of the specified operating range, the system may not work properly.

Workaround

Do not operate these devices at the V_{DDP} = 3.3 V range (3.6 V or below), when the ambient temperature is below 0°C.

ESD_XC8.001 ESD Human Body Model robustness

ESD susceptibility according to HBM is up to 1kV limit.

ESD HBM is up to 600V on VDDC pin

The rest of pins are up to 2KV.

Workaround

None.

4 Application Hints

ADC_XC8.H001 Arbitration mode when using external trigger at the selected input line REQTR

If an external trigger is expected at the selected input line REQTR to trigger a pending request, the arbitration mode should be set (PRAR.ARBM=1) where the arbitration is started by pending conversion request. This selection will minimize the jitter between asynchronous external trigger with respect to the arbiter and the start of the conversion. The jitter can only be minimized while no other conversion is running and no higher priority conversion can cancel the triggered conversion. In this case, a constant delay (no jitter) has to be taken into account between the trigger event and the start of the conversion.

BROM_XC8.H001 SYSCON0.RMAP handling in ISR

The ISR has to handle SYSCON0.RMAP correctly when Flash user routines provided in the Boot ROM are used together with the interrupt system. Any ISR with the possibility of interrupting these user routines has to do the following in the interrupt routine:

save the value of the RMAP bit at the beginning

restore the value before the exit

This is to prevent access of the wrong address map upon return to the Flash user routine since the RMAP bit may be changed within the interrupt routine. The critical point is when Flash user routines sets RMAP to ‘1’ and the interrupt occurs that needs RMAP at ‘0’ in the ISR.

Please note that NMI is an interrupt as well.

EVR_XC8.H001 Consideration for Application with Fast Toggling Pin(s)

When the pins of the chip are toggling at a relatively high rate, the EVR functionality may be affected due to injected noise to the V_{DDP}. In order to increase robustness against V_{DDP} noise, it is recommended to use a fast ceramic blocking capacitor (100nF - 300nF) between V_{DDP} and V_{SSP}. In addition, it is also essential to put it as close as possible to the chip.

INT_XC8.H001 Interrupt Request With No Interrupt Flag Set

It might occur in the application, that sometimes interrupt requests are serviced, but no active interrupt flags are found. This would happen for those interrupt nodes which is shared by several interrupt sources.

Consider the following interrupt service routine pseudo-code, and scenerio where a interrupt source A event leads to interrupt request of the node and CPU vectors to the interrupt routine. Meanwhile, interrupt source B and its flag becomes active any time in the duration indicated by T:

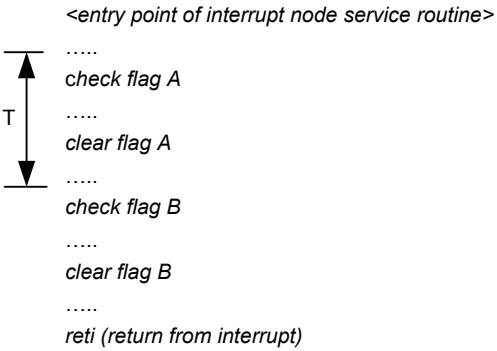

Figure 6

In this case, flag B will be cleared as a standard procedure by the interrupt routine in the current service of the interrupt node. However, the pending

interrupt request for the node remains activated after RETI, as it is only cleared by hardware when CPU acknowledge the interrupt and vectors to the interrupt routine.

This leads to following servicing of the interrupt node again, but potentially no active flag is found, i.e. dummy interrupt service. The point to note is that the interrupt source B is not lost, as it was actually serviced in the current service of the interrupt node.

The recommendation is to ignore these dummy interrupt vectoring.

INT_XC8.H002 Effect of Interrupt Node Enable Bit on Interrupt Behavior

A-Step

When the interrupt event occurs while the respective interrupt node enable bit (`IENx.y`) is disabled, the interrupt status flag will be set and the interrupt request will be captured. Hence, enabling the respective `IENx.y` bit later will result in the CPU servicing of the interrupt node.

It is not possible to clear any pending interrupt request by software. Refer to INT_XC8.001 on the Dummy Interrupt issue for further detail.

B-Step

When the interrupt event occurs while the respective `IENx.y` bit is disabled, the interrupt status flag will be set but the interrupt request will not be captured. Hence, enabling the respective `IENx.y` bit later will not result in the CPU servicing the interrupt node for the prior interrupt event.

Clearing the `IENx.y` bit of a node will clear its pending interrupt request.

In both steps, the interrupt enable bit (`EA`) may be used to globally disable all interrupts. All incoming interrupt requests will be captured in respective interrupt nodes provided the `IENx.y` bit is enabled. All interrupt requests will be serviced according to priority when `EA` bit is enabled.

INT_XC8.H003 Interrupt Flags of External Interrupt 0 and 1

External interrupt 0 and 1 may individually be selected via respective bits (EXINTx) in EXICON0 register, to request interrupt on falling edge, rising edge, both edges or to bypass the edge detection.

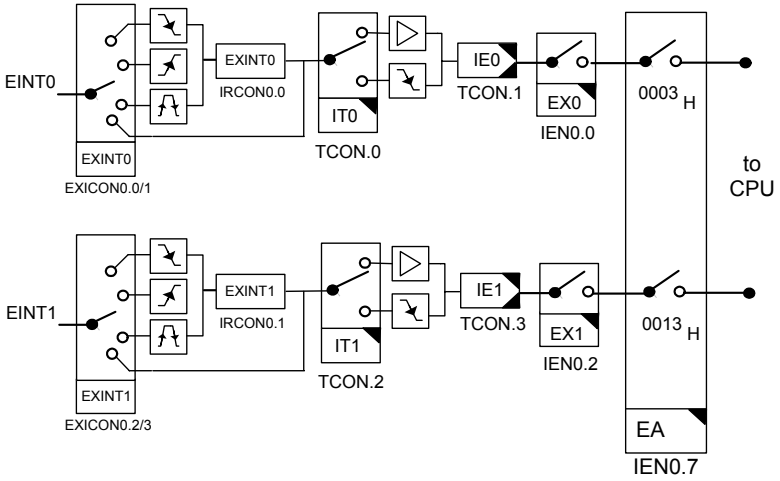

Figure 7

Edge detection is done in the system unit. If enabled, an active event will set the EXINTx flag and correspondingly set the IEx flag in TCON. It should be noted that after any external interrupt x event, flag EXINTx must be cleared. In case of falling edge as active event, this allows any future active event to be able to set the flag IEx as interrupt request. In case of low level as active event, this prevents unintended recurring triggering of interrupt request.

Besides the above notes, the following should be noted on the behavior regarding setting and clearing of the external interrupt x (x = 0 or 1) flags, applicable to both edge and bypass edge detection modes:

Setting of External Interrupt x Flags

1. The flag TCON.IEx will be set in all modes selectable via EXICON0 register.
2. Flag IRCON0.EXINTx will be set in all modes as long as an active edge is detected; flag EXINTx will not be set for low level as active event.

Clearing of External Interrupt x Flags

1. Flag IEx is cleared automatically by hardware when the interrupt is being vectored to.
2. Flag EXINTx has to be cleared by software.
3. Clearing one external interrupt x flag will not clear the other. Especially, clearing flag EXINTx will not clear the flag IEx. Being of interrupt structure 1, the flag IEx is the request polled by the CPU for interrupt servicing. Therefore user has to take care to clear the flag IEx before switching from SW polling method to enabling the external interrupt x node, to prevent potential dummy interrupt request.
4. Always clear both EXINTx and IEx flags before (if) changing the trigger select in EXICON0 register.

LIN_XC8.H001 LIN BRK field detection logic

Based on the hardware implementation, the maximum number of bits in the BRK field must follow the formula:

$$\text{Maximum number of bits in BRK field} = \text{Baud Rate x} \frac{4095}{\text{Sample Frequency}}$$

$$\text{Sample Frequency} = \frac{\text{PCLK}}{8 \times 2^{\text{BGSEL}}}$$

For example, if LIN baudrate is 19.2kbps, BGSEL = 0 and CPU frequency is 26.67MHz, the maximum number of bits in BRK field would be:

19.2k x 4095 / ((26.67M / 8)) = ~23.6 bits

If the maximum number of bits in the BRK field exceeded, the internal counter will overflow which results in baudrate detection error. Therefore, the user is advised to choose the appropriate BGSEL value for the required baudrate detection range.

The calculated value above does not consider sample error and transmission error, nevertheless it can be used as a guideline.

LIN_XC8.H002 EOFSYN and ERRSYN bits status after exit from Boot ROM

When the device exits from Boot ROM and starts user code execution (either in XRAM or flash memory), the bitfields EOFSYN and ERRSYN in SFR FDCON remain in their last accessed states. Therefore these bitfields must be initialized according to the application requirement by the user code.

The following are the possible application scenarios that could lead to the above:

1) When any LIN BSL supported modes is activated before the device starts to execute user code (either in XRAM or flash memory).

2) After the device is reset and remain in the LIN BSL, if there is any error frame detected in the LIN bus, the user code in the flash memory will be executed.

User code should initialize the register properly according to the application requirement.

OCDS_XC8.H002 Any NMI request is lost on Debug entry and during Debug

All NMI events are disabled while in debug mode. This has two main effects:

1. On debug entry, any pending NMI request will be lost, although the status flag remains set. The probability of losing an NMI request in this way is very low, since NMI always has the highest priority to be serviced.

2. Any NMI event that occurs during debugging is not able to generate an NMI request (event interrupt is lost) although the status flag will be set. It is normally not critical that on exit from debug mode, the CPU must service NMI requests that had occurred while in debug mode.

The fact that the debug system is not specified to support NMI interrupt while in debug mode makes the above trivial. As precaution, avoid starting any debug session while expecting an NMI event.