Errata Sheet

Rel. 1.3, 28.02.2011

Device	**XC866L-1FR Series**
Marking/Step	**AB**
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.

This Errata Sheet covers the following devices:

- XC866-1FR
- XC866L-1FR

Table 1 Current Documentation

XC866 User's Manual	V1.3	Feb 2007
XC866 Data Sheet	V1.2	Oct 2007
SAA-XC866 Data Sheet	V1.5	Sep 2010

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 / XC2000 (16 bit)
  - **XC8**: XC800 (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
1.0	02.03.2007	
1.1	23.11.2007	
1.2	13.02.2009	

Table 3 Errata fixed in this step

Errata	Short Description	Chg
DC_XC8.002	V_{DDP} 3.3V Range	Fixed
EVR_XC8.005	Reset toggling issue for repeated power up	Fixed
OCDS_XC8.009	Break while CPU is in NMI service mode	Fixed

Table 4 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
BROM_XC8.006	IRAM data is corrupted after any warm reset		7
BROM_XC8.010	SYSCON0.RMAP Switching Error		7
INT_XC8.003	Unintended External Interrupt 0/1 Request in Bypass Edge Detection Mode		8

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
INT_XC8.004	Unable to Detect New Interrupt Request if Any One of Timer 2/Timer 21/UART1 Interrupt Flags Is Not Cleared (Unexpectedly)		8
INT_XC8.005	Write to IRCON0 Blocks Interrupt Request of External Interrupt 0,1		11
INT_XC8.006	Write to IRCON1 Blocks Concurrent Hardware Set of EIR Flag		12
LIN_XC8.001	Fast LIN BSL does not support baud rate of up to 115.2 kHz		12
OSC_XC8.002	OSC_CON register is not reset following a wake-up reset		12
PIN_XC8.007	External pull-up device is required on MBC pin to enter user mode		13
PLL_XC8.001	PLL N-Divider is reset to default value after a WDT reset		13
PM_XC8.003	Wake-up triggered before power down sequence completed	New	14
SYS_XC8.001	MOV (direct, direct) instruction might cause a wrong value to be written to the destination register		14
T2_XC8.001	Timer 2 is not suspended during debug in counter mode		20
WDT_XC8.002	WDT should not be refreshed during Prewarning period		20
WDT_XC8.003	WDTRST bit cannot be set if software writes to PMCON0 register at the same time		20

Table 5 Deviations from Electrical- and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg

Table 6 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H001	Arbitration Mode with disabled Arbitration Slots	New	22
ADC_XC8.H001	Arbitration mode when using external trigger at the selected input line REQTR		22
BROM_XC8.H001	SYSCON0.RMAP handling in ISR		23
CCU6_XC8.H002	CCU6 PM event in center-aligned mode	New	23
INT_XC8.H001	Interrupt Request With No Interrupt Flag Set		25
INT_XC8.H003	Interrupt Flags of External Interrupt 0 and 1		26
LIN_XC8.H001	LIN BRK field detection logic		27
LIN_XC8.H002	LIN Break/Synch field detection	Update	28
OCDS_XC8.H002	Any NMI request is lost on Debug entry and during Debug		28
PIN_XC8.H001	Current over GPIO pin must not source V_{DDP} higher than 0.3V		29
PLL_XC8.H001	Check oscillator run bit 2048 VCO cycles after oscillator run detection is restarted.		29
SYS_XC8.H001	Usage of the Bit Protection Scheme		29
SYS_XC8.H003	Effective write for Read-Modify-Write instructions of two bytes, one machine cycle	New	30

Table 6 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
SYS_XC8.H004	Switch PLL to prescaler mode before a WDT reset	New	31
T2_XC8.H002	External Input in Timer 2 Capture Mode		31

2 Functional Deviations

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.

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/Timer 21/UART1 Interrupt Flags Is Not Cleared (Unexpectedly)

Note: The modules Timer 21 and UART1 are not available in the device. Therefore, the errata describes only for the case of Timer 2.

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.

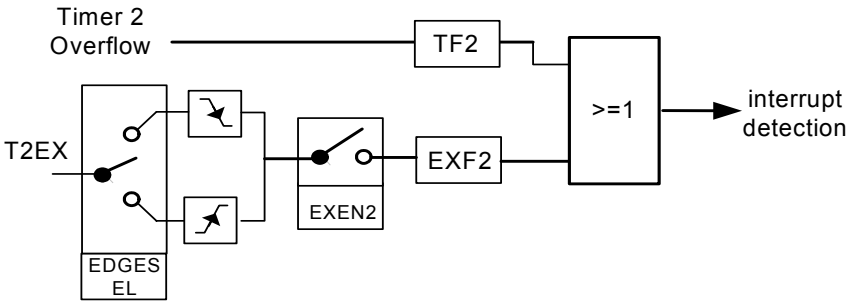

Figure 1

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:

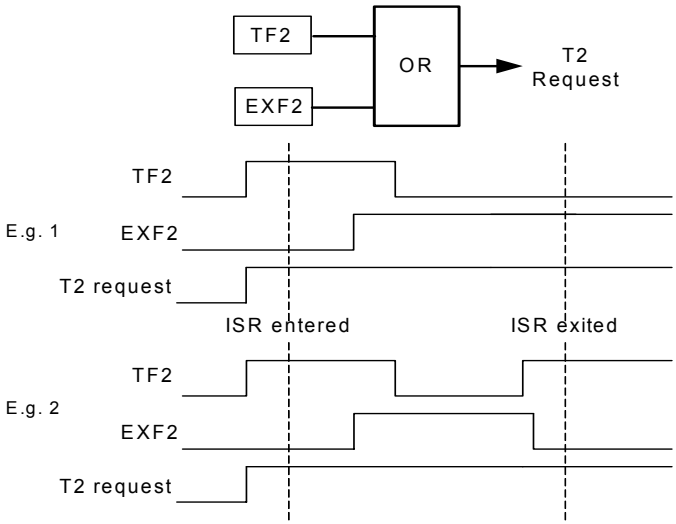

Figure 2

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 3

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.

LIN_XC8.001 Fast LIN BSL does not support baud rate of up to 115.2 kHz

Fast LIN BSL supports only a baud rate of up to 57.6 kHz and not up to 115.2 kHz as described in the user's manual.

Workaround

None

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 two possible scenarios:

1. If 9.375 MHz ≤ f_{OSC} ≤ 12 MHz, the system starts executing the user code correctly, even though it might run at a reduced frequency.
2. If f_{OSC} < 9.375 MHz, the system might hang in an endless loop 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 MHz. There is no impact on the user if the system is using the on-chip oscillator.

PIN_XC8.007 External pull-up device is required on MBC pin to enter user mode

Although MBC pin is specified to be pull-up in the reset state, an external pull-up device is still required to enter user mode.

Workaround

Implement external pull-up (in the range of 4.7 kΩ to 100 kΩ) on the external circuitry for MBC pin. Alternatively, MBC pin could be tied to high.

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 9.375 MHz ≤ f_{OSC} ≤ 12 MHz, the system starts executing the user code correctly, even though it might run at a reduced frequency.
2. If f_{OSC} < 9.375 MHz, the system might hang in an endless loop 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 MHz (scenario 2). If the external oscillator falls within 9.375 to 12 MHz (scenario 1), 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.

PM_XC8.003 Wake-up triggered before power down sequence completed

Setting the PD bit in PMCON0 register will cause the device to go into power down mode. When this bit is set, flash memory will start the powered down sequence immediately. If a wakeup event happens before flash powers down completely, the device will not be able to wake-up normally. The flash memory requires the following maximum time to complete power down:

- 160 usec if the flash is in program or erase mode
- 350 nsec if the flash is in read or idle mode

Workaround

None.

SYS_XC8.001 MOV (direct, direct) instruction might cause a wrong value to be written to the destination register

The MOV (direct, direct) instruction (hex code 85_H) that access registers (direct address ranging from 80_H to FF_H), does not write the correct value of the source register to the destination register if the destination register is a register listed in the table below.

The source register can be any register from the direct address range 80_H to FF_H.

Table 7

Module	Register	SFR Address	RMAP	Page	Products Affected
SCU	IRCON0	$B4_H$	0	0	XC88x, XC878
	IRCON1	$B5_H$	0	0	XC88x, XC878
	IRCON2	$B6_H$	0	0	XC88x, XC878
	IRCON3	$B4_H$	0	3	All
	IRCON4	$B5_H$	0	3	All
	NMISR	BC_H	0	0	XC88x, XC878
	FDCON	$E9_H$	0	0	XC88x, XC878
	PMCON0	$B4_H$	0	1	XC88x, XC878
	OSC_CON	$B6_H$	0	1	XC88x, XC878
	PLL_CON	$B7_H$	0	1	XC88x
	MISC_CON	$E9_H$	0	1	XC88x, XC878
WDT	WDTCON	BB_H	1	-	XC88x, XC878
CORDIC	CD_STATC	$A0_H$	1	-	XC88x, XC878
MDU	MDUSTAT	$B0_H$	1	-	XC88x, XC878
SSC	CONH (Operating Mode)	AB_H	0	-	All
UART1	SCON	$C8_H$	1	-	XC88x, XC878
	FDCON	CC_H	1	-	XC88x, XC878
T2	T2CON	$C0_H$	0	-	All
T21	T2CON	$C0_H$	1	-	XC88x, XC878
OCDS	MMCR2	$E9_H$	1	-	All
	MMCR	$F1_H$	1	-	All
	MMSR	$F2_H$	1	-	All
	MMICR	$F4_H$	1	-	All

Table 7

Module	Register	SFR Address	RMAP	Page	Products Affected
T2CCU	CCTCON	$C6_H$	0	1	XC878
	COSHDW	$C0_H$	0	2	XC878
	COCON	$C0_H$	0	3	XC878
	CC0L	$C1_H$	0	2	XC878
	CC0H	$C2_H$	0	2	XC878
	CC1L	$C3_H$	0	2	XC878
	CC1H	$C4_H$	0	2	XC878
	CC2L	$C5_H$	0	2	XC878
	CC2H	$C6_H$	0	2	XC878
	CC3L	$C1_H$	0	3	XC878
	CC3H	$C2_H$	0	3	XC878
	CC4L	$C3_H$	0	3	XC878
	CC4H	$C4_H$	0	3	XC878
	CC5L	$C5_H$	0	3	XC878
	CC5H	$C6_H$	0	3	XC878
CCU6	CC63SRL	$9A_H$	0	0	All
	CC63SRH	$9B_H$	0	0	All
	MCMOUTSL	$9E_H$	0	0	All
	MCMOUTSH	$9F_H$	0	0	All
	CC60SRL	FA_H	0	0	All
	CC60SRH	FB_H	0	0	All
	CC61SRL	FC_H	0	0	All

Table 7

Module	Register	SFR Address	RMAP	Page	Products Affected
CCU6 (cont'd)	CC61SRH	FD_H	0	0	All
	CC62SRL	FE_H	0	0	All
	CC62SRH	FF_H	0	0	All
	T12PRL	$9C_H$	0	1	All
	T12PRH	$9D_H$	0	1	All
	T13PRL	$9E_H$	0	1	All
	T13PRH	$9F_H$	0	1	All
	T12DTCL	$A4_H$	0	1	All
	T12DTCH	$A5_H$	0	1	All
	TCTR0L	$A6_H$	0	1	All
	TCTR0H	$A7_H$	0	1	All
	T12MSELL	$9A_H$	0	2	All
	T12MSELH	$9B_H$	0	2	All
	IENL	$9C_H$	0	2	All
	IENH	$9D_H$	0	2	All
	INPL	$9E_H$	0	2	All
	INPH	$9F_H$	0	2	All
	PSLR	$A6_H$	0	2	All
	MCMCTR	$A7_H$	0	2	All
	TCTR2L	FA_H	0	2	All
	TCTR2H	FB_H	0	2	All
	MODCTRL	FC_H	0	2	All
	MODCTRH	FD_H	0	2	All
	TRPCTRL	FE_H	0	2	All
	TRPCTRH	FF_H	0	2	All
	PISEL0L	$9E_H$	0	3	All
	PISEL0H	$9F_H$	0	3	All
	PISEL2	$A4_H$	0	3	All

Table 7

Module	Register	SFR Address	RMAP	Page	Products Affected
CCU6 (cont'd)	T13L	FC_H	0	3	All
	T13H	FD_H	0	3	All
	CMPSTATH	FF_H	0	3	All
ADC	GLOBCTR	CA_H	0	0	All
	PRAR	CC_H	0	0	All
	LCBR	CD_H	0	0	All
	INPCR0	CE_H	0	0	All
	ETRCR	CF_H	0	0	All
	CHCTR0	CA_H	0	1	All
	CHCTR1	CB_H	0	1	All
	CHCTR2	CC_H	0	1	All
	CHCTR3	CD_H	0	1	All
	CHCTR4	CE_H	0	1	All
	CHCTR5	CF_H	0	1	All
	CHCTR6	$D2_H$	0	1	All
	CHCTR7	$D3_H$	0	1	All
	RCR0	CA_H	0	4	All
	RCR1	CB_H	0	4	All
	RCR2	CC_H	0	4	All
	RCR3	CD_H	0	4	All
	CHINPR	CD_H	0	5	All
	EVINPR	$D3_H$	0	5	All
	CRCR1	CA_H	0	6	All
	CRPR1	CB_H	0	6	All
	CRMR1	CC_H	0	6	All
	QMR0	CD_H	0	6	All

For example, in the sample code below, there are two MOV (direct, direct) instructions that write the value of one register into another. All the source and destination registers in these two instructions are from the direct address range 80_H to FF_H.

The P1_DATA register is not one of the affected registers listed in the table above and therefore, it is written with the correct value of the CC60SRL register. On the other hand, the CC60SRH register is one of the affected registers and therefore, it is written with the wrong value of the B register.

Sample Code:

```
interrupt:
MUL A, B
MOV CC60SRL, A
MOV P1_DATA, CC60SRL
MOV CC60SRH, B
RETI
```

Workaround

Instead of using the MOV (direct, direct) instruction, use other instructions or an intermediate variable to write to the targeted register.

For example, the two MOV (direct, direct) instructions in the earlier sample code can be replaced with MOV (direct, A) instructions (hex code $F5_H$). Both the P1_DATA and CC60SRH registers will now be written with the correct source register values.

Sample Code:

```
interrupt:
MUL A, B
MOV CC60SRL, A
MOV P1_DATA, A
XCH A, B
MOV CC60SRH, A
RETI
```

T2_XC8.001 Timer 2 is not suspended during debug in counter mode

Timer 2 is not suspended (even if Timer 2 suspend control is enabled by setting bit MODSUSP.T2SUSP) during debug if Timer 2 is set to counter mode.

Workaround

None.

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, refreshing of the Watchdog Timer should not be possible and the microcontroller is expected to reset at the end of this period.

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 WDT Prewarning feature should always be enabled (bit `NMICON.NMIWDT` is set to enable WDT NMI) so that the WDT NMI routine is always entered before a WDT reset. In the WDT NMI routine, care must be taken not to write to the `PMCON0` register.

3 Deviations from Electrical- and Timing Specification

4 Application Hints

ADC_AI.H001 Arbitration Mode with disabled Arbitration Slots

In arbitration mode (bit ARBM = 1_B in register PRAR), the arbiter only runs while at least one conversion request is pending, otherwise it waits in an idle state for a request to become active. This leads to a constant and reproducible latency from an incoming request to the conversion start.

Each request source x (x = 0, 1) can be individually selected (via the Arbitration Slot Enable bits ASENx in register PRAR) to take part in the arbitration round. However, if a disabled request source (bit ASENx = 0_B) has a pending request, the related conversion is not started, but the arbiter does not stop and wait. As a result, the latency for requests generated on other arbitration slots is not constant.

To avoid this effect, first disable the request generation of the respective source by setting bit ENGT = 0_B in the corresponding register QMR0 or CRMR1 before disabling the arbitration slot via ASENx = 0_B.

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.

CCU6_XC8.H002 CCU6 PM event in center-aligned mode

After detecting a period match (PM A) in centre-aligned mode, T12 counts down from PM + 1 as shown below:

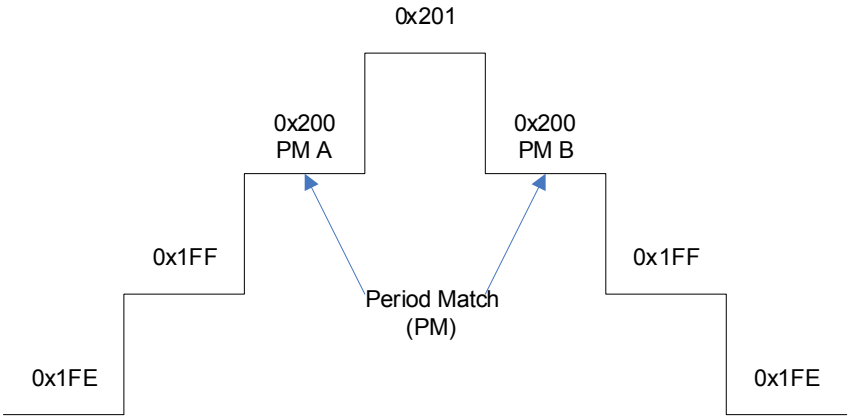

Figure 4 Counting sequence of T12 in center-aligned mode

This means a second PM event (PM B) will occur during the counting down. If ADC is triggered externally via ETRx2 (T12PM), it will be triggered twice in succession. Depending on how real-time the application code is running as well as the T12 count rate and ADC conversion rate, the application could observe two ADC interrupts - once at PM A and once at PM B.

To avoid triggering twice the ADC interrupts, it is suggested to use ETRx6 from multi-channel mode instead of ETRx2 as the trigger source for ADC. Additional initialization are as follows:

- Configure MCMCTR.SWSEL = 101_B (Transfer on T12 period match)
- Configure MCMCTR.SWSYN = 00_B (Direct transfer)
- Write to MCMOUTSTL = CF_H (To enable multi-Channel PWM pattern on CC6x and COUT6x)

Note: Independent of the external trigger, the CCU6 internal triggers based on T12 PM (e.g. T12 PM interrupt or shadow transfer) are only activiated once while T12 is counting up.

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:

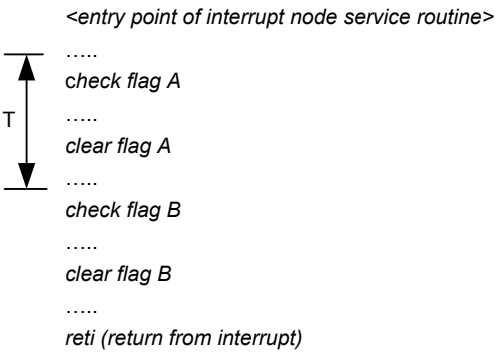

Figure 5

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.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.

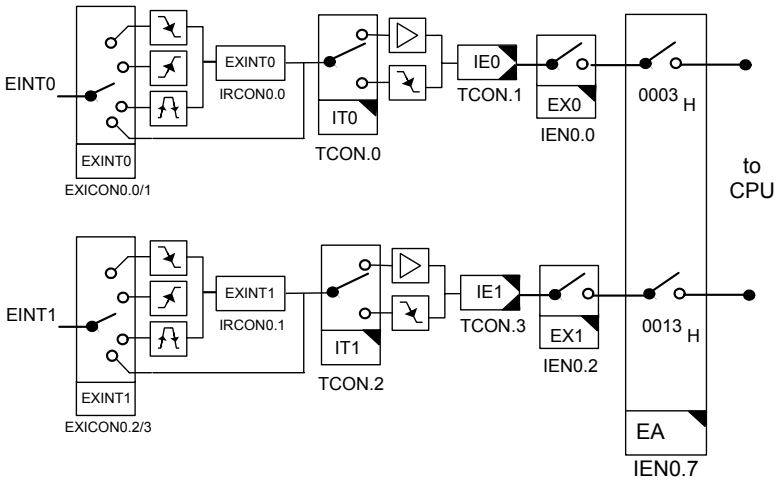

Figure 6

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.

In case of bypass edge detection, the input is connected directly to the core (bypass the system unit). The flag IEx in TCON will be set on the selected falling edge or low level (at least 2 clock cycles) event. In case of falling edge selected as active event by TCON.ITx, the flag EXINTx is set in addition to the flag IEx. However to use this mode properly, user must not clear the flag EXINTx and if necessary, to only access the flag TCON.IEx (refer to INT_XC8.003).

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:

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

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

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 LIN Break/Synch field detection

The LIN Break/Synch field detection is default enabled after every reset (`BCON.BRDIS` bit = 0). However, to ensure the first standard LIN frame can always be detected, it is recommended to initialize the detection logic in the user code before receiving the frame. This is through the following two steps:

1. Toggle `BCON.BRDIS` bit (set the bit to 1 before clearing it back to 0).
2. Clear the three status flags `FDCON.BRK`, `FDCON.EOFSYN` and `FDCON.ERRSYN` to 0.

It is also recommended to toggle the `BCON.BRDIS` bit after the reception of each complete LIN frame to avoid a wrong Break field detection in noisy environments (i.e. spikes on the LIN bus).

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.

PIN_XC8.H001 Current over GPIO pin must not source V_{DDP} higher than 0.3V

When V_{DDP} is not powered on, the current over a GPIO pin has to be limited in such a way that V_{DDP} - $V_{SSP} \leq 0.3V$. This prevents the supply of the device via the ESD diode between the GPIO pin and V_{DDP}.

However, for applications with strict low power-down current requirements, it is mandatory that no active voltage source is supplied at any GPIO pin when V_{DDP} is not powered on.

PLL_XC8.H001 Check oscillator run bit 2048 VCO cycles after oscillator run detection is restarted.

When performing a loss-of-lock recovery or changing to an external oscillator, one of the steps required is to restart the oscillator run detection logic by setting `OSC_CON.ORDRES` bit to 1.

After the oscillator run detection logic is restarted, the user should wait for minimum 2048 VCO cycles (approximately between 30 µs and 200 µs depending on VCO base frequency) before checking the status of the oscillator run bit `OSC_CON.OSCR`.

SYS_XC8.H001 Usage of the Bit Protection Scheme

When the bit protection scheme is enabled, bit field `PASSWD.PASS` should always be used to open and close write access to the protected bits. The scheme should be disabled only if it is not required in the application.

In the unlikely event that the scheme is enabled again after disabling it while the write access is still open, the write access will remain open until the count of 32 CCLK cycles is completed.

SYS_XC8.H003 Effective write for Read-Modify-Write instructions of two bytes, one machine cycle

When read-modify-write instructions requiring 2 bytes and 1 machine cycle (equivalent to 2 CCLK cycles) for execution, such as INC dir, are executed from memories without any wait states[1)], the actual write to the destination is delayed by the internal bus for up to one CCLK cycle. This means that even though the CPU completes the instruction execution after 2 CCLK cycles, the write through the internal bus may take effect only after a further CCLK cycle.

The list of affected read-modify-write instructions is shown below:

Table 8

Mnemonic	Hex Code	Bytes	No. of CCLK cycles (without wait states)
INC dir	05	2	2
DEC dir	15	2	2
ANL dir, A	52	2	2
ORL dir, A	42	2	2
XRL dir, A	62	2	2
XCH A, dir	C5	2	2
CLR bit	C2	2	2
SETB bit	D2	2	2
CPL bit	B2	2	2

1) Applicable also to Flash memory with parallel read feature.

SYS_XC8.H004 Switch PLL to prescaler mode before a WDT reset

A WDT reset does not reset the clock system. If a WDT reset occurs while the PLL is in base mode (PLL_CON.OSCDISC = 1), the system will be held in reset until the next power-on or hardware reset. This is because the system requires the PLL to be locked (PLL_CON.LOCK = 1) or in the prescaler mode (PLL_CON.OSCDISC = 0; PLL_CON.VCOBYP = 1) before the reset can be released.

If the above behaviour is not desired, it is recommended to switch the PLL to prescaler mode, and with the on-chip oscillator as the input clock source (OSC_CON.OSCSS = 0), whenever a WDT prewarning is entered.

T2_XC8.H002 External Input in Timer 2 Capture Mode

In Timer 2 capture mode, the external input is sampled in every PCLK cycle. When a sampled input shows a low (high) level in one PCLK cycle and a high (low) in the next PCLK cycle, a transition is recognized. Therefore, the frequency of input signal on Pin T2EX must be lower than PCLK/2.