Errata Sheet

Rel. 1.2, 17.01.2013

Device	**XC878CLM Series**
Marking/Step	**AC**
Package	**PG-LQFP-64**

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:

- XC878CM-13/16FF
- XC878CLM-13/16FF

Table 1 Current Documentation

XC878CLM User's Manual	V1.1	Apr 2009
XC87xCLM Data Sheet	V1.5	Mar 2011

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	11.09.2009	
1.1	26.05.2010	

Table 3 Errata fixed in this step

Errata	Short Description	Chg
T2CCU_XC8.001	External Trigger of ADC when CCT of T2CCU overflows	Fixed

Table 4 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
BROM_XC8.006	IRAM data is corrupted after any warm reset		7
CD_XC8.001	Set and Clear of Error Bit in CORDIC Linear Vectoring Mode		7
CD_XC8.002	Data Fetch to CD_STATC Register may capture an incorrect error status		8
MultiCAN_AI.040	Remote frame transmit acceptance filtering error		8
MultiCAN_AI.041	Dealloc Last Obj		9

Table 4 **Functional Deviations** (cont'd)

Functional Deviation	Short Description	Chg	Pg
MultiCAN_AI.042	Clear MSGVAL during transmit acceptance filtering		9
MultiCAN_AI.043	Dealloc Previous Obj		10
MultiCAN_AI.044	RxFIFO Base SDT		11
MultiCAN_AI.045	OVIE Unexpected Interrupt		11
MultiCAN_AI.046	Transmit FIFO base Object position		11
MultiCAN_TC.025	RXUPD behavior		12
MultiCAN_TC.026	MultiCAN Timestamp Function		12
MultiCAN_TC.027	MultiCAN Tx Filter Data Remote		13
MultiCAN_TC.028	SDT behavior		13
MultiCAN_TC.029	Tx FIFO overflow interrupt not generated		14
MultiCAN_TC.030	Wrong transmit order when CAN error at start of CRC transmission		16
MultiCAN_TC.031	List Object Error wrongly triggered		16
MultiCAN_TC.032	MSGVAL wrongly cleared in SDT mode		17
MultiCAN_TC.035	Different bit timing modes		17
MultiCAN_TC.037	Clear MSGVAL		19
MultiCAN_TC.038	Cancel TXRQ		20
SYS_XC8.001	MOV (direct, direct) instruction might cause a wrong value to be written to the destination register		20
SYS_XC8.003	Brownout Reset	New	26
T2CCU_XC8.003	T2CCU Capture Functions		26
UART_XC8.002	Bits FDEN and FDM in UART1_FDCON SFR cannot be Written by Read-Modify-Write Instructions		26

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_XC8.H001	Arbitration mode when using external trigger at the selected input line REQTR		29
BROM_XC8.H001	SYSCON0.RMAP handling in ISR		29
CCU6_XC8.H001	Pin Configuration of CCU6 Functions		30
CCU6_XC8.H002	CCU6 PM event in center-aligned mode		30
EVR_XC8.H002	Enhancement for Noise Immunity		31
FLASH_XC8.H004	Disable TRAP When Calling ROM Flash Routine		32
LIN_XC8.H001	LIN BRK field detection logic		32
MultiCAN_AI.H005	TxD Pulse upon short disable request		33
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off	New	33
MultiCAN_TC.H002	Double Synchronization of receive input		34
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		34
MultiCAN_TC.H004	Double remote request		35
PIN_XC8.H001	Current over GPIO pin must not source V_{DDP} higher than 0.3V		35
PM_XC8.H001	Clock source selection before entering power-down mode		36
PM_XC8.H002	SAK product variant does not support power-down mode		36
SYS_XC8.H001	Usage of the Bit Protection Scheme		36

Table 6 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
SYS_XC8.H002	External Clock switching routine after a WDT reset		37
SYS_XC8.H003	Effective write for Read-Modify-Write instructions of two bytes, one machine cycle	New	37
T2_XC8.H003	Accessing Timer 21 registers		38

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 - 07_H

Workaround

None

CD_XC8.001 Set and Clear of Error Bit in CORDIC Linear Vectoring Mode

In linear vectoring mode, the Error bit of register CD_STATC is set immediately on detecting overflow. When detected between iterations – the Error status is not held internally till the end of the calculation.

As the Error bit is defined such that it is cleared on any read access to the register (e.g. JB BSY), SW checking of the Error bit only at the end of calculation may miss to detect an overflow error condition.

Workaround

Especially in linear vectoring mode, if the error condition setting of Error bit must be detected, any read access should be done on the whole CD_STATC register (e.g. MOV) and the Error bit checked in all read instances.

CD_XC8.002 Data Fetch to CD_STATC Register may capture an incorrect error status

The error bit CD_STATC.ERROR is defined such that the bit is cleared on any read access to the register. Therefore, it is necessary to perform a data fetch on the register and check for the error bit in order not to lose the error status.

However, if CORDIC is clocked at two times PCLK and the execution time of the read instruction is more than one machine cycle, multiple read accesses will be performed on the CD_STATC register and the error bit will be cleared by the time the CPU performs the final read. As a result, the CPU does not capture the correct error status.

There is no problem if the CORDIC is clocked at the same frequency as PCLK.

Workaround

The following workarounds can be used to avoid incorrect data fetching from the CD_STATC register:

- The PUSH dir and POP dir instructions can be used to read the CD_STATC register in all conditions;
- The following one-machine cycle MOV instruction can be used to read the CD_STATC register when the CORDIC is clocked at two times of PCLK:
  - MOV A, dir

MultiCAN_AI.040 Remote frame transmit acceptance filtering error

Correct behaviour:

Assume the MultiCAN message object receives a remote frame that leads to a valid transmit request in the same message object (request of remote answer), then the MultiCAN module prepares for an immediate answer of the remote request. The answer message is arbitrated against the winner of transmit acceptance filtering (without the remote answer) with a respect to the priority class (`MOARn.PRI`).

Wrong behaviour:

Assume the MultiCAN message object receives a remote frame that leads to a valid transmit request in the same message object (request of remote answer), then the MultiCAN module prepares for an immediate answer of the remote request. The answer message is arbitrated against the winner of transmit acceptance filtering (without the remote answer) with a respect to the CAN arbitration rules and not taking the PRI values into account.

If the remote answer is not sent out immediately, then it is subject to further transmit acceptance filtering runs, which are performed correctly.

Workaround

Set MOFCRn.FRREN=1_B and MOFGPRn.CUR to this message object to disable the immediate remote answering.

MultiCAN_AI.041 Dealloc Last Obj

When the last message object is deallocated from a list, then a false list object error can be indicated.

Workaround

- Ignore the list object error indication that occurs after the deallocation of the last message object.

or

- Avoid deallocating the last message object of a list.

MultiCAN_AI.042 Clear MSGVAL during transmit acceptance filtering

Assume all CAN nodes are idle and no writes to MOCTRn of any other message object are performed. When bit MOCTRn.MSGVAL of a message object with valid transmit request is cleared by software, then MultiCAN may not start transmitting even if there are other message objects with valid request pending in the same list.

Workaround

- Do not clear MOCTRn.MSGVAL of any message object during CAN operation. Use bits MOCTRn.RXEN, MOCTRn.TXEN0 instead to disable/reenable reception and transmission of message objects.

or

- Take a dummy message object, that is not allocated to any CAN node. Whenever a transmit request is cleared, set MOCTRm.TXRQ of the dummy message object thereafter. This retriggers the transmit acceptance filtering process.

MultiCAN_AI.043 Dealloc Previous Obj

Assume two message objects m and n (message object n = MOCTRm.PNEXT, i.e. n is the successor of object m in the list) are allocated. If message m is reallocated to another list or to another position while the transmit or receive acceptance filtering run is performed on the list, then message object n may not be taken into account during this acceptance filtering run. For the frame reception message object n may not receive the message because n is not taken into account for receive acceptance filtering. The message is then received by the second priority message object (in case of any other acceptance filtering match) or is lost when there is no other message object configured for this identifier.For the frame transmission message object n may not be selected for transmission, whereas the second highest priority message object is selected instead (if any). If there is no other message object in the list with valid transmit request, then no transmission is scheduled in this filtering round. If in addition the CAN bus is idle, then no further transmit acceptance filtering is issued unless another CAN node starts a transfer or one of the bits MSGVAL, TXRQ, TXEN0, TXEN1 is set in the message object control register of any message object.

Workaround

- After reallocating message object m, write the value one to one of the bits MSGVAL, TXRQ, TXEN0, TXEN1 of the message object control register of any message object in order to retrigger transmit acceptance filtering.

- For frame reception, make sure that there is another message object in the list that can receive the message targeted to n in order to avoid data loss (e.g. a message object with an acceptance mask=0_D and PRI=3_D as last object of the list).

MultiCAN_AI.044 RxFIFO Base SDT

If a receive FIFO base object is located in that part of the list, that is used for the FIFO storage container (defined by the top and bottom pointer of this base object) and bit SDT is set in the base object (CUR pointer points to the base object), then MSGVAL of the base object is cleared after storage of a received frame in the base object without taking the setting of MOFGPRn.SEL into account.

Workaround

Take the FIFO base object out of the list segment of the FIFO slave objects, when using Single Data Transfer.

MultiCAN_AI.045 OVIE Unexpected Interrupt

When a gateway source object or a receive FIFO base object with MOFCRn.OVIE set transmits a CAN frame, then after the transmission an unexpected interrupt is generated on the interrupt line as given by MOIPRm.RXINP of the message object referenced by m=MOFGPRn.CUR.

Workaround

Do not transmit any CAN message by receive FIFO base objects or gateway source objects with bit MOFCRn.OVIE set.

MultiCAN_AI.046 Transmit FIFO base Object position

If a message object n is configured as transmit FIFO base object and is located in the list segment that is used for the FIFO storage container (defined by

MOFGPRn.BOT and MOFGPRn.TOP) but not at the list position given by MOFGPRn.BOT, then the MultiCAN uses incorrect pointer values for this transmit FIFO.

Workaround

The transmit FIFO works properly when the transmit FIFO base object is either at the bottom position within the list segment of the FIFO (MOFGPRn.BOT=n) or outside of the list segment as described above.

MultiCAN_TC.025 RXUPD behavior

When a CAN frame is stored in a message object, either directly from the CAN node or indirectly via receive FIFO or from a gateway source object, then bit MOCTR.RXUPD is set in the message object before the storage process and is automatically cleared after the storage process.

Problem description

When a standard message object (MOFCR.MMC) receives a CAN frame from a CAN node, then it processes its own RXUPD as described above (correct).

In addition to that, it also sets and clears bit RXUPD in the message object referenced by pointer MOFGPR.CUR (wrong behavior).

Workaround

The “foreign” RXUPD pulse can be avoided by initializing MOFGPR.CUR with the message number of the object itself instead of another object (which would be message object 0 by default, because MOFGPR.CUR points to message object 0 after reset initialization of MultiCAN).

MultiCAN_TC.026 MultiCAN Timestamp Function

The timestamp functionality does not work correctly.

Workaround

Do not use timestamp.

MultiCAN_TC.027 MultiCAN Tx Filter Data Remote

Message objects of priority class 2 (MOAR.PRI = 2) are transmitted in the order as given by the CAN arbitration rules. This implies that for 2 message objects which have the same CAN identifier, but different DIR bit, the one with DIR = 1 (send data frame) shall be transmitted before the message object with DIR = 0, which sends a remote frame. The transmit filtering logic of the MultiCAN leads to a reverse order, i.e the remote frame is transmitted first. Message objects with different identifiers are handled correctly.

Workaround

None.

MultiCAN_TC.028 SDT behavior

Correct behavior

Standard message objects:

MultiCAN clears bit MOCTR.MSGVAL after the successful reception/transmission of a CAN frame if bit MOFCR.SDT is set.

Transmit Fifo slave object:

MultiCAN clears bit MOCTR.MSGVAL after the successful reception/transmission of a CAN frame if bit MOFCR.SDT is set. After a transmission, MultiCAN also looks at the respective transmit FIFO base object and clears bit MSGVAL in the base object if bit SDT is set in the base object and pointer MOFGPR.CUR points to MOFGPR.SEL (after the pointer update).

Gateway Destination/Fifo slave object:

MultiCAN clears bit MOCTR.MSGVAL after the storage of a CAN frame into the object (gateway/FIFO action) or after the successful transmission of a CAN frame if bit MOFCR.SDT is set. After a reception, MultiCAN also looks at the respective FIFO base/Gateway source object and clears bit MSGVAL in the base object if bit SDT is set in the base object and pointer MOFGPR.CUR points to MOFGPR.SEL (after the pointer update).

Problem description

Standard message objects:

After the successful transmission/reception of a CAN frame, MultiCAN also looks at message object given by MOFGPR.CUR. If bit SDT is set in the referenced message object, then bit MSGVAL is cleared in the message object CUR is pointing to.

Transmit FIFO slave object:

Same wrong behaviour as for standard message object. As for transmit FIFO slave objects CUR always points to the base object, the whole transmit FIFO is set invalid after the transmission of the first element instead after the base object CUR pointer has reached the predefined SEL limit value.

Gateway Destination/Fifo slave object:

Correct operation of the SDT feature.

Workaround

Standard message object:

Set pointer MOFGPR.CUR to the message number of the object itself.

Transmit FIFO:

Do not set bit MOFCR.SDT in the transmit FIFO base object. Then SDT works correctly with the slaves, but the FIFO deactivation feature by CUR reaching a predefined limit SEL is lost.

MultiCAN_TC.029 Tx FIFO overflow interrupt not generated

Specified behaviour

After the successful transmission of a Tx FIFO element, a Tx overflow interrupt is generated if the FIFO base object fulfils these conditions:

- Bit MOFCR.OVIE=1, AND
- MOFGPR.CUR becomes equal to MOFGPR.SEL

Real behaviour

A Tx FIFO overflow interrupt will not be generated after the transmission of the Tx FIFO base object.

Workaround

If Tx FIFO overflow interrupt needed, take the FIFO base object out of the circular list of the Tx message objects. That is to say, just use the FIFO base object for FIFO control, but not to store a Tx message.

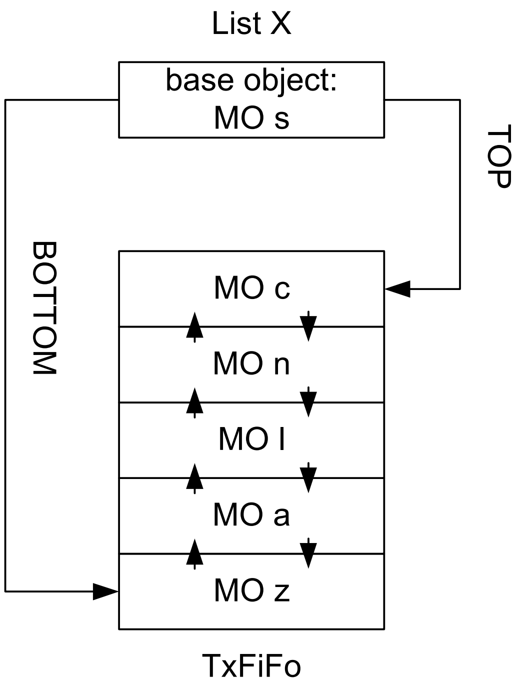

Figure 1 FIFO structure

MultiCAN_TC.030 Wrong transmit order when CAN error at start of CRC transmission

The priority order defined by acceptance filtering, specified in the message objects, define the sequential order in which these messages are sent on the CAN bus. If an error occurs on the CAN bus, the transmissions are delayed due to the destruction of the message on the bus, but the transmission order is kept. However, if a CAN error occurs when starting to transmit the CRC field, the arbitration order for the corresponding CAN node is disturbed, because the faulty message is not retransmitted directly, but after the next transmission of the CAN node.

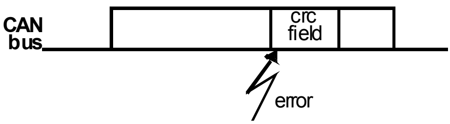

Figure 2

Workaround

None.

MultiCAN_TC.031 List Object Error wrongly triggered

If the first list object in a list belonging to an active CAN node is deallocated from that list position during transmit/receive acceptance filtering (happening during message transfer on the bus), then a "list object" error may occur (NSRx.LOE=1_B), which will cause that effectively no acceptance filtering is performed for this message by the affected CAN node.

As a result:

- for the affected CAN node, the CAN message during which the error occurs will not be stored in a message object. This means that although the message is acknowledged on the CAN bus, its content will be ignored.

- the message handling of an ongoing transmission is not disturbed, but the transmission of the subsequent message will be delayed, because transmit acceptance filtering has to be started again.
- message objects with pending transmit request might not be transmitted at all due to failed transmit acceptance filtering.

Workaround

EITHER:

- Avoid deallocation of the first element on active CAN nodes. Dynamic reallocations on message objects behind the first element are allowed, OR
- Avoid list operations on a running node. Only perform list operations, if CAN node is not in use (e.g. when `NCRx.INIT`=1_B)

MultiCAN_TC.032 MSGVAL wrongly cleared in SDT mode

When Single Data Transfer Mode is enabled (`MOFCRn.SDT`=1_B), the bit `MOCTRn.MSGVAL` is cleared after the reception of a CAN frame, no matter if it is a data frame or a remote frame.

In case of a remote frame reception and with `MOFCR.FRREN` = 0_B, the answer to the remote frame (data frame) is transmitted despite clearing of `MOCTRn.MSGVAL` (incorrect behaviour). If, however, the answer (data frame) does not win transmit acceptance filtering or fails on the CAN bus, then no further transmission attempt is made due to cleared `MSGVAL` (correct behaviour).

Workaround

- To avoid a single trial of a remote answer in this case, set `MOFCR.FRREN` = 1_B and `MOFGPR.CUR` = this object.

MultiCAN_TC.035 Different bit timing modes

Bit timing modes (`NFCRx.CFMOD`=10_B) do not conform to the specification.

When the modes 001_B-100_B are set in register `NFCRx.CFSEL`, the actual configured mode and behaviour is different than expected.

Table 7

Bit timing mode (`NFCR.CFSEL`) according to spec	Value to be written to `NFCR.CFSEL` instead	Measurement
001_B	Mode is missing (not implemented) in MultiCAN	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010_B	011_B	Whenever a dominant edge is received as a result of a transmitted dominant edge the time (clock cycles) between both edges is stored in CFC.
011_B	100_B	Whenever a recessive edge is received as a result of a transmitted recessive edge the time (clock cycles) between both edges is stored in CFC.
100_B	001_B	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.

Workaround

None.

MultiCAN_TC.037 Clear MSGVAL

Correct behaviour:

When MSGVAL is cleared for a message object in any list, then this should not affect the other message objects in any way.

Message reception (wrong behaviour):

Assume that a received CAN message is about to be stored in a message object A, which can be a standard message object, FIFO base, FIFO slave, gateway source or gateway destination object.

If during of the storage action the user clears MOCTR.MSGVAL of message object B in any list, then the MultiCAN module may wrongly interpret this temporarily also as a clearing of MSGVAL of message object A. The result of this is that the message is not stored in message object A and is lost. Also no status update is performed on message object A (setting of NEWDAT, MSGLST, RXPND) and no message object receive interrupt is generated. Clearing of MOCTR.MSGVAL of message object B is performed correctly.

Message transmission (wrong behaviour):

Assume that MultiCAN is about to copy the message content of a message object A into the internal transmit buffer of the CAN node for transmission.

If during of the copy action the user clears MOCTR.MSGVAL of message object B in any list, then the MultiCAN module may wrongly interpret this also as a clearing of MSGVAL of message object A. The result of this is that the copy action for message A is not performed, bit NEWDAT is not cleared and no transmission takes place (clearing MOCTR.MSGVAL of message object B is performed correctly). In case of idle CAN bus and the user does not actively set the transmit request of any message object, this may lead to not transmitting any further message object, even if they have a valid transmit request set.

Single data transfer feature:

When the MultiCAN module clears MSGVAL as a result of a single data transfer (MOFCR.SDT = 1 in the message object), then the problem does not occur. The problem only occurs if MSGVAL of a message object is cleared via CPU.

Workaround

Do not clear MOCTR.MSGVAL of any message object during CAN operation. Use bits MOCTR.RXEN, MOCTR.TXEN0 instead to disable/reenable reception and transmission of message objects.

MultiCAN_TC.038 Cancel TXRQ

When the transmit request of a message object that has won transmit acceptance filtering is cancelled (by clearing MSGVAL, TXRQ, TXEN0 or TXEN1), the CAN bus is idle and no writes to MOCTR of any message object are performed, then MultiCAN does not start the transmission even if there are message objects with valid transmit request pending.

Workaround

To avoid that the CAN node ignores the transmission:

- take a dummy message object, that is not allocated to any CAN node. Whenever a transmit request is cleared, set TXRQ of the dummy message object thereafter. This retriggers the transmit acceptance filtering process.

or:

- whenever a transmit request is cleared, set one of the bits TXRQ, TXEN0 or TXEN1, which is already set, again in the message object for which the transmit request is cleared or in any other message object. This retriggers the transmit acceptance filtering process.

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 8

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 8

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 8

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 8

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.

<u>Sample Code:</u>

```
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.

<u>Sample Code:</u>

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

SYS_XC8.003 Brownout Reset

Brownout reset may not be triggered when the core supply voltage (VDDC) drops below operating limit. It is recommended to use an external voltage detector and perform a power-on reset when the core supply voltage drops below 2.2V (minimum).

Workaround

None.

T2CCU_XC8.003 T2CCU Capture Functions

Capture mode 1 is the only T2CCU capture mode in XC878 AC step. Capture mode 0 of T2CCU is not functioning. In mode 1, a capture will occur upon writing to the low byte of the corresponding channel capture register, CCxL.

Workaround

None.

UART_XC8.002 Bits FDEN and FDM in UART1_FDCON SFR cannot be Written by Read-Modify-Write Instructions

The bits FDEN and FDM in UART1_FDCON SFR are not updated when written with the read-modify-write instructions listed in the table below:

Table 9

Affected Read-Modify-Write Instructions	Hex Code
INC dir	05
DEC dir	15
ANL dir,A	52
ANL dir,#data	53
ORL dir,A	42
ORL dir,#data	43

Table 9

Affected Read-Modify-Write Instructions	Hex Code
XRL dir,A	62
XRL dir,#data	63
XCH A,dir	C5
DJNZ dir,rel	D5

Workaround

Use MOV instructions, except MOV dir, dir (Hex Code: 85), when writing to the bits `FDEN` and `FDM` in `UART1_FDCON` SFR.

3 Deviations from Electrical- and Timing Specification

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.

CCU6_XC8.H001 Pin Configuration of CCU6 Functions

Table 10 shows the updated pin configuration of CCU6 pins in XC878.

Table 10 Updated CCU6 Pin Configuration

CCU6 Functions		Pin Symbol
CCPOS0_0	CCU6 Hall Input 0	P5.4
CCPOS1_0	CCU6 Hall Input 1	P5.3
CC60_3	Input of Capture/Compare Channel 0	P5.3
CC61_3	Input of Capture/Compare Channel 1	P5.4
CC62_3	Input of Capture/Compare Channel 2	P5.5

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:

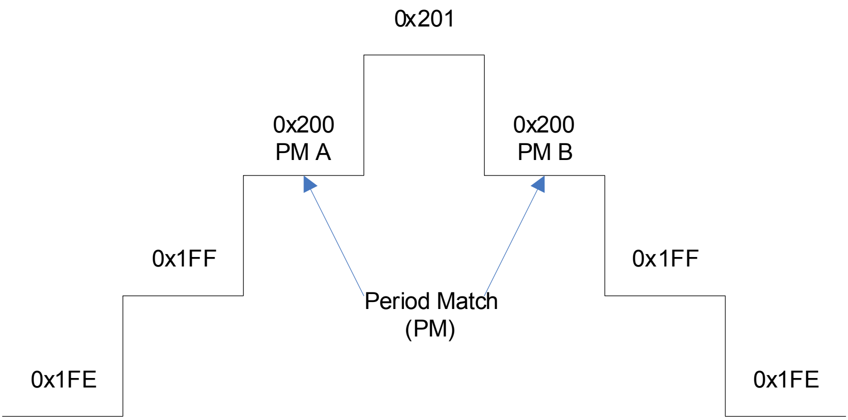

Figure 3 Counting sequence of T12 in center-aligned mode

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.

EVR_XC8.H002 Enhancement for Noise Immunity

During power up, the EVR functionality may be affected due to injected noise from any functional pin. In order to enhance the noise immunity, the external reset pin RESET must be asserted until VDDC reaches 0.9 * VDDC. The delay of external reset can be realized by an external capacitor at RESET pin. This capacitor value must be selected so that VRESET reaches 0.4 V, but not before VDDC reaches 0.9 * VDDC.

A typical application example is shown in **Figure 4**. A 220 nF capacitor is connected to VDDP pin, VDDC pin and RESET pin. In addition, it is also essential to put it as close as possible to the chip.

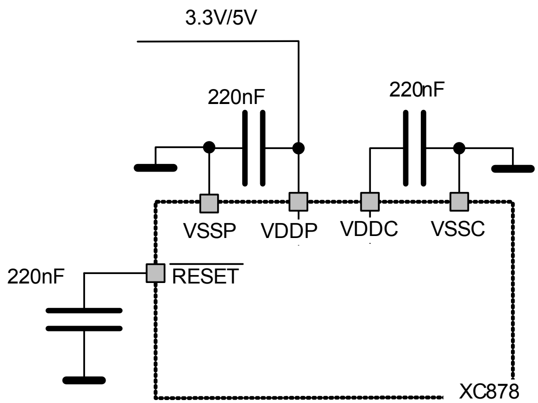

Figure 4 Noise Immunity Enhancement Circuitry

FLASH_XC8.H004 Disable TRAP When Calling ROM Flash Routine

User code has to disable TRAP by clearing EO.TRAP_EN bit before any CALL to ROM Flash routines, and to restore the bit after that.

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:

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

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

For example, if LIN baudrate is 19.2kbps, BGSEL = 0 and CPU frequency is 24MHz, the maximum number of bits in BRK field would be:

19.2k x 4095 / (24M / 8) = ~26.2 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.

MultiCAN_AI.H005 TxD Pulse upon short disable request

If a CAN disable request is set and then canceled in a very short time (one bit time or less) then a dominant transmit pulse may be generated by MultiCAN module, even if the CAN bus is in the idle state.

Example for setup of the CAN disable request:

`PMCON1.CAN_DIS` = 1 and then `PMCON1.CAN_DIS` = 0

Workaround

Set all INIT bits to 1 before requesting module disable.

MultiCAN_AI.H007 Alert Interrupt Behavior in case of Bus-Off

The MultiCAN module shows the following behavior in case of a bus-off status:

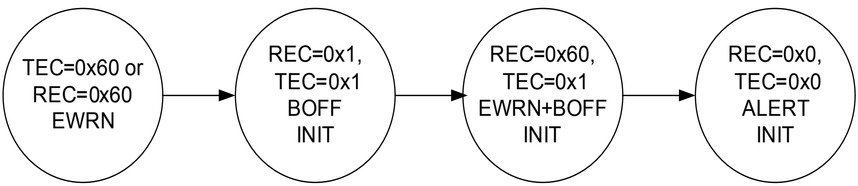

Figure 5 Alert Interrupt Behavior in case of Bus-Off

When the threshold for error warning (EWRN) is reached (default value of Error Warning Level EWRN = 0x60), then the EWRN interrupt is issued. The bus-off (BOFF) status is reached if TEC > 255 according to CAN specification, changing the MultiCAN module with REC and TEC to the same value 0x1, setting the INIT bit to 1_B, and issuing the BOFF interrupt. The bus-off recovery phase starts automatically. Every time an idle time is seen, REC is incremented. If REC = 0x60, a combined status EWRN+BOFF is reached. The corresponding interrupt can also be seen as a pre-warning interrupt, that the bus-off recovery phase will be finished soon. When the bus-off recovery phase has finished (128 times idle time have been seen on the bus), EWRN and BOFF are cleared, the ALERT interrupt bit is set and the INIT bit is still set.

MultiCAN_TC.H002 Double Synchronization of receive input

The MultiCAN module has a double synchronization stage on the CAN receive inputs. This double synchronization delays the receive data by 2 module clock cycles. If the MultiCAN is operating at a low module clock frequency and high CAN baudrate, this delay may become significant and has to be taken into account when calculating the overall physical delay on the CAN bus (transceiver delay etc.).

MultiCAN_TC.H003 Message may be discarded before transmission in STT mode

If `MOFCRn.STT`=1 (Single Transmit Trial enabled), bit TXRQ is cleared (TXRQ=0) as soon as the message object has been selected for transmission and, in case of error, no retransmission takes places.

Therefore, if the error occurs between the selection for transmission and the real start of frame transmission, the message is actually never sent.

Workaround

In case the transmission shall be guaranteed, it is not suitable to use the STT mode. In this case, `MOFCRn.STT` shall be 0.

MultiCAN_TC.H004 Double remote request

Assume the following scenario: A first remote frame (dedicated to a message object) has been received. It performs a transmit setup (`TXRQ` is set) with clearing `NEWDAT`. MultiCAN starts to send the receiver message object (data frame), but loses arbitration against a second remote request received by the same message object as the first one (`NEWDAT` will be set).

When the appropriate message object (data frame) triggered by the first remote frame wins the arbitration, it will be sent out and `NEWDAT` is not reset. This leads to an additional data frame, that will be sent by this message object (clearing `NEWDAT`).

There will, however, not be more data frames than there are corresponding remote requests.

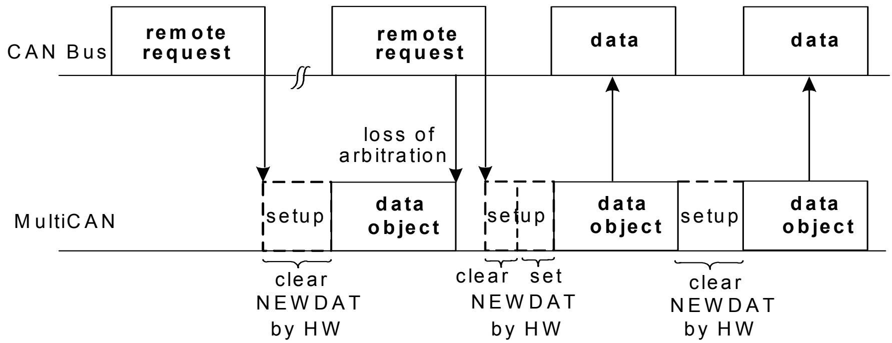

Figure 6 Loss of Arbitration

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} ? 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.

PM_XC8.H001 Clock source selection before entering power-down mode

There are two oscillator sources available in the clock system: on-chip oscillator and external oscillator via XTAL pad. When external oscillator is selected to be the clock source (OSC_CON.OSCSS=1), the XTAL pad will not be shut down automatically during power-down mode. If optimal power-down current is required, on-chip oscillator should be chosen as the clock source before entering power-down mode.

Note: SAK product variant does not support power-down mode.

PM_XC8.H002 SAK product variant does not support power-down mode

Power-down mode is not available in the SAK product variant. It is only supported in SAF and SAX product variants. The profile of these variants is described in **Table 11**.

Table 11 Temperature Profile

Variant Type	Temperature Profile (°C)
SAF	-40 to 85
SAX	-40 to 105
SAK	-40 to 125

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.H002 External Clock switching routine after a WDT reset

When a WDT reset happens, the clock system will not be reset. If user needs to run the clock switching routine after a WDT reset, the bit field `OSC_CON.OSCSS` should be clear to 0 before activating the routine.

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 12

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

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

Table 12

Mnemonic	Hex Code	Bytes	No. of CCLK cycles (without wait states)
SETB bit	D2	2	2
CPL bit	B2	2	2

T2_XC8.H003 Accessing Timer 21 registers

To access Timer 21 registers, T2 page needs to be setup by clearing bit field `T2_PAGE.PAGE` to 000_B.