Errata Sheet

V1.0, 2005-06-09

Device **XC164CS-8F20F, -8F40F**

Marking/Step **Step ES+BA, BA**

Package **P-TQFP-100**

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 applies to all temperature (SAB-/SAF-/SAK-.....) and frequency versions (.20./.40.), unless explicitly noted otherwise.

Versions beginning from V1.0 apply to the current Infineon documentation rules.

Current Documentation

- **XC164CS Data Sheet, V2.1, 2003-06**
- **XC164-16 User's Manual V2.1, Volume 1: System Units, 2004-03**
- **XC164-16 User's Manual V2.1, Volume 2: Peripheral Units, 2004-03**
- **C166S V2 User's Manual (Core, Instruction Set) V1.7, 2001-01**

Note: Devices additionally marked with EES- or ES- or E followed by a 3-digit date code 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.

Note: For simplicity all versions are referred to by the term XC164CS-8FF throughout this document

Contents

1 History List/Change Summary

(from step AD, previous errata sheet: V0.1, step: (E)ES-BA)

Table 1 Functional Deviations

Functional Problem	Short Description	Fixed in step	Change
EBC_X.003	TwinCAN access with EBC enabled		
CPU_X.002	Branch to wrong target after mispredicted JMPI		
RTC_X.001	Real Time Clock operation during sleep or power down mode		
SCU_X.008.1	Modification of register PLLCON while CPSYS = 1	(E)ES-BA	
SCU_X.009.1	Software or Watchdog Timer Reset while Oscillator is not locked	(E)ES-BA	
SCU_X.010	Reset while PLL is not locked - step (E)ES-BA only	ES+BA BA	fixed
SCU_X.011	Register Security Mechanism after Write Access in Secured Mode		new
TwinCAN2.005	Double Send	(E)ES-BA	
TwinCAN2.006	CPUUPD fast set/reset	(E)ES-BA	
TwinCAN_X.007	Transmit after error (former name: TwinCAN2.007)		re-named
TwinCAN_X.008	Double remote request (former name: TwinCAN2.008)		re-named
TwinCAN_X.009	CPUUPD remote (steps ≥ BA only) (former name: TwinCAN2.009)		re-named
FLASH_X.004	PACER trap after wake-up from Sleep/Idle mode (steps ≥ BA only)		
ASC_X.001	ASC Autobaud Detection in 8-bit Modes with Parity		new
ADC_X.003	Coincidence of Result Read and End of next Conversion or Start of Injected Conversion in Wait for Read Mode	(E)ES-BA	

Table 1 Functional Deviations (cont'd)

Functional Problem	Short Description	Fixed in step	Change
ADC_X.004	ADC Overrun Error Generation when result is read during last cycle of conversion	(E)ES-BA	
CC6_X.002	T12 Shadow Transfer for Phase Delay Function	(E)ES-BA	
SLEEP_X.001	Wake up trigger during last clock cycle before entry into sleep mode	(E)ES-BA	

Table 2 OCDS and OCE Modules

Functional Problem	Short Description	Fixed in step	Change
OCDS_X.001	$\overline{\text{BRKOUT}}$ pulsing after Instruction Pointer Debug Event	(E)ES-BA	
OCDS_X.002	OCDS indicates incorrect status after break_now requests if PSW.ILVL $\geq$ CMCTR.LEVEL		
OCE_X.001	Wrong MAC Flags are declared valid at Core - OCE interface		

Table 3 AC/DC Deviations

AC/DC Deviations	Short Description	Fixed in step	Change
TAP_X.85	Maximum ambient temperature during flash programming 85°C	(E)ES-BA	
FCPUR_X.162832	Frequency Limits for Flash Read Accesses	(E)ES-BA	
M40_WLE_X.001	Minimum Ambient Temperature for Flash Wordline Erase Command		new

Table 4 Application Hints

Hint	Short Description	Change
CPU_X.H1	Configuration of Registers CPUCON1 and CPUCON2	
CPU_X.H2	Special Characteristics of I/O Areas	
FLASH_X.H1.1	Access to Flash Module after Program/Erase	
FLASH_X.H2.2	Access to Flash Module after Shut-Down	modified
FLASH_X.H3.2	Read Access to internal Flash Module with modified Margin Level	modified
FLASH_X.H4	Minimum active time after wake-up from sleep or idle mode	
SLEEP_X.H3.2	Clock system after wake-up from Sleep Mode (steps ≥ BA)	
IDLE_X.H1	Entering Idle Mode after Flash Program/Erase	
ADC_X.H1	Polling of Bit ADBSY	
BREAK_X.H1	Break on MUL/DIV followed by zero-cycle jump	
POWER_X.H1.1	Initialization of SYSCON3 for Power Saving Modes	
POWER_X.H2.2	Power Consumption during Clock System Configuration	
RSTOUT_X.H1	$\overline{\text{RSTOUT}}$ driven by weak driver during HW Reset	
SCU_X.H1	Shutdown handshake by software reset (SRST) instruction	
SCU_X.H3	Effect of PLLODIV on Duty Cycle of CLKOUT	
SCU_X.H4	Changing PLLCON in Emergency Mode	
SCU_X.H5	Sleep/Idle/Power Down Mode not entered while PLLODIV = $0F_H$	
SCU_X.H6	Interrupt request during entry into sleep mode	new
RTC_X.H1.2	Resetting and Disabling of the Real Time Clock	
FOCON_X.H1	Read Access to register FOCON	
TwinCAN_X.H2	Reading Bitfield INTID	new

Table 5 Documentation Update

Name	Short Description	Change
INT_X.D1	Interrupt Vector Location of CAPCOM Register 28	new

Table 5 Documentation Update (cont'd)

Name	Short Description	Change
PORTS_X.D2_100	Internal Pull-up Device active on P3.12 during Reset	new
SCU_X.D1.1	Oscillator gain reduction (steps ≥ BA)	modified
SCU_X.D2.2	Functionality of register OPSEN (steps ≥ BA)	modified
SCU_X.D3	Register PLLCON after software reset	
SCU_X.D5	VCO band after hardware/watchdog reset in single chip mode	
SCU_X.D6	Register Security Mechanism - Unprotected Mode active until execution of EINIT instruction	new
FLASH_X.D1	Interaction between Program Flash and Security Sector Programming	new

2 Functional Problems

EBC_X.003 TwinCAN Access with EBC enabled

If the External Bus Controller (EBC) is enabled, a read or write access to the TwinCAN module fails when an external bus access with TCONCSx.PHA ≠ 00b precedes the TwinCAN access.

Workaround:

Since it is hard to predict the order of external bus and TwinCAN accesses (in particular when PEC transfers are involved), it is recommended to set bitfield PHA to '00' in all TCONCSx registers which are used for external bus accesses.

CPU_X.002 Branch to wrong target after mispredicted JMPI

After a JMPI is initially mispredicted according to the static branch prediction scheme of the C166S V2, code execution may continue at a wrong target address in the following situations:

Situation I:

- a memory write operation is processed by the DMU
- followed by a MUL(U)
- followed by the mispredicted JMPI

```
Example_1:
MOV mem, [Rwn]
MUL R13, R14
JMPI cc_NV, [R6]
```

Situation II:

- MUL(U) or DIV(L/U/LU)
- followed by not-mispredicted zero-cycle jump
  (e.g. JMPA, JMPR, JMPS; bit ZCJ (CPUCON1.0) = 1)
- followed by the mispredicted JMPI

```
Example_2a:
MULU R13, R14
JMPA- cc_V, _some_target  ; predicated not taken => correct
JMPI cc_NV, [R6]          ; taken, but predicted not taken
```

It could be possible that the JMPI is at the jump target of the JMPA, if it is taken:

```
Example_2b:
MULU R13, R14
JMPA+ cc_NZ, _jmpi_addr   ; predicted taken => correct
..... other code ....
_jmpi_addr: JMPI cc_NV, [R6] ; taken but predicted not taken
```

Effect on tools:

In the **Altium/Tasking** compiler (V7.0 and above) the problem is not present. The result of a MUL/DIV instruction is available through the MDL/MDH SFRs. These SFRs are not allocatable by the register allocator. Therefore, the compiler always needs a MOV instruction to transfer MDL/H to a GPR. This avoids the problem.

In the RT- and FP-libraries (v7.0 and above) the problem was not found. Versions lower than v7.0 do not explicitly support the C166S V2 core.

In case optimizations are implemented in future versions which could cause this problem to occur, also a workaround will be included.

All **Keil** C166 tool Versions (compiler and libraries) since V3.xx do not generate a MUL(U) or a DIV(L/U/LU) followed by either of the jump instructions JMPR, JMPS, JMPA, JMPI. Basically the support of the C166S V2 core requires anyway V4.21 or higher.

Workarounds:

Examples for program parts written in assembly language:

- generally disable overrun of pipeline bubbles by clearing bit CPUCON2.OVRUN (CPUCON2.4 = 0). This will result only in a negligible performance decrease, and will prohibit corruption of the target IP of the JMPI.

or:

- provide a NOP (or any other suitable instruction) between the MUL/DIV instructions and the succeeding jump in the above cases. To simplify, place a NOP between any MUL/DIV and a JMPR, JMPS, JMPA, JMPI that might follow it. Other branches (CALLs, jump-on-bit instructions) do not need to be taken into account.

RTC_X.001 Real Time Clock operation during sleep or power down mode

If the Real Time Clock shall run during sleep or power down mode, bit REFCLK (RTC_CON.4) must be set to '1' in addition to bit RUN (RTC_CON.0).

Register RTC_CON is not affected by a hardware/software/watchdog reset. After power-up, it is undefined. A reset of the RTC module is achieved by setting bit RTCRST (SYSCON0.15) = 1. This way, register RTC_CON is set to 8003_H (RTC runs, prescaler

by 8 enabled, bit REFCLK = 0). As a general recommendation, bit REFCLK should be set to '1' after any start-up (hardware, software, watchdog reset), and whenever the RTC module is reset by setting bit RTCRST (SYSCON0.15) = 1.

SCU_X.011 Register Security Mechanism after Write Access in Secured Mode

To modify an SFR that is protected by the register security mechanism, a certain security level has to be selected and/or a command sequence has to be executed prior to the write access to one of these registers. Table 6-15 in the User's Manual, volume System Units, lists all registers protected by the security mechanism (see copy of Table 6-15 below).

After selecting Secured Mode (bitfield SL = 01_B in register SCUSLS), a single command (command4) enables one single write access to a protected register. After this write access the protected registers are locked again automatically.

Exception:

After modification of registers CPUCON1, CPUCON2, EBCMOD0, EBCMOD1, TCONCSx, FCONCSx, ADDRSELx (which are not part of the SCU), all registers listed in Table 6-15 are **not locked** until the next write access to an SCU register (i.e. a register which is different from the group CPUCON1 .. ADDRSELx).

Workaround:

In order to lock all registers again, after a write access to the non-SCU registers CPUCON1 .. ADDRSELx a "dummy" write access to an SCU register should be executed. It is therefore proposed to use e.g. the read-only register IDCHIP for this purpose. The registers of the identification control block also belong to the SCU, and a write access to these read-only registers re-enables secured mode:

Example:

```
MOV  R4, #2000H      ; value to be stored in register EBCMOD0
EXTR #1              ; Access sequence in secured mode
MOV  SCUSLC, #8E12H  ; Command4: current password = EDH
OR   EBCMOD0, R4     ; Access to EBCMOD enabled by preceding Command4
MOV  IDCHIP, ZEROS   ; dummy write to a read-only SCU register
                     ;  re-enables secured mode
```

Table 6 Registers Protected by the Security Mechanism

Register Name	Function	Loc.
RSTCON	Reset control	SCU
SYSCON0	General system control	SCU
SYSCON1	Power management	SCU
PLLCON	Clock generation control	SCU
SYSCON3	Peripheral management	SCU
FOCON	Peripheral management (CLKOUT/FOUT)	SCU
IMBCTR	Control of internal instruction memory block	SCU
OPSEN	Emulation control	SCU
EMUCON	Emulation control	SCU
WDTCON	Watchdog timer properties	SCU
EXICON	Ext. interrupt control	SCU
EXISEL0, EXISEL1	Ext. interrupt control	SCU
CPUCON1, CPUCON2	CPU configuration	CPU
EBCMOD0, EBCMOD1	EBC mode selection	EBC
TCONCSx	EBC timing configuration	EBC
FCONCSx	EBC function configuration	EBC
ADDRSELx	EBC address window configuration	EBC

TwinCAN_X.007 Transmit after Error

During a CAN error, transmission may stop (after EOF or an error frame), until a successful reception or a write access to the TwinCAN module.

Detailed Description

In case of a CAN error and when there is no other activity on the CAN module (e.g. frame reception or frame transmission on the other CAN node or write access to any CAN register), the transmission of messages may stop, even if some transmit requests are still set.

The CAN module will start transmitting immediately, after a reception or a write access to the module.

Workarounds:

- Write periodically $FFFF_H$ to one of the MSGCTRx registers, as this value is having no effect on the register.
- In case writing to a CAN register shall be the exception, use the last error code (LEC) interrupt. This shall start writing to one of the MSGCTRx register $FFFF_H$, in case the LEC value is unequal to 0.

TwinCAN_X.008 Double remote request

After the transmission of the remote request, TXRQ is not cleared in the receive object, if NEWDAT is set. As a consequence the remote request is transmitted once again.

Workaround:

Clear NEWDAT after the reception of a data frame.

TwinCAN_X.009 CPUUPD remote

In case of a remote request to a standard message object which is chosen for transmission, a transmit of the data frame takes place, even if CPUUPD is currently set.

Detailed Description

If a transmit message object gets a remote request and there is no other message object with higher transmit priority pending for transmission, then the transmit object sends the data frame to answer the remote request, even if CPUUPD is set.

Workaround:

This workaround is only required in systems where remote requests are used.

To answer remote requests, the MMC bitfield in MSGFGCR has to be configured to a FIFO slave object instead of a standard message object for transmission. To reach this goal, the following settings for the corresponding message object are needed:

- bitfield MMC (MSGFGCRHn.10-8) = 011_B (FIFO functionality enabled (slave object))
- bitfield CANPTR (MSGFGCRHn.4-0) = n (the CAN Pointer shall reference itself, by referring to the message object number of this object)
- bit FD (MSGFGCRLn.13) = 0 (the CANPTR is updated after a correct reception)

FLASH_X.004 PACER Trap after Wake-Up from Sleep/Idle Mode

An unexpected Program Access Error Trap (flag PACER = 1 in register TFR) occurs after a wake-up event from sleep or idle mode under the following conditions:

- bit field PFCFG = 01b in register SYSCON1, i.e. the flash module is switched off during sleep or idle mode
- a wake-up event (interrupt, PEC transfer, $\overline{\text{NMI}}$) occurs in a specific time window (few clock cycles) after execution of the IDLE instruction during the flash deactivation process
- and the corresponding interrupt/trap routine or the instruction following IDLE (in case interrupts are disabled) or the PEC source data are located in the internal flash.

Workaround 1:

Do not switch off the flash module in sleep or idle mode, i.e. leave bit field PFCFG = 00b in register SYSCON1 (default after reset). This increases power consumption to a certain extent while reducing the time overhead for sleep/idle mode entry and exit (in clocking modes where the clock is derived from the VCO).

Workaround 2

In order to avoid the problem (when PFCFG = 01), make sure that the wake-up trigger only occurs after the device has completely entered sleep or idle mode.

If the RTC is used as wake-up source, check e.g. the RTC before entering sleep mode. If the wake-up trigger will occur soon, either skip entry into sleep mode, or extend the time for the next wake-up. If the RTC time interval is reprogrammed, make sure that no interrupt occurs between reprogramming and entry into sleep mode.

Workaround 3

In order to differentiate an unexpected PACER trap due to a wake-up trigger in the critical time window from other events that can lead to a PACER trap, set a semaphore bit before executing the IDLE instruction., e.g.

```
ATOMIC #2
BSET   sema_idle
IDLE
```

In the trap handler for the PACER trap, if the semaphore bit is set and no other indications for the PACER trap are found (e.g. error flags in register FSR are set), clear the semaphore bit and the PACER flag and return from the trap handler with RETI. The stack contains a valid return address in this case (e.g. address of instruction following IDLE in case interrupts were disabled during wake-up).

For **PEC transfers** during sleep or idle mode, which will cause a PACER trap if they read data from flash, automatic return to sleep/idle mode is not accomplished with the concept described so far.

- In order to support return to idle/sleep mode after a PEC transfer (which is performed after the RETI instruction from the PACER trap routine is executed), e.g. a semaphore bit may be used. This bit may be set to '1' before the IDLE instruction is executed. All trap (except PACER) and interrupt service routines invoked after wake up from idle/ sleep should clear this bit to '0'. After having returned from the PACER trap routine to the program in the internal flash, this bit should be tested (allow a sufficient time of e.g. 12 cycles for interrupt arbitration), and if it is still at '1' (i.e. no interrupts/traps have occurred), repeat the IDLE instruction for re-entry into idle/sleep mode.

Workaround 4

Use an auxiliary sequence in internal PRAM that bridges the time until the flash is ready after wake-up from sleep/idle mode, e.g.:

- Disable interrupts, and execute the IDLE instruction to enter sleep mode from the internal PRAM. After wake-up, the instruction following IDLE will be executed (if no hardware trap or NMI# has occurred).
- Wait until the internal flash is ready after wake-up (check register FSR) before reading from the internal flash). If the sequence in internal PRAM that includes the IDLE instruction is not CALLed from internal flash (i.e. it is not terminated with a RETx instruction), at least 8 instructions that do not read from the internal flash should be inserted after the IDLE instruction to avoid speculative prefetches
- Enable the interrupt system again.

The following details should be considered:

- If hardware traps (including NMI#) can occur, add the corresponding interrupt vectors to PRAM and modify register VECSEG to point to the PRAM space.
- In order to support return to idle/sleep mode after a PEC transfer, e.g. a semaphore bit may be used. This bit may be set to '1' before the IDLE instruction is executed. All trap and interrupt service routines invoked after wake up from idle/sleep should clear this bit to '0'. After having returned to the program in the internal flash and having enabled the interrupt system, this bit should be tested (allow a sufficient time of e.g. 12 cycles for interrupt arbitration), and if it is still at '1' (i.e. no interrupts/traps have occurred), repeat the auxiliary routine that prepares for re-entry into idle/sleep mode.

ASC_X.001 ASC Autobaud Detection in 8-bit Modes with Parity

The Autobaud Detection feature of the Asynchronous/Synchronous Serial Interface (ASC) does **not** work correctly for **8-bit** modes **with** even or odd **parity**.

The Autobaud Detection feature works correctly for 7-bit modes with even or odd parity, and for 8-bit modes without parity.

3 OCDS and OCE Modules

The following issues have been found in the OCDS and OCE modules. Please see the debugger or emulator manufacturer's documentation whether or not these issues actually cause a problem or restriction when the respective tool is used.

OCDS_X.002 OCDS indicates incorrect status after break_now requests if PSW.ILVL ≥ CMCTR.LEVEL

When the OCDS processes a break_now request while the CPU priority level (in PSW.ILVL) is not lower than the OCDS break level (in CMCTR.LEVEL), the actual break is delayed until either PSW.ILVL or CMCTR.LEVEL is reprogrammed such that CMCTR.LEVEL > PSW.ILVL. If in the meantime further debug events have occurred, register DBGSR will still indicate the status of the first break_now request. If e.g. a software break is executed, the OCDS will accept this, but register DBGSR will indicate the wrong cause of break.

Workarounds:

1. If the application uses tasks with different levels and debugging is to take place using the OCDS break level feature (e.g. only tasks up to a maximum level are halted, higher-level tasks aren't halted, and the OCDS level is programmed in between), there is no problem if:

- only classic hardware breakpoints (IP address) or software breakpoints are used (i.e. no trigger on address, data, TASKID)
- no external pin assertions are used to trigger breaks
- no direct writes to DBGSR.DEBUG_STATE are used to force breaks

2. If break_now request sources are to be used, the maximum level of the application (PSW.ILVL) should always be lower than the programmed OCDS break level (e.g. PSW.ILVL ≤ 14_D and CMCTR.LEVEL = 15_D). This means that all generated break_now requests by the OCDS will always be accepted, independent of the CPU or interrupt priority.

OCE_X.001 Wrong MAC Flags are declared valid at Core - OCE interface

In case a MAC instruction (Co...) is directly followed by a MOV MSW, #data16 instruction, the upper byte of data16 is output instead of the flags corresponding to the MAC instruction. The bug was found with code:

```
COSHR   #00001h
MOV     MSW, #00100h  ;(+ other variations of data16)
```

Workaround

Add a NOP instruction between the two instructions:

```
COSHR  #00001h
NOP
MOV    MSW, #00100h  ;(+ other variations of data16)
```

4 Deviations from Electrical- and Timing Specification

M40_WLE_X.001 Minimum Ambient Temperature for Flash Wordline Erase Command

When the Erase Wordline command is executed at low temperature, data in other wordlines of the same physical 64 Kbyte sector may unintentionally be modified in addition.

The effect depends primarily on the ambient temperature and the internal CPU frequency, and may vary from device to device.

Workarounds (choices):

- Do not use the Erase Wordline Command below -20°C
- Reduce the internal CPU frequency to 0.5 MHz when using the Erase Wordline Command below -20°C. After the Erase Wordline Command has been started, the CPU frequency may be increased again to the target value.
- Use the Erase Sector Command instead of the Erase Wordline Command.

5 Application Hints

CPU_X.H1 Configuration of Registers CPUCON1 and CPUCON2

The default values of registers CPUCON1 and CPUCON2 have been chosen to provide optimized performance directly after reset. It is recommended

- not to modify the performance related parts of register CPUCON1
- not to modify register CPUCON2, except for test purposes or for enabling specific workarounds under special conditions (see e.g. problem CPU_X.002 or application hint BREAK_X.H1).

CPUCON2: reset/recommended value = $8FBB_H$; enables several performance features

CPUCON1: reset/recommended value = 0..0 0XXX $X111_B$; only the 3 LSBs are performance related

Bit Position	Field Name	Value	Description
CPUCON1.[15:7]	0	0	reserved
CPUCON1.[6:5]	VECSC	00	scaling factor for vector table, value depends on application, '00' is compatible to C166 systems
CPUCON1.4	WDTCTL	0	configuration for scope and function of DISWDT/ ENWDT instructions, value depends on application, '0' is compatible to C166 systems
CPUCON1.3	SGTDIS	0	segmentation enable/disable control, value depends on application
CPUCON1.2	INTSCXT	1	enable interruptibility of switch context
CPUCON1.1	BP	1	enable branch prediction unit
CPUCON1.0	ZCJ	1	enable zero cycle jump function

CPU_X.H2 Special Characteristics of I/O Areas

As an element of performance optimization, the pipeline of the C166S V2 core may perform speculative read accesses under specific conditions. In case the prediction for the speculative read was wrong, the read to the actually required location is restarted. While this method is uncritical e.g. for accesses to non-volatile memories or SRAMs, it may cause problems on devices which do not tolerate speculative reads (e.g. FIFOs which are advanced on every read access).

No speculative reads are performed in memory areas which are marked as I/O area. This memory area includes

- the SFR and ESFR space (e.g. with buffers for received data from serial interfaces or A/D results)
- the 4 Kbyte internal I/O area ($00'E000_H$ ..$00'EFFF_H$), including IIC[1)] and SDLM[1)] module
- the 2 Mbyte external I/O area ($20'0000_H$ ..$3F'FFFF_H$), including the TwinCAN[1)] module (default: from $20'0000_H$ .. $20'07FF_H$)

It is therefore recommended to map devices which do not tolerate speculative reads into the 2 Mbyte external I/O area ($20'0000_H$ ..$3F'FFFF_H$).

For further special properties of the I/O areas, see section IO Areas (3.6) in chapter Memory Organization in the User's Manual.

FLASH_X.H1.1 Access to Flash Module after Program/Erase

After the last instruction of a program or erase command, the BUSY bit in register FSR is set to '1' (status = busy) after a delay of one instruction cycle. When polling the BUSY flag, one NOP or other instruction which is not evaluating the BUSY flag must be inserted after the last instruction of a program or erase command.

No additional delay is required when performing the first operand read or instruction fetch access from the flash module after the BUSY bit has returned to '0' (status = not busy).

FLASH_X.H2.2 Access to Flash Module after Shut-Down

When the flash is disabled by software (shut-down) by writing bit PFMDIS = 1 in register SYSCON3,

- and it is (at some later time) enabled again by writing PFMDIS = 0
- and the instruction immediately following the instruction which sets PFMDIS = 0 is fetched or reads operands from internal flash

then the PACER flag in register TFR is set and the BTRAP routine is entered.

Therefore, it is recommended to insert 4 NOPs before the internal flash is accessed again after PFMDIS has been set to 0.

1) this module is implemented in specific derivatives of the XC16x family

FLASH_X.H3.2 Read Access to internal Flash Module with modified Margin Level

When the internal flash module is read (e.g. for test purposes) with bitfield margin = 0001_B (low level margin) in register MAR, an additional wait state must be selected in bitfield WSFLASH in register IMBCTR. This waitstate must be added to the number of flash waitstates that are required to match the flash access time to the CPU operating frequency.

FLASH_X.H4 Minimum active time after wake-up from sleep or idle mode

If the flash module is automatically disabled upon entry into sleep or idle mode (bit field PFCFG = 01_B in register SYSCON1), sleep or idle mode should not be re-entered before a minimum active ("awake") time has elapsed. Otherwise, the current consumption during this sleep/idle phase will be ~ 1 mA above the specified limits of the Data Sheet. Therefore,

- If code is executed from the **internal flash** after wake-up, at least 16 instructions should be executed from the internal flash before re-entering sleep/idle mode. This ensures that the flash module is actually accessed after wake-up, since more instructions are required than can be stored in the prefetch queue.
- If code is executed from **external memory or PRAM**, wait until the flash BUSY bit returns to '0' before re-entering sleep/idle mode.
- If **PEC transfers** with automatic return to sleep/idle mode shall be triggered by the wake-up event, use e.g. the following procedure:

  Use an auxiliary routine in internal flash with the required minimum active time after wake-up from sleep or idle mode, e.g.
  - define a semaphore bit that is set to '1' before the IDLE instruction is executed. All trap and interrupt service routines invoked after wake up from idle/sleep should clear this bit to '0'
  - disable interrupts
  - execute the IDLE instruction
  - if idle or sleep mode is terminated by an interrupt request, the instructions following the IDLE instruction will be executed (the interrupt request flags remain set)
  - if idle or sleep mode was terminated by an $\overline{\text{NMI}}$, the trap handler will be invoked
  - enable interrupts to allow prioritization of requests for interrupt or PEC service
  - the instructions following the IDLE instruction should test the flash BUSY bit in register FSR; when the flash is ready (BUSY = 0), and at least 12 instructions have been executed after the interrupt system has been enabled, and if the semaphore bit is still at '1' (i.e. no interrupts/traps have occurred), disable interrupts and return to the IDLE instruction.

SLEEP_X.H3.2 Clock system after wake-up from Sleep Mode

There are different wake-up behaviors, depending on the PLL control setting used in register PLLCON during entry into sleep mode, and depending on whether the RTC is running on the main oscillator. Note that in either case, the VCO is turned off during sleep mode, and does not contribute to any additional power consumption.

- In bypass mode with VCO off (**PLLCTRL = 00_B**), the device will directly continue to run on the frequency derived from the external oscillator input after wake-up from sleep. If the **RTC** is running on the **main** oscillator, the device is immediately clocked, since the oscillator (input XTAL1) is not turned off during sleep mode.

  If the **RTC** was **not** running on the main oscillator, the system will not be clocked until the amplitude on the external oscillator input XTAL1 exceeds the input hysteresis. This requires typ. a few ms, depending on external crystal/oscillator circuit.

  With this mode, there is **no oscillator watchdog function**, and the system will not be clocked until the external oscillator input XTAL1 receives a clock that exceeds the input hysteresis.

- In bypass mode with VCO on (**PLLCTRL = 01_B**), the device will directly continue to run on the frequency derived from the external oscillator input after wake-up from sleep if the **RTC** continues to run on the **main** oscillator in sleep mode.

  In case the PLL was **locked** before entry into sleep mode, **emergency mode** is entered. This results in PLLODIV = $0F_H$ and bit SYSSTAT.EM = 1. This change of configuration will **not** be notified by the PLL Unlock/OWD interrupt (flag PLLIR). This condition will remain until an external HW reset is applied, or a wake-up event from sleep mode with main oscillator off (i.e. RTC not running on main oscillator) occurs.

  If the **RTC** was **not** running on the main oscillator, (i.e. the main oscillator was off during sleep mode), the device will wake-up using the internal PLL base frequency from the VCO (f_{base}/16) and will temporarily stay in emergency mode (i.e. run on the frequency derived from the VCO) until bit OSCLOCK in register SYSSTAT gets set to 1.

  It is not possible to switch to direct drive (VCO bypass) mode within this timeframe. If bypass mode (PLLCTRL = 00_B, i.e. no oscillator watchdog support) is required by an application after wake-up from sleep, it is therefore recommended to switch to bypass mode already before entry into sleep mode (check PLLCON for its target value before executing the IDLE instruction to enter sleep mode). See also SCU_X.H5.

- In PLL mode with input clock from XTAL1 disconnected (**PLLCTRL = 10_B**), the device will **only** wake up from sleep if the RTC was **not** running on the main oscillator (i.e.

when the main oscillator is off during sleep mode). In this case, the device will run using the internal PLL base frequency from the VCO (f_{base}/16) until the amplitude on the external oscillator input XTAL1 exceeds the input hysteresis, and then switch to f_{base}/k with the output divider selected by PLLODIV.

If the **RTC** is running on the **main** oscillator, the device will **not wake-up** from sleep mode with this PLLCTRL setting. It is therefore recommended to switch to bypass mode (PLLCTRL = 00_B) before entry into sleep mode (check PLLCON for its target value before executing the IDLE instruction to enter sleep mode).

- In PLL mode with input clock from XTAL1 connected to the VCO (**PLLCTRL = 11_B**), if the **RTC** was **not** running on the main oscillator, the device will wake-up in emergency mode and run using the internal PLL base frequency from the VCO (f_{base}/16) until the amplitude on the external oscillator input XTAL1 exceeds the input hysteresis. Then the PLL resynchronizes to the target frequency determined by the settings in register PLLCON. When bit OSCLOCK gets set in register SYSSTAT, the output divider PLLODIV will be set to the target value.

  If the **RTC** is running on the main oscillator, the device will wake-up and resynchronize to the target frequency determined by the settings in register PLLCON.

  In case the PLL was **locked** before entry into sleep mode, **emergency mode** is entered. This results in PLLODIV = $0F_H$ and bit SYSSTAT.EM = 1. This change of configuration will not be notified by the PLL Unlock/OWD interrupt (flag PLLIR). This condition will remain until an external HW reset is applied, or a wake-up event from sleep mode with main oscillator off (i.e. RTC not running on main oscillator) occurs.

  As an alternative, switch to bypass mode with VCO on and PLL unlocked before entering sleep mode (e.g. PLLCON = 2000_H). After wake-up, PLLCON may be reconfigured to the desired PLL operating mode.

IDLE_X.H1 Entering Idle Mode after Flash Program/Erase

After a program/erase operation, idle mode should not be entered before the BUSY bit in register FSR has returned to '0' (status = not busy).

ADC_X.H1 Polling of Bit ADBSY

After an A/D conversion is started (standard conversion by setting bit ADST = 1, injected conversion by setting ADCRQ = 1), flag ADBSY is set 5 clock cycles later. When polling

for the end of a conversion, it is therefore recommended to check e.g. the interrupt request flags ADC_CIC_IR (for standard conversions) or ADC_EIC_IR (for injected conversions) instead of ADBSY.

BREAK_X.H1 Break on MUL/DIV followed by zero-cycle jump

When a MUL or DIV instruction is immediately followed by a falsely predicted conditional zero-cycle jump (JMPR or JMPA on any condition other than cc_UC),

and

- either a 'break now' request is set at the time the MUL / DIV instruction is being executed (i.e. a break request on operand address, data, task ID, $\overline{\text{BRKIN}}$ pin etc. is generated by one of the instructions (may be up to four) preceding MUL/DIV)
- or a 'break-before-make' request (break on IP address) is derived from the instruction immediately following the jump (jump target or linear following address, depending whether the jump is taken or not)

then the internal program counter will be corrupted (equal to last value before jump), which will lead to a false update of the IP with the next instruction modifying the IP.

This problem occurs for debugging with OCDS as well as with OCE.

Note: The Tasking and Keil compilers (including libraries) do not generate this type of critical instruction sequence.

Workarounds (choices)

For assembler programmers, one of the following workarounds may be used

1. disable zero-cycle operation for jumps when debugging code (set CPUCON1.ZCJ to '0'), or
2. include a NOP after any MUL/DIV instruction followed by a conditional jump (JMPR, JMPA), or
3. do not set any 'break-before-make'-type breakpoints on the instruction following the jump, or 'break now'-type breakpoints shortly before or on the MUL / DIV instructions

POWER_X.H1.1 Initialization of SYSCON3 for Power Saving Modes

For minimum power consumption during power saving modes, all modules which are not required should be disabled in register SYSCON3, i.e. the corresponding disable bits should be set to '1', including bits which are marked as 'reserved' (this provides compatibility with future devices, since all SYSCON3 bits are disable bits). Reading these bits will return the written value, as for peripherals without shut-down handshake.

For peripherals equipped with peripheral shut-down handshake, reading allows to check their shut-down status.

POWER_X.H2.2 Power Consumption during Clock System Configuration

In the following situations

1. after wake-up from sleep mode until oscillator lock in case the main oscillator was turned off during sleep mode
2. after a clock failure (PLL unlock or oscillator fail) until clock reconfiguration by software

the device is internally clocked by the VCO running on the base frequency of the currently selected VCO band divided by 16. This results in an operating frequency range of 1.25 .. 11.25 MHz, depending on the currently selected VCO band.

Systems designed for lower target frequencies should consider the increased power consumption due to the potential frequency increase during these phases of operation.

Exception in **bypass mode with VCO off**: in case (1), if the RTC is not running on the main oscillator, and case in (2) the device stops until it again receives a clock from the oscillator.

RSTOUT_X.H1 $\overline{\text{RSTOUT}}$ driven by weak driver during HW Reset

A weak driver (see specification in Data Sheet) has been implemented on pin $\overline{\text{RSTOUT}}$ which is driven low while $\overline{\text{RSTIN}}$ is asserted low. After the end of the internal reset sequence, $\overline{\text{RSTOUT}}$ operates in default mode (strong driver/sharp edge mode, i.e. POCON20.PDM3N[15:12] = 0000_B).

The software setting POCON20.PDM3N[15:12] = $xx11_B$ is not supported and should not be selected by software, otherwise pin $\overline{\text{RSTOUT}}$ floats.

SCU_X.H1 Shutdown handshake by software reset (SRST) instruction

In the pre-reset phase of the software reset instruction, the SCU requests a shutdown from the active modules equipped with shutdown handshake (see section Peripheral Shutdown Handshake (6.3.3) in chapter Central System Control Functions in the User's Manual). The pre-reset phase is complete as soon as all modules acknowledge the shutdown state.

As a consequence, e.g. the A/D converter will only acknowledge the request after the current conversion is finished (fixed channel single conversion mode), or after

conversion of channel 0 (auto scan single conversion mode). If the 'Wait for Read Mode' mode is selected (bit ADWR = 1), the ADC does not acknowledge the request if the conversion result from register ADC_DAT has not been read.

Therefore, before the SRST instruction is executed, it is recommended e.g. in the continuous (fixed or auto scan) conversion modes to switch to fixed channel single conversion mode (ADM = 00_B) and perform one last conversion in order to stop the ADC in a defined way. In the auto scan conversion modes, this switch is performed after conversion of channel 0. If a 0-to-1 transition is forced in the start bit ADST by software, a new conversion is immediately started. If the 'Wait for Read Mode' is selected, register ADC_DAT must be read after the last conversion is finished.

The external bus controller e.g. may not acknowledge a shutdown request if bus arbitration is enabled and the $\overline{\text{HOLD}}$ input is asserted low.

SCU_X.H3 Effect of PLLODIV on Duty Cycle of CLKOUT

When using even values (0..14) for the output divider PLLODIV in register PLLCON, the duty cycle for signal CLKOUT may be below its nominal value of 50%. This should only be a problem for applications that use both the rising and the falling edge of signal CLKOUT.

When using odd values (1..15) for PLLODIV, where PLLODIV = 15 (0Fh) is selected by hardware only during clock system emergency mode or reconfiguration, the duty cycle for signal CLKOUT is on its nominal value of 50%

PLLODIV	0	2	4	6	8	10	12	14
Duty Cycle [%]	45	33.33	40	42.86	44.44	45.45	46.13	46.67

SCU_X.H4 Changing PLLCON in emergency mode

While the clock system is in emergency mode (e.g. after wake-up from sleep, or due to an external clock failure), the clock output divider is set to 16, i.e. PLLODIV = $0F_H$ in register PLLCON. Emergency mode is only terminated if the internal oscillator lock counter has received 2048 clock ticks from XTAL1 after wake-up from sleep mode (when the oscillator was off during sleep).

If PLLCON is written in emergency mode, all settings except bypass modes (PLLCTRL = $0X_B$) become effective immediately within a few clock cycles. As long as the system clock is still derived from the VCO, and if a relatively small value k is written

to PLLODIV, this results in the system running on an internal frequency of f_{VCO}/k that may exceed the specified frequency limit for the device.

In general, it is recommended to wait until PLLODIV < $0F_H$ before PLLCON is written. Use a timeout limit in case a permanent clock failure is present.

SCU_X.H5 Sleep/Idle/Power Down Mode not entered while PLLODIV = $0F_H$

While the clock system is in reconfiguration (e.g. after write to PLLCON, or after wake-up from sleep when an oscillator lock event occurs, or during transition to emergency mode after clock failure), entry into power saving modes is delayed. If e.g. the IDLE instruction to enter sleep mode is executed in this state, the peripherals are already stopped, and the CPU goes into hold state, but the internal clock system will not be switched off until the reconfiguration is complete.

Unless it is guaranteed that the clock system will become stable after a reconfiguration, it is recommended to wait until the clock system is stable (i.e. check for PLLODIV < $0F_H$, use a timeout limit in case a permanent clock failure is present) before executing the IDLE or PWRDN instruction to enter the respective power saving mode.

SCU_X.H6 Interrupt request during entry into sleep mode

After the IDLE instruction has been executed in order to enter sleep mode (SLEEPCON (SYSCON1.1-0) = 01_B), clock system emergency mode (with $f_{VCObase}$ / 16) will become active during the shut down phase before the clock is finally switched off under the following conditions:

- the clock system is not running in bypass mode (PLLCTRL (PLLCON.14-13) = 00_B), **and**
- the RTC is not running on the clock derived from XTAL1 during sleep mode.

If an interrupt request (from an internal or external peripheral module) is generated during this time period, sleep mode is not entered, but instead the associated interrupt service routine is entered. The internal system frequency at that time might not yet be the intended target frequency, since the clock system requires some time to return to its previous state.

To avoid operation on a frequency that is different from the target frequency, either

- do not enter sleep mode (execute the IDLE instruction) while interrupt requests can still occur, **or**
- wait at the beginning of the interrupt service routine until the contents of register PLLCON has returned to the intended target configuration, **or**
- if the interrupt service is not time critical, disable interrupts (IEN (PSW.11) = 0, or select CPU priority level ILVL (PSW.15-12) = $0F_H$) before executing the IDLE instruction. The interrupt service will then be performed after the wake-up from sleep

mode has occurred and the interrupt system has been re-enabled (see also SLEEP_X.H3.2).

RTC_X.H1.2 Resetting and Disabling of the Real Time Clock

Register RTC_CON is not affected by a hardware/software/watchdog reset. After power-up, it is undefined. A reset of the RTC module is achieved by setting bit RTCRST (SYSCON0.15) = 1. This way, register RTC_CON is set to 8003_H (RTC runs, prescaler by 8 enabled).

The RTC clocking mode (synchronous, asynchronous) is determined by bit RTCCM (SYSCON0.14). Note that register SYSCON0 is not affected by a software or watchdog reset. This means that when a software or watchdog reset occurred while the RTC module was in asynchronous mode (selected by bit RTCCM (SYSCON0.14) = 1), it will return to asynchronous mode after a RTC reset triggered by setting bit RTCRST (SYSCON0.15) = 1 with a bit instruction.

For a software or watchdog reset that is followed by an initialization of the RTC module, it is recommended to

- select synchronous RTC clocking mode, i.e. clear bit RTCCM (SYSCON0.14) = 0
- reset the RTC module, i.e. set bit RTCRST (SYSCON0.15) = 1.

This may be achieved with one word or bit field instruction, e.g.

```
    EXTR   #1
    BFLDH  SYSCON0, #0C0h, #80h  ; RTCRST = 1, RTCCM = 0
wait_accpos:
    EXTR   #1
    JNB    ACCPOS, wait_accpos   ; wait until bit ACCPOS = 1
```

When the RTC module is not used and shall be disabled after a (power-on) hardware reset, the following steps are recommended:

1. reset the RTC by setting bit RTCRST (SYSCON0.15) = 1
2. clear the RTC run bit by setting RUN (RTC_CON.0) = 0
3. disable the RTC module by setting bit RTCDIS (SYSCON3.14) = 1.

FOCON_X.H1 Read access to register FOCON

Bit FOTL and bit field FOCNT in register FOCON are marked as 'rh' in the User's Manual, i.e. they can not be modified by software. If register FOCON is read directly after it was written, the value read back from the positions of FOTL and FOCNT represents the value that was written by the preceding instruction, but not the actual contents of FOTL and FOCNT. In order to obtain correct values for FOTL and FOCNT, either insert one NOP

or other instruction that does not write to FOCON, or read FOCON twice and discard the first result.

TwinCAN_X.H2 Reading Bitfield INTID

It is not recommended to use the information stored in bitfield INTID in register AIR/BIR, as it is updated with low priority within the CAN controller. Instead, similar information can be obtained from registers RXIPND and TXIPND.

6 Documentation Update

INT_X.D1 Interrupt Vector Location of CAPCOM Register 28

The vector location for requests from CAPCOM Register 28 is xx'00F0h (not xx'00E0h as documented in User's Manual)

PORTS_X.D2.100 Internal Pull-up Device active on pin P3.12 during Reset

In addition to the pins mentioned in chapter 6.1.4. (System Startup Configuration), an internal pull-up device is active on pin P3.12 during the internal reset phase of each reset (including hardware, software and watchdog timer reset) .

The electrical characteristics of this internal pull-up device is described in section 'DC Characteristics' of the Data Sheet by the parameters 'Level inactive hold current' (symbol I_{LHI}) and 'Level active hold current' (symbol I_{LHA}).

If P3.12 is required to be on a low level during reset, an external pull-down device that meets this specification must be used.

SCU_X.D1.1 Oscillator gain reduction

The gain of the main oscillator (pins XTAL1/2) may be reduced via software by setting bit OSCGRED (SYSCON0.12) = 1. After HW reset, and after wake-up from sleep mode where the main oscillator was switched off, high gain is selected by default to guarantee safe start-up. When OSCGRED has been set to ‘1’ by software, the gain is reduced automatically when bit OSCSTAB (SYSSTAT.11) = 1. Bit OSCSTAB (and OSCLOCK) is cleared after HW reset or after wake-up from sleep when the main oscillator was switched off during sleep (i.e. RTC not running on frequency derived from XTAL1). Bit OSCSTAB is set to ‘1’ after 2^{15} (32768) cycles of f_{XTAL1} that exceed the input hysteresis have been detected.

SCU_X.D2.2 Functionality of register OPSEN

When a breakpoint is hit, the on-chip peripherals selected in register OPSEN are stopped and placed in power-down mode the same way as if disabled via register SYSCON3. Registers of peripherals which are stopped this way can be read, but not written. A read access will not trigger any actions within a disabled peripheral.

The SYSCON3 bits return the shutdown status independently of the reason for the shutdown (static shutdown via SYSCON3 or intermediate shutdown via OPSEN), i.e. when SYSCON3 is read via the debugger after a breakpoint has been hit, it returns the contents of SYSCON3 ORed bitwise with the contents of OPSEN.

It is recommended to leave bit OPSEN.5 (PFLSEN) at its default value '0'. Otherwise, the program flash is deactivated when a breakpoint is hit (i.e. it can not be read), and it has to ramp up when program execution is resumed (i.e. synchronization between software and peripherals is lost).

SCU_X.D3 Register PLLCON after software reset

Register PLLCON is not affected by a software reset, i.e. the current clock configuration remains unchanged. PLLCON may be reconfigured by software, or an endless loop terminated by a watchdog timer reset may be used to force PLLCON to its hardware reset value.

SCU_X.D5 VCO band after hardware/watchdog reset in single chip mode

From the falling edge of $\overline{\text{RSTIN}}$ until 2048 stable oscillator clocks after the rising edge of $\overline{\text{RSTIN}}$, always the lowest VCO band is selected. After that, for a hardware or watchdog timer reset in single-chip mode, PLL bypass mode with the lowest VCO band is selected during the internal reset phase (PLLCON = 2710_H). See also application hint POWER_X.H2.2.

SCU_X.D6 Register Security Mechanism - Unprotected Mode active until execution of EINIT instruction

After reset, the Unprotected Mode is selected by default (bitfield SL = 00_B in register SCUSLS). This allows the initialization software to handle all SFRs (with or without register security mechanism) identically. Unprotected Mode remains effective until execution of the EINIT instruction. After execution of the EINIT instruction, Write Protected Mode becomes active (bitfield SL = 11_B in register SCUSLS). Only after execution of EINIT the security level can be changed to Secured Mode or Write Protected Mode by software.

FLASH_X.D1 Interaction between Program Flash and Security Sector Programming

The on-chip Flash module of the XC16x uses specific internal status information for the Program Flash area and for the Security Sector. This internal status information is

updated with an erase operation or a programming operation (write page command). After reset, always the status information for the program flash is selected.

As documented in the User's Manual, an area in the program flash or security sector must be erased before it is reprogrammed:

"**Caution**: Writing to a flash page (space for the 128-byte buffer) more than once before erasing may destroy data stored in neighbor cells! This is especially important for programming algorithms that do not write to sequential locations."

For the interaction of erase/programming operations to the program or security areas, the following additional rule must be considered:

To ensure correct programming, make sure that a programming operation to either area (program/security) is always preceded by a programming or erase operation to the same area, otherwise wrong data may be stored.

An erase operation to either area (program/security) is always executed correctly, independent of the preceding operation.

In other words:

- make sure that no erase/program operation to the program flash or reset occurs between erasing and programming of an area in the security sector.
- make sure that no erase/program operation to the security sector is executed between erasing and programming of an area in the program flash.

ID-Registers

		Register:	**IDMANUF**	**IDCHIP**	**IDMEM**	**IDPROG**
Device	**Step**	**Address:**	**F07E$_H$**	**F07C$_H$**	**F07A$_H$**	**F078$_H$**
XC164CS-8FF	EES-BA, ES-BA		1820$_H$	2304$_H$	3020$_H$	4040$_H$
XC164CS-8FF	ES+BA, BA		1820$_H$	2304$_H$	3020$_H$	4040$_H$

Product and Test Engineering Group, Munich