Errata Sheet

August 8, 2003 / Release 1.6

Device:	**SAB-C161PI-LM, -L25M** **SAF-C161PI-LM, -L25M** **SAB-C161PI-LM3V** **SAF-C161PI-LM3V** **SAB-C161PI-LF, -L25F** **SAF-C161PI-LF, -L25F** **SAB-C161PI-LF3V** **SAF-C161PI-LF3V**
Stepping Code / Marking:	**ES-AA, AA** **ES-BA-H, BA-H**
Package:	**P-MQFP-100-2, P-TQFP-100-1**

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.

The current documentation is: Data Sheet: C161PI Data Sheet 1999-07
User's Manual: C161PI User's Manual V1.0 1999-08
Instruction Set Manual V2.0, 2001-03

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.

Change summary to Errata Sheet for Device Step AA, BA-H, Rel.1.5:

- Versions SAF-C161PI-LM3V and SAF-C161PI-LF3V included
- Text module RST18.4 (Undefined Clock Configuration after Power-On) included, summarizes/replaces RST.13 (Power Up with missing clock) and POWER.16 (SDD and Sleep Mode cannot be terminated)

- Z Flag after PUSH and PCALL (CPU.22)
- Sleep Mode with RTC on cannot be terminated (POWER.17)
- Note on Early (Unlatched) Chip Select Option (Section Application Hints)
- PLL lock after temporary clock failure (Section Application Hints)

Functional Problems:

ADC.11: Modifications of ADM field while bit ADST = 0

The A/D converter may unintentionally start one auto scan single conversion sequence when the following sequence of conditions is true:

(1) the A/D converter has finished a fixed channel single conversion of an analog channel n > 0 (i.e. contents of ADCON.ADCH = n during this conversion)
(2) the A/D converter is idle (i.e. ADBSY = 0)
(3) then the conversion mode in the ADC Mode Selection field ADM is changed to Auto Scan Single (ADM = 10b) or Continuous (ADM = 11b) mode without setting bit ADST = 1 with the same instruction

Under these conditions, the A/D converter will unintentionally start one auto scan single conversion sequence, beginning with channel n-1, down to channel number 0.

In case the channel number ADCH has been changed before or with the same instruction which selected the auto scan mode, this channel number has no effect on the unintended auto scan sequence (i.e. it is not used in this auto scan sequence).

Note:
When a conversion is already in progress, and then the configuration in register ADCON is changed,

- the new conversion mode in ADM is evaluated after the current conversion
- the new channel number in ADCH and new status of bit ADST are evaluated after the current conversion when a conversion in fixed channel conversion mode is in progress, and after the current conversion sequence (i.e. after conversion of channel 0) when a conversion in an auto scan mode is in progress.

In this case, it is a specified operational behaviour that channels n-1 .. 0 are converted when ADM is changed to an auto scan mode while a fixed channel conversion of channel n is in progress (see e.g. C167 User's Manual, V2.0, p16-4)

Workaround:

When an auto scan conversion is to be performed, always start the A/D converter with the same instruction which sets the configuration in register ADCON.

PRT.3: Special Thresholds for Port 2 pins and P3.0, P3.1

When bit P2HIN = 1 in register PICON, the Port 2 pins will still recognize external input signals according to the standard (TTL) thresholds V_{IL} and V_{IH} without hysteresis.

When bit P3LIN = 1 in register PICON, the Port 3 pins P3.0 and P3.1 will recognize external input signals according to the specific threshold V_{IL1} and V_{IH} for I^2C bus operation.

PWRDN.1: Execution of PWRDN Instruction while pin NMI# = high

When instruction PWRDN is executed while pin NMI# is at a high level, power down mode should not be entered, and the PWRDN instruction should be ignored. However, under the conditions described below, the PWRDN instruction may not be ignored, and no further instructions are fetched from external memory, i.e. the CPU is in a quasi-idle state. This problem will only occur in the following situations:

a) the instructions following the PWRDN instruction are located in external memory, and a **multiplexed bus** configuration **with memory tristate waitstate** (bit MTTCx = 0) is used, or
b) the instruction preceding the PWRDN instruction **writes** to external memory or an XPeripheral (XRAM, CAN), and the instructions following the PWRDN instruction are located in external memory. In this case, the problem will occur for any bus configuration.

Note: the on-chip peripherals are still working correctly, in particular the Watchdog Timer will reset the device upon an overflow. Interrupts and PEC transfers, however, can not be processed. In case NMI# is asserted low while the device is in this quasi-idle state, power down mode is entered.

Workaround:

Ensure that no instruction which writes to external memory or an XPeripheral precedes the PWRDN instruction, otherwise insert e.g. a NOP instruction in front of PWRDN. When a multiplexed bus with memory tristate waitstate is used, the PWRDN instruction should be executed out of internal RAM or XRAM.

POWER.14: Wake Up from Sleep Mode not possible in PLL Mode

Sleep mode can not be terminated by external interrupt (NMI or fast external interrupts including alternate sources – see register EXISEL).

Workarounds:

- Avoid sleep mode with PLL mode (use deep idle instead: all peripherals off and slow down divider/32 and idle mode)
- Use HW reset instead of interrupt

For devices with register RSTCON[1)] only: Switch to direct drive mode before selecting sleep mode.
([1)] RSTCON currently not implemented in C161PI)

POWER.15: Sleep Mode not possible in Prescaler Mode

In prescaler mode ($f_{CPU} = f_{OSC}/2$) instead of sleep the device enters an undefined state and no wake up is possible except HW reset.

Workaround:

- Avoid sleep mode with prescaler mode (use deep idle instead: all peripherals off and slow down divider/32 and idle mode)
- For devices with register RSTCON[1)] only: Switch to direct drive mode before selecting sleep mode (but take care on max. f_{CPU}) ([1)] RSTCON currently not implemented in C161PI)

POWER.16: SDD and Sleep Mode cannot be terminated

After each power-on a problem with the internal clock lock detection can occur. Due to this the Slow Down Mode and the Sleep Mode (for derivatives with Sleep Mode implemented only) cannot be terminated (as they wait for the clock lock condition). It is not possible to switch back the clock path from Slow Down Mode to Direct Drive-, Prescaler- or PLL-Mode.

Note: In this case the bit CLKLOCK in register SYSCON2 always remains cleared.

Workaround:

(1) Direct Drive or Prescaler Mode:
Instead of switching back to Direct Drive or Prescaler Mode it is recommended to stay in Slow Down Mode and to set the Slow Down Factor to One (SYSCON2: CLKREL = 0000b).

(2) PLL Mode:
It is recommended to increase the external oscillator frequency via hardware, up to CPU frequency and to use Direct Drive Mode with workaround (1)

POWER.17: Sleep Mode with RTC on cannot be terminated

After the wake up signal from sleep mode is recognized, the device enables the clock system. However, the device does not wait for stable PLL clock but wakes up the CPU and peripherals before. Due to this the device remains in sleep mode.

Workarounds:

- Select direct drive mode.
- Select sleep mode with RTC off.

X9: Read Access to XPERs in Visible Mode

The data of a read access to an XBUS-Peripheral (XRAM, I^2C) in Visible Mode is not driven to the external bus. PORT0 is tristated during such read accesses.

Note that in Visible Mode PORT1 will drive the address for an access to an XBUS-Peripheral, even when only a multiplexed external bus is enabled.

CPU.21 BFLDL/BFLDH Instructions after Write Operation to internal IRAM

The result of a BFLDL/BFLDH (=BFLDx) instruction may be incorrect if the following conditions are true at the same time:

(1) the previous 'instruction' is a PEC transfer which writes to IRAM, or any instruction with result write back to IRAM (addresses 0F200h..0FDFFh for 3 Kbyte module, 0F600h..0FDFFh for 2 Kbyte module, or 0FA00h..0FDFFh for 1 Kbyte module). For further restrictions on the destination address see case (a) or case (b) below.

(2) the BFLDx instruction immediately follows the previous instruction or PEC transfer within the instruction pipeline ('back-to-back' execution), i.e. decode phase of BFLDx and execute phase of the previous instruction or PEC transfer coincide. This situation typically occurs during program execution from internal program memory (ROM/OTP/Flash), or when the instruction queue is full during program execution from external memory

(3) the 3rd byte of BFLDx (= ***#mask8*** field of BFLD**L** or ***#data8*** field of BFLD**H**) and the destination address of the previous instruction or PEC transfer match in the following way:

(a) value of #mask8 of BFLDL or #data8 of BFLDH = 0F**y**h (**y** = 0..Fh),
and the previous instruction or PEC writes to (the low and/or high byte of) GPR R**y** or the memory address of R**y** (determined by the context pointer CP) via any addressing mode.

(b) value of #mask8 of BFLDL or #data8 of BFLDH = **00h..0EFh**,
and the lower byte $\mathbf{v}_L$ of the **contents v** of the IRAM location or (E)SFR or GPR which is read by BFLDx is $\mathbf{00h \leq v_L \leq 7Fh}$
and the previous instruction or PEC transfer writes to the (low and/or high byte of) the specific bit-addressable IRAM location $\mathbf{0FD00h + 2\ v_L}$ (i.e. the 8-bit offset address of the location in the bit-addressable IRAM area (0FD00h..0FDFFh) equals $\mathbf{v_L}$).

When the problem occurs, the actual result (all 16 bits) of the BFLDx instruction is bitwise ORed with the (byte or word) result of the previous instruction or PEC transfer.

Notes:

Write operations in the sense of the problem description include implicit write accesses caused by
- auto-increment operations of the PEC source or destination pointers (which are located on 0FCE0h..0FCFEh in IRAM)
- post-increment/pre-decrement operations on GPRs (addressing modes with [R+] or [-R])
- write operations on the system stack (which is located in IRAM).

In case **PEC write operations** to IRAM locations which match the above criteria (bit-addressable or active register bank area, PEC pointers not overlapping with register bank area) can be **excluded**, the problem will **not** occur when the instruction preceding BFLDx in the dynamic flow of the program is one of the following instructions (which do not write to IRAM):

NOP
ATOMIC, EXTx
CALLA/CALLI/JBC/JNBS when branch condition = false
JMPx, JB, JNB
RETx (except RETP)
CMP(B) (except addressing mode with [Rwi+]), BCMP
MULx, DIVx
IDLE, PWRDN, DISWDT, SRVWDT, EINIT, SRST

For implicit IRAM write operations caused by **auto-increment operations of the PEC source or destination pointers**, the problem can only occur if the value of #mask8 of BFLDL or #data8 of BFLDH = 0F**y**h (**y** = 0..Fh), and the range which is covered by the context pointer CP (partially or completely) overlaps the PEC source and destination pointer area (0FCE0h..0FCFEh), and the address of the source or destination pointer which is auto-incremented after the PEC transfer is equal to the address of GPR R**y** (included in case 3a).

For **system stack write operations**, the problem can only occur if the system stack is located in the bit-addressable portion of IRAM (0FD00h..0FDFFh), or if the system stack can overlap the register bank area (i.e. the register bank area is located below the system stack, and the distance between the contents of the context pointer CP and the stack pointer SP is ≤ 20h).

Workaround 1:

When a critical instruction combination or PEC transfer to IRAM can occur, then substitute the BFLDx instruction by

(a) an equivalent sequence of single bit instructions. This sequence may be included in an uninteruptable ATOMIC or EXTEND sequence to ensure completion after a defined time.

(b) an equivalent byte- or word MOV or logical instruction.

Note that byte operations to SFRs always clear the non-addressed complementary byte.
Note that protected bits in SFRs are overwritten by MOV or logical instructions.

Workaround 2:

When a critical instruction combination occurs, and **PEC write operations** to IRAM locations which match the above criteria (bit-addressable or active register bank area) **can be excluded**, then rearrange the BFLDx instruction within the instruction environment such that a non-critical instruction sequence is generated.

Workaround 3:

When a critical instruction combination or PEC transfer to IRAM can occur, then

- replace the BFLDx instruction by the instruction sequence
  ATOMIC #1
  BFLDx
  This means e.g. when BFLDx was a branch target before, ATOMIC # 1 is now the new branch target.

In case the BFLDx instruction is included at position ***n*** in an ATOMIC or EXTEND sequence with range operator #**m** (n, m = 2..4, n ≤ m), then

- insert (repeat) the corresponding ATOMIC or EXTEND instruction at position ***n*** with range operator #**z** where **z** = (***m*** - ***n***) + 1

	Range of original ATOMIC/EXTEND statement			
Position of BFLDx within ATOMIC/ EXT.. sequence	1	2	3	4
1	no problem / no workaround	no problem / no workaround	no problem / no workaround	no problem / no workaround
2	- -	z = 1	z = 2	z = 3
3	- -	- -	z = 1	z = 2
4	- -	- -	- -	z = 1

- - : case can not occur

Note on devices with power management and register RSTCON:

for unlock sequences, which are sequences of four instructions operating on ESFRs SYSCON1/2/3 and RSTCON and which are included in an EXTR #4 sequence, **this workaround must not be implemented when the 4th instruction is a BFLDx instruction**, otherwise the unlock sequence will not work correctly (in fact unlock sequences don't require a workaround).

Tool Support

The **Keil** C166 Compiler V3.xx generates BFLD instructions only in the following cases:

- when using the _bfld_ intrinsic function.
- at the beginning of interrupt service routines, when using #pragma disable
- at the end of interrupt service routines, when using the chip bypass directive FIX166.

The C166 Compiler V4.xx uses the BFLD instruction to optimize bit-field struct accesses. Release C166 V4.10 offers a new directive called FIXBFLD that inserts an ATOMIC #1 instruction before every BFLD instruction that is not enclosed in an EXTR sequence. Detailed information can be found in the C166\HLP\RELEASE.TXT of C166 Version 4.10.

The C166 Run-Time Library for C166 V3.xx and V4.xx uses BFLD instructions only in the START167.A66 file. This part of the code should be not affected by the CPU.21 problem but should be checked by the software designer.

The RTX166 Full Real-Time Operating system (any version) does not use BFLD instructions.
For RTX166 Tiny, you should rebuild the RTX166 Tiny library with the SET FIXBFLD = 1 directive. This directive is enabled in the assembler source file RTX166T.A66. After change of this setting rebuild the RTX166 Tiny library that you are using in your application.

The **Tasking** support organization provides a v7.0r1 A166 Assembler (build 177) including a check for problem CPU.21 with optional pec/no_pec feature. This assembler version can also be used to check code which was generated with previous versions of the Tasking tool chain. A v7.0r1 C166 Compiler (build 368) offering a workaround for problem CPU.21 is also available from Tasking.

The scan tool **aiScan21** analyzes files in hex format plus user-supplied additional information (locator map file, configuration file), checks whether they may be affected by problem CPU.21, and produces diagnostic information about potentially critical instruction sequences. This tool is included in AP1628 'Scanning for Problem CPU.21' on
http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8137

CPU.22: Z Flag after PUSH and PCALL

The Z flag in the PSW is erroneously set to '1' by ***PUSH reg*** or ***PCALL reg, rel*** instructions when all of the following conditions are true:

(a) for ***PUSH reg*** instructions:
- the contents of the high byte of the GPR or (E)SFR which is pushed is 00h, and
- the contents of the low byte of the GPR or (E)SFR which is pushed is > 00h, and
- the contents of GPR Rx is odd, where x = 4 msbs of the 8-bit 'reg' address of the pushed GPR or (E)SFR

Examples:

`PUSH R1`	(coding: **F**1 EC):	incorrect setting of Z flag if contents of R**15** is odd, and 00FFh ≥ contents of R1 ≥ 0001h
`PUSH DPP3`	(coding: **0**3 EC):	incorrect setting of Z flag if contents of R**0** is odd, and 00FFh ≥ contents of DPP3 ≥ 0001h

(b) for ***PCALL reg, rel*** instructions:
- when the contents of the high byte of the GPR or (E)SFR which is pushed is 00h, and
- when the contents of the low byte of the GPR or (E)SFR which is pushed is odd

This may lead to wrong results of instructions following PUSH or PCALL if those instructions explicitly (e.g. BMOV .. , Z; JB Z, ..; ..) or implicitly (e.g. JMP cc_Z, ..; JMP cc_NET, ..; ..) evaluate the status of the Z flag before it is newly updated.

Note that some instructions (e.g. CALL, ..) have no effect on the status flags, such that the status of the Z flag remains incorrect after a PUSH/PCALL instruction until an instruction that correctly updates the Z flag is executed.

Example:

```
   PUSH R1        ; incorrect setting of Z flag if R15 is odd
   CALL proc_xyz  ; Z flag remains unchanged (is a parameter for proc_xyz)
   ...
proc_xyz:
   JMP cc_Z,end_xyz ; Z flag evaluated with incorrect setting
   ...
end_xyz:
```

Effect on Tools:

The **Hightec** C166 tools (all versions) don't use the combination of PUSH/PCALL and the evaluation of the Z flag. Therefore, these tools are not affected.

The code generated by the **Keil** C166 Compiler evaluates the Z flag only after MOV, CMP, arithmetic, or logical instructions. It is never evaluated after a PUSH instruction. PCALL instructions are not generated by the C166 Compiler.

This has been checked with all C166 V3.xx and V4.xx compiler versions. Even the upcoming V5.xx is not affected by the CPU.22 problem.

The assembler portions of the C166 V3.xx and V4.xx Run-Time Libraries, the RTX166 Full and TX166 Tiny Real Time Operating system do also not contain any evaluation of the Z flag after PUSH or PCALL.

The **TASKING** compiler V7.5r2 never generates a PCALL instruction, nor is it used in the libraries. The PUSH instruction is only used in the entry of an interrupt frame, and sometimes on exit of normal functions. The zero flag is not a parameter or return value, so this does not give any problems.

Previous versions of TASKING tools: V3.x and higher are not affected, versions before 3.x are most likely not affected. Contact TASKING when using versions before V3.x.

Since code generated by the C166 **compiler** versions mentioned before is **not** affected, analysis and workarounds are **only** required for program parts written in **assembler**, or instruction sequences inserted via inline assembly.

Workaround (for program parts written in assembler):

Do not evaluate the status of the Z flag generated by a PUSH or PCALL instruction. Instead, insert an instruction that correctly updates the PSW flags, e.g.

```
   PUSH reg
   CMP reg, #0         ; updates PSW flags
                       ; note: CMP additionally modifies the C and V flags,
                       ; while PUSH or MOV leaves them unaffected
   JMPR cc_Z, label_1  ; implicitly tests Z flag

or
   PCALL reg, procedure_1
   ...
procedure_1:
   MOV ONES, reg       ; updates PSW flags
   JMPR cc_NET, label_1; implicitly tests flags Z and E
```

Hints for Detection of Critical Instruction Combinations

Whether or not an instruction following *PUSH reg* or *PCALL reg, rel* actually causes a problem depends on the program context. In most cases, it will be sufficient to just analyze the instruction following PUSH or PCALL. In case of PCALL, this is the instruction at the call target address.

- Support Tool for Analysis of Hex Files

For complex software projects, where a large number of assembler source (or list) files would have to be analyzed, Infineon provides a tool **aiScan22** which scans hex files for critical instruction sequences and outputs diagnostic information. This tool is available as part of the Application Note **ap1679** 'Scanning for Problem CPU.22' on the 16-bit microcontroller internet pages of Infineon Technologies:
http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8137

direct links:
http://www.infineon.com/cmc_upload/documents/040/841/ap1679_v1.1_2002_05_scanning_cpu22.pdf
http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=40840&parent_oid=-8137

- Individual Analysis of Assembler Source Code

With respect to problem CPU.22, all instructions of the C166 instruction set can be classified into the following groups:

- **Arithmetic/logic/data movement** instructions as successors of PUSH/PCALL (correctly) modify the condition flags in the PSW according to the result of the operation.

- These instructions may **only** cause a problem if the **PSW** is a source or source/destination operand:

  ADD/B, ADDC/B, CMP/B, CMPD1/2, CMPI1/2, SUB/B, SUBC/B
  AND/B, OR/B, XOR/B
  ASHR
  MOV/B, MOVBZ/MOVBS
  SCXT
  PUSH, PCALL ; analysis must be repeated for successor of PUSH/PCALL

- The following instructions (most of them with immediate or register (Rx) addressing modes) can **never** cause a problem when they are successors of PUSH/PCALL:

  CPL/B, NEG/B
  DIV/U, DIVL/U, MUL/U
  SHL/SHR, ROL/ROR, PRIOR
  POP
  RETI ; updates complete PSW with stacked value
  RETP ; updates condition flags
  PWRDN ; program restarts after reset
  SRST ; program restarts

- Conditional branch instructions which may evaluate the Z flag as successors of PUSH/PCALL:

  JB/JNB Z, *rel* ; directly evaluates Z flag
  CALLA/CALLI, JMPA/JMPI/JMPR with the following condition codes
  cc_Z, cc_EQ, cc_NZ, cc_NE
  cc_ULE, cc_UGT, cc_SLE, cc_SGT
  cc_NET

- For these branch conditions, the branch may be performed in the wrong way.
- For other branch conditions, the branch target as well as the linear successor of the branch instruction must be analyzed (since these branch instruction don't modify the PSW flags).

- For **instructions that have no effect on the condition flags** and that don't evaluate the Z flag, the instruction that follows this instruction must be analyzed. These instructions are

  NOP
  ATOMIC, EXTxx
  DISWDT, EINIT, IDLE, SRVWDT
  CALLR, CALLS, JMPS ; branch target must be analyzed
  RET, RETS ; return target must be analyzed (value pushed by PUSH/PCALL = return IP,
  ; Z flag contains information whether intra-segment target address = 0000h or not)
  TRAP ; both trap target and linear successor must be analyzed, since Z flag may be
  ; incorrect in PSW on stack as well as in PSW at entry of trap routine

- For **bit modification instructions**, the problem may only occur if a source bit is the Z flag, and/or the destination bit is in the PSW, but not the Z flag. These instructions are:

  BMOV/BMOVN
  BAND/BOR/BXOR
  BCMP
  BFLDH
  BFLDL ; problem only if bit 3 of @@ mask = 0, i.e. if Z is not selected
  BCLR/BSET ; problem only if operand is not Z flag
  JBC/JNBS ; wrong branch if operand is Z flag

RST.18.4 Undefined Clock Configuration after Power-On

Problem Description

During power-on or during a power off/on sequence it is possible that the internal clock configuration of the power management is not set to a defined state. An undefined state of the power management clock configuration can cause different kinds of malfunctions in the system. The below described malfunction does not always occur but can appear sporadically.

Conditions to get a defined Clock Configuration at Power-On

During power-on or during a power off/on sequence all of the following three conditions must be fulfilled that the internal oscillator lock signal can be set and the real time clock registers will be cleared (power-on detection OK)

- Power off/on recovery time must be 4 seconds or longer.
- The input voltage V_{IN} at all pins has to be $V_{IN} \leq 0.3$ V for the duration of the power off/on recovery time.
- During power-on (rise time of V_{DD}) RSTIN# has to be active and is required at least until the clock at XTAL1 is stable. The input level of RSTIN# has to be
  $V_{RSTIN\#} \leq 0.3 * V_{DD}$ for $0 \text{ V} \leq V_{DD} < 2.7$ V and
  $V_{RSTIN\#} \leq 0.8$ V for $2.7 \text{ V} \leq V_{DD} \leq 4.5$ V.

Defined state of the Power Management Clock Configuration after Power-On

After a power-on which fulfills the three mentioned conditions the internal oscillator lock signal is set after oscillator start-up and the real time clock registers are cleared (T14, T14REL, RTCH RTCL).

Root cause of the Problem:

If the conditions to get a defined clock configuration at power-on are not fulfilled then it is possible that the internal power-on detection unit does not detect power-on. Whether power-on was detected or not can be checked with WDTCON.5. If power-on was detected then WDTCON.5 = 1 else WDTCON.5 = 0. The function of bit WDTCON.5 is implemented for evaluation purposes and is not guaranteed.

If power-on was not detected by the on-chip power-on detection unit then this can cause a malfunction of the internal oscillator lock detect unit. In that case the internal oscillator lock signal is not set after 2048 XTAL1 clock periods.

Effects on the system in case of an undefined Power Management Clock Configuration and possible Workarounds

Because of the high flexibility of the clock configuration during and after power-on, different cases of clock sources and clock paths have to be considered.

Case 1: This is only included because of Compatibility of Numbering

Case 2: This is only included because of Compatibility of Numbering

Case 3: Clock @ XTAL1 and Power-on Detection FAIL

When the power-on detection fails during power-on because of not considering the necessary conditions, the register bit CLKLOCK and the RTC registers are undefined. Further the oscillator lock signal is not set.

When the system is running in direct drive or PLL clock mode, it is possible to switch the clock path to SDD mode via software. But it is NOT possible to switch back to the clock path of direct drive or PLL clock mode (basic clock) because the switch back mechanism will wait for the oscillator lock signal.

Workarounds:

- Consider the conditions to get a defined clock configuration at power-on
- In systems where direct drive or prescaler mode is used it is possible to set the SDD to factor one or two and to stay in SDD mode (no oscillator watchdog function possible).

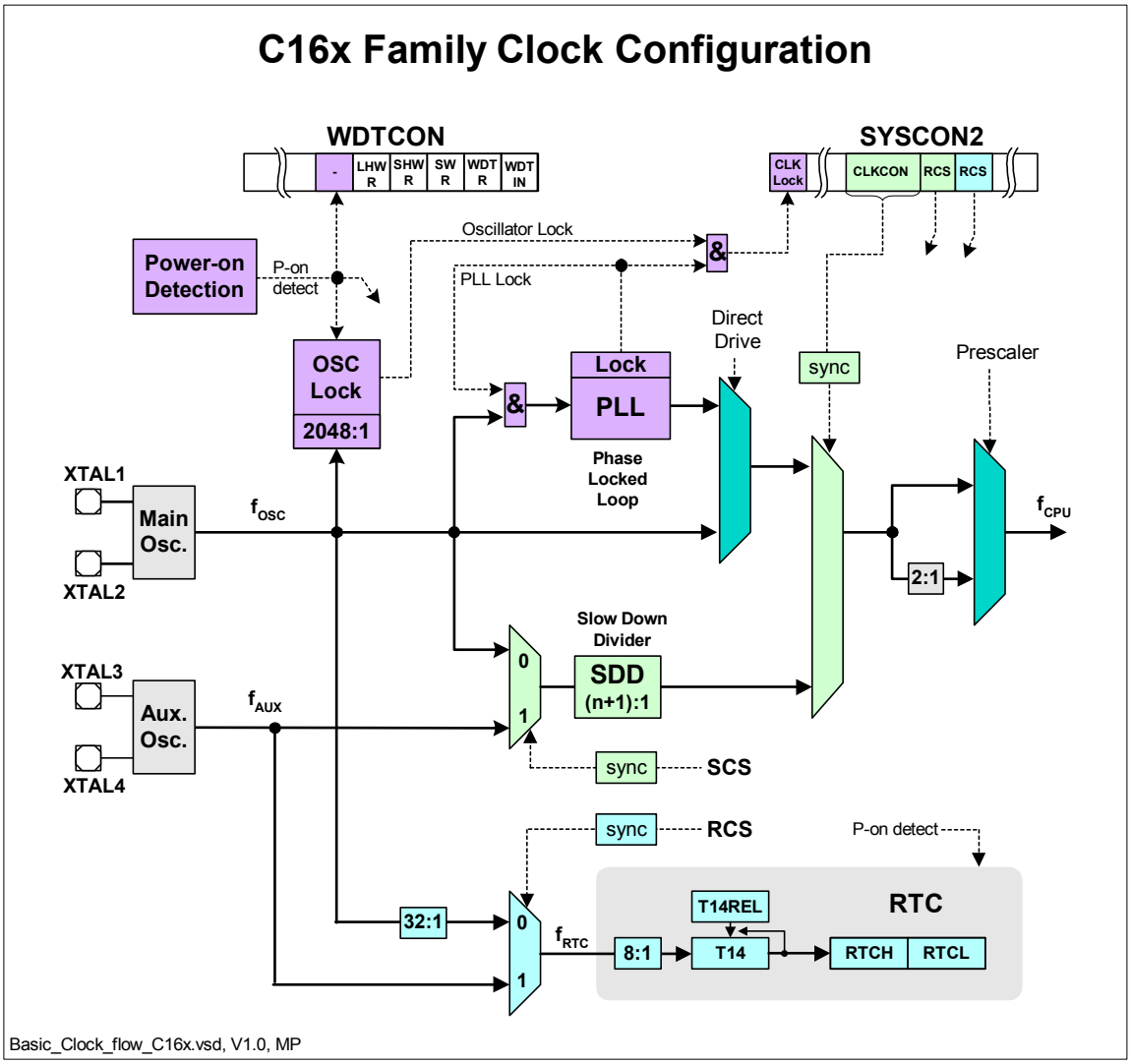
Case 4: NO Clock @ XTAL1 and Power-on Detection FAIL

If the system gets no clock at XTAL1 during power-on and the power-on detection fails then the following settings can be found:

- The real time clock registers are undefined (T14, T14REL, RTCH, RTCL).
- The slow down divider (SDD) may be selected as f_{CPU} clock source (no oscillator watchdog function possible). In that case the system will stop.

Workaround:

Consider the conditions to get a defined clock configuration at power-on.



Note: The C161PI includes no auxiliary oscillator. Bits SCS and RCS are set to zero.

RST.13: Power Up with missing Clock

While no clock is provided at XTAL1 **and**
- RSTIN = High (while $V_{DD} \leq$ 2V) **or**
- power off/on recovery time is below 1 second (beginning from ≤ 0.3V at all pins)

then the following settings can be found:

- The realtime clock registers are not cleared (T14, T14REL, RTCH, RTCL).
- The slow down divider (SDD) can be selected as f_{CPU} clock source (no oscillator watchdog function possible – the device will stop while external clock is missing)

Workaround: see RST18.4, case 4

RST.9.3: Software/Watchdog/Short HW Reset during Slow Down operation with PLL off

When a watchdog timer or software reset or short hardware reset occurs during slow down mode where the PLL has been switched off (CLKCON = 10 in register SYSCON2), the internal reset condition is extended until the next long HW reset.

After the reset, if the PLL has not locked, flag PLLIR in register ISNC **may not be set correctly, but** the device continues operation on the PLL base frequency (2 .. 5 MHz).

This problem only occurs when the PLL is used to generate the internal clock during normal operation, and it will not occur in direct drive or prescaler mode.

Workarounds:

Enable bi-directional reset mode (bit BDRSTEN = 1 in register SYSCON). The external reset circuit should prolong this reset until the PLL is locked again (approx. 1 ms).

RST.8: Clock failure detection during external Reset

A missing clock at pin XTAL1 during an external reset cannot be recognized via bit PLLIR (PLL/OWD interrupt request flag) in register ISNC (Interrupt Subnode Control) because bit PLLIR is cleared during reset. When the clock at pin XTAL 1 is missing the internal CPU clock is the PLL base frequency (2...5 MHz).

Workaround:

(1) Clock options at PORT0 (P0H.7 ... 5) are set to Direct Drive or Prescaler mode:

Use the Real Time Clock with main oscillator (default after Power-on reset) as input clock source. Check bit RTCIR (T14IR) of the Real Time Clock instead of bit PLLIR. If the RTC interrupt request does not occur in the expected time frame then the main oscillator does not oscillate.

Instead of an interrupt controlled verification of the RTC activity also polling of register T14 for at least 256 XTAL1 cycles can be used to check whether the main oscillator is running.

(2) Clock options at PORT0 (P0H.7 ... 5) are set to PLL mode:

Bit CLKLOCK in SYSCON2 can be tested instead of bit PLLIR. CLKLOCK = 0 while the PLL is unlocked. The workaround described in case (1) can also be used instead of the CLKLOCK bit.

Note: not all problems listed in the Errata Sheet may actually occur for every individual device of the C161PI. However, it is recommended to follow the workarounds given in the Errata Sheet to ensure stable production.

Deviations from Electrical- and Timing Specification:

The following table lists the deviations of the DC/AC characteristics from the specification in the C161PI Data Sheet 1999-07:

Problem short name	Parameter	Symbol	Limit Values		Unit	Test Condition
			min.	max.		
AC.t34.n2	CLKOUT rising edge to ALE falling edge	t_{34}	**-2** + t_A instead of 0 + t_A	**-**	ns	Standard Supply Voltage
AC.t34.n10	CLKOUT rising edge to ALE falling edge	t_{34}	**-10** + t_A instead of 0 + t_A	8 + t_A	ns	Reduced Supply Voltage
AC.t33.10	CLKOUT fall time	t_{33}	**-**	**10** instead of 4 (8)	ns	Standard (Reduced) Supply Voltage
DC.VDDRM.1	Reduced digital supply voltage	V_{DD}	**3.15** instead of 3.0	3.6	V	Devices with date code ≥ 0141

Notes:

1) During **reset** and **adapt mode**, the **internal pull-ups on P6.[4:0]** are active, independent whether the respective pins are used for CS# function after reset or not.

Application Hints

Maintenance of ISNC Register

The RTC and PLL interrupts share one interrupt node (XP3IC). If an interrupt request occurs, the request bit in the Interrupt Subnode Control register has to be checked and cleared by software. To avoid a collision with the next hardware interrupt request of the same source, it is recommended to first clear the request and the enable bit, and then to set the enable bit again.

- Example for a XP3 interrupt service routine (for Tasking C compiler):

```
...
if (PLLIR)
      {
      _bfld (ISNC, 0x000C, 0x0000); // clear PLLIE and PLLIR
      _putbit (1, ISNC, 3);         // set PLLIE
      ...                           // further actions concerning PLL/OWD
      }
if (RTCIR)
      {
      _bfld (ISNC, 0x0003, 0x0000); // clear RTCIE and RTCIR
      _putbit (1, ISNC, 1);         // set RTCIE
      ...                           // further actions concerning RTC
      }
...
```

- In assembly language:

```
...
  EXTR      #1
  JNB       PLLIR, no_pll_request
  EXTR      #2          ; no further interruption of this sequence possible
  BFLDL     ISNC, #0Ch, #00h        ; clear PLLIE and PLLIR
  BSET      PLLIE                   ; set PLLIE
...                                 ; further actions concerning PLL/OWD
no_pll_request:
  EXTR      #1
  JNB       RTCIR, no_rtc_request
  EXTR      #2          ; no further interruption of this sequence possible
  BFLDL     ISNC, #03h, #00h        ; clear RTCIE and RTCIR
  BSET      RTCIE                   ; set RTCIE
...                                 ; further actions concerning RTC
no_rtc_request:
...
```

Maximum (Main: Type_LP2) Oscillator Frequency = 16 MHz

The main oscillator is optimized for oscillation with a crystal within a frequency range of 4...16 MHz. When driven by an external clock signal it will accept the specified frequency range. Operation at lower input frequencies is possible but is guaranteed by design only (not 100% tested).

Main Oscillator Type_LP2: Negative Resistance and Start-up Reliability

Compared to other C16x microcontrollers the gain of the on-chip oscillator (Type_LP2) is slightly different. It is recommended to check the negative resistance and the start-up reliability of the oscillator circuit in the original application. Please refer to the limits specified by the quartz crystal or ceramic resonator supplier.

See also Application Note AP2420 'Crystal Oscillator of the C500 and C166 Microcontroller Families' and Application Note AP2424 'Ceramic Resonator Oscillators of the C500 and C166 Microcontroller Families' on http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8137

Oscillator Watchdog and Prescaler Mode

The OWD replaces the missing oscillator clock signal with the PLL clock (base frequency).
In direct drive mode the PLL base frequency is used directly (f_{CPU} = 2...5 MHz).
In prescaler mode the PLL base frequency is divided by 2 (f_{CPU} = 1...2.5 MHz).

PLL lock after temporary clock failure

When the PLL is locked and the input clock at XTAL1 is interrupted then the PLL becomes unlocked, provides the base frequency (2 ... 5 MHz) and the PLL unlock interrupt request flag is set. If the XTAL1 input clock starts oscillation again then the PLL stays in the PLL base frequency. The CPU clock source is only switched back to the XTAL1 oscillator clock after a hardware reset. This can be achieved via a normal hardware reset or via a software reset with enabled bidirectional reset. It is important that the hardware reset is at least active for 1 ms, after that time the PLL is locked in any case.

Handling of the SSC Busy Flag (SSCBSY)

In master mode of the High-Speed Synchronous Serial Interface (SSC), when register SSCTB has been written, flag SSCBSY is set to '1' when the baud rate generator generates the next internal clock pulse. The maximum delay between the time SSCTB has been written and flag SSCBSY=1 is up to 1/2 bit time. SSCBSY is cleared 1/2 bit time after the last latching edge.

When polling flag SSCBSY after SSCTB has been written, SSCBSY may not yet be set to '1' when it is tested for the first time (in particular at lower baud rates). Therefore, e.g. the following alternative methods are recommended:

1. test flag SSCRIR (receive interrupt request) instead of SSCBSY (in case the receive interrupt request is not serviced by CPU interrupt or PEC), e.g.

```
loop:           BCLR SSCRIR                 ;clear receive interrupt request flag
                MOV SSCTB, #xyz             ;send character
wait_tx_complete:
                JNB SSCRIR, wait_tx_complete ;test SSCRIR
                JB SSCBSY, wait_tx_complete     ;test SSCBSY to achieve original
                                                timing(SSCRIR may be set 1/2 bit time before
                                                SSCBSY is cleared)
```

2. use a software semaphore bit which is set when SSCTB is written and is cleared in the SSC receive interrupt routine

Maximum I^2C Bus Data Rate at low f_{CPU}

The I^2C Bus module was designed for a high data rate (400 Kbit/s @ f_{CPU} = 20 MHz). All module internal clocks are derived from f_{CPU}. When a lower CPU frequency is used system limitations have to be considered like maximum data rate and clock symmetry. These effects may occur in particular for values of the baud rate prescaler BRP ≤ 5. More details are under evaluation.

Overload Protection Circuit for I^2C Bus Pins P3.0/P3.1, P6.5 ..P6.7

Since these pins are used for alternate functions by the I^2C Bus module, they have a different internal protection circuit than the other I/O pins in order to avoid interference with the I^2C bus lines when the microcontroller is switched off. As a consequence, the input voltage on these pins must not exceed Vdd + 0.5 V, independent of any current limitation. When Vin < Vss - 0.5 V, the overload current must be limited to 5 mA (Operating Conditions) or 10 mA, respectively (Absolute Maximum Ratings), see C161PI Data Sheet 1999-07, p. 36/37.

Note on Early (Unlatched) Chip Select Option

As described in the User's Manuals (e.g. C161PI User's Manual, V1.0, 1999-08, p.9-11), an early (unlatched) address chip select signal (SYSCON.CSCFG = ‘1’) becomes active together with the address and BHE (if enabled) and remains active until the end of the current bus cycle. Early address chip select signals are not latched internally and may toggle intermediately while the address is changing.

These effects may also occur on CSx# lines which are configured as RDCSx# and/or WRCSx# signals (BUSCONx.CSRENx = 1 and/or CSWENx = 1).

The position of these transitions (spikes) is at the beginning of an external bus cycle or an internal XBUS cycle, indicated by the rising edge of signal ALE. The width of these transitions is ~ 5 ns (measured at a reference level of 2.0 V with Vdd = 5.0 V). The falling edge of the spike occurs in the same relation to RD#, WR#/WRH#/WRL# and to other CS# signals as if it was an address chip select signal with early chip select option.

When CS# lines configured as RDCS# and/or WRCS# are used e.g. as output enable (OE#) signals for external devices or as clock input for shift registers, problems might occur (temporary bus contention during data float times (may be solved by tristate wait state), unexpected shift operations, etc.). When CS# lines configured as WRCS# are used as write enable (WE#) signals for external devices or FIFOs, internal locations may be overwritten with undefined data.

When CS# lines are used as chip enable (CE#) signals for external memories, usually no problems are expected, since the falling edge of the spikes has the same characteristics as the falling edge of an access with a regular early (unlatched) address CS# signal. At this time, the memory control signals RD#, WR# (WRH#/WRL#) are on their inactive (high levels).

Documentation Update

(reference: C161PI User's Manual V1.0 1999-08)

Structure of P3.0, P3.1

Figure 7-13 on page 7-22 does not exactly represent the structure of P3.0 and P3.1 with respect to the alternate functions SCL0 and SDA0. These pins have the same structure as shown for Port 6 in Figure 7-20 on page 7-32: the alternate output function and direction is controlled via the pin selection bits in register ICCFG.

History List (since device step AA of C161RI)

Functional Problem	Short Description	Fixed in step
ADC.11	Modifications of ADM field while bit ADST = 0 (C161PI only)	
I2C.1	Bit AL in register ICST	BB/ C161RI
PRT.3	Special Thresholds for Port 2 pins and P3.0, P3.1 (C161PI only)	
BUS.17	Spikes on CS# Lines after access with RDCS# and/or WRCS#	AA/ C161PI
BUS.18	PEC Transfers after JMPR Instruction	AA/ C161PI
X9	Read Access to XPERs in Visible Mode	
X12	P0H spikes after XPER write access and external 8-bit Non-multiplexed bus	AA/ C161PI
CPU.17	Arithmetic Overflow by DIVLU instruction	AA/ C161PI
CPU.21	BFLDL/BFLDH Instructions after Write Operation to internal IRAM	
CPU.22	Z Flag after PUSH and PCALL	
PWRDN.1	Execution of PWRDN Instruction while pin NMI# = high	
POWER.9	ASC0 and SSC Output Latches disabled when PCDDIS = 1	AA/ C161PI
POWER.14	Wake Up from Sleep Mode not possible in PLL Mode	
POWER.15	Wake Up from Sleep Mode not possible in Prescaler Mode	
POWER.16	SDD and Sleep Mode cannot be terminated	see also RST18.4
POWER.17	Sleep Mode with RTC on cannot be terminated	
RST.4	Power On Reset (not in AA-step of C161RI)	BB/ C161RI
RST.8	Clock failure detection during external Reset	
RST.9.3	Software/Watchdog Reset during Slow Down Operation with PLL off	
RST.13	Power Up with missing clock	see also RST18.4
RST18.4	Undefined Clock Configuration after Power-On	
ADP.1	Oscillator in Adapt Mode	AA/ C161PI

AC/DC Deviation	Short Description	Fixed in step
AC.t34.n2	CLKOUT rising edge to ALE falling edge –2ns (min, standard supply voltage) - C161PI only	
AC.t34.n10	CLKOUT rising edge to ALE falling edge –10ns (min, reduced supply voltage) - C161PI only	
AC.t33.10	CLKOUT fall time 10ns (max, standard and reduced supply voltage) - C161PI only	
DC.VDDRM.1	Reduced digital supply voltage (minimum) 3.15V - C161PI devices with date code ≥ 0141 only	

Functional Improvements/Compatibility Aspects

The C161PI as the successor of the C161RI provides various technology-based and functional enhancements:

- Increased operating frequency range
- Reduced power consumption
- Optimized 3 Volt performance
- Clock generation also via on-chip PLL
- Enhanced Power Management with
  - Sleep mode (wakeup via external interrupts)
  - programmable frequency output signal
- Port output drivers with two selectable edge characteristics
- A/D converter with 10-bit resolution and auto scan/continuous conversion modes

These enhancements are listed in the **C161PI Specification Update 1998-12** and are included in the **C161PI User's Manual V1.0 1999-08**.

Most of the changes and improvements are activated only under software control and therefore will be transparent for existing C161PI applications. The following items should be considered in particular when replacing the C161RI with the C161PI in an existing application:

A/D Converter

The **conversion result** of the 10-bit A/D converter of the C161PI is always 10 bits wide and stored in bitfield ADRES/ADDAT.[9:0]. There is no result positioning option (as it is controlled via bit ADRP/ADCON.6 in the C161RI), **therefore bit ADCON.6 is specified as 'reserved' in the C161PI, i.e. it must not be set to 1.** For the AA-step of the C161PI, it is allowed that bit ADCON.6 = 1 after execution of instruction EINIT, but not before. The effects on software which was originally written for the C161RI are as follows:

- when bit ADCON.6 = 0 (default after reset): the 2 LSBs read from ADDAT.[9:0] may be different from 00b, while on the C161RI, they were always 00b.
- when bit ADCON.6 = 1: the MSB of the result is at position ADDAT.9 in the C161PI instead of ADDAT.7 as in the C161RI, therefore it is no longer possible to directly read the 8 MSBs of the result from ADDAT with a single byte instruction.

The **conversion time** of the 10-bit A/D converter is slightly longer than the conversion time of the 8-bit converter (at the same internal CPU clock frequency fcpu), and also the encoding of bit field ADCTC/ADCON.[15:14] is different, e.g.

	C161PI [10-bit]				**C161RI [8-bit]**			
f_{CPU} [MHz]	ADCTC	ADC basic clock f_{BC} [MHz]	sample time t_S [µs]	conversion time t_C [1)] [µs]	ADCTC	ADC basic clock f_{BC} [MHz]	sample time t_S [µs]	conversion time t_C [µs]
25	00b	6.25	1.28	7.8	- [2)]	-	-	-
16	00b	4.0	2.0	12.125	01b	4.0	1.5	7.625
12	00b	3.0	2.66	16.166	01b	3.0	2.0	10.166
12	01b	6.0	1.33	8.083	00b	- [3)]	-	-

notes: 1) sample time on C161PI is programmable, calculation here with shortest option (ADCON.[13:12] = 00b, $t_S = 8\ t_{BC}$)
2) operation at 25 MHz f_{CPU} not specified for C161RI
3) setting would result in 6 MHz ADC basic clock f_{BC}, not specified for C161RI

Clock System

The **oscillator circuit** which is implemented in the C161PI and the C161RI is optimized for low power consumption. Its recommended input frequency range for operation with a crystal is **4** .. **16 MHz**. In order to obtain internal CPU frequencies beyond this range, the PLL or an external oscillator should be used. The internal oscillator circuit is compatible to Type LP_2 as described in the Application Note AP2420 ('Crystal Oscillator of the C500 and C166 Microcontroller Families'). When a crystal is used, it is advised to follow the recommendations for the size of the external capacitors of the oscillator circuit, and to check the safety factor of the oscillator circuit.

The Application Note with detailed information on all oscillator types is available on the Infineon Microcontroller Application Note page on

http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8137

Reset Source Indication Flags in register WDTCON

The following table summarizes the behaviour of the reset source indication flags. It applies to both C161RI and C161PI:

Flag Event	LHWR WDTCON.4	SHWR WDTCON.3	SWR WDTCON.2	WDTR WDTCON.1
Long HW Reset	1	1	1	0
Short HW Reset	- / *	1	1	0
SRST instruction	- / *	- / *	1	-
WDT Reset	- / *	- / *	1	1
EINIT instruction	0	0	0	-
SRVWDT instruction	-	-	-	0

Legend: 1 = flag is set, 0 = flag is cleared, - = flag is not affected,
* = flag is set when bi-directional reset option is enabled

Identifier Register IDCHIP

The contents of register IDCHIP (address F07Ch) on the C161PI is **13**XXh (C161RI: 09XXh).

For step **AA** of the C161**PI**, it is **1301h** or **1302h**

For step **BA-H** of the C161**PI**, it is **1303h**

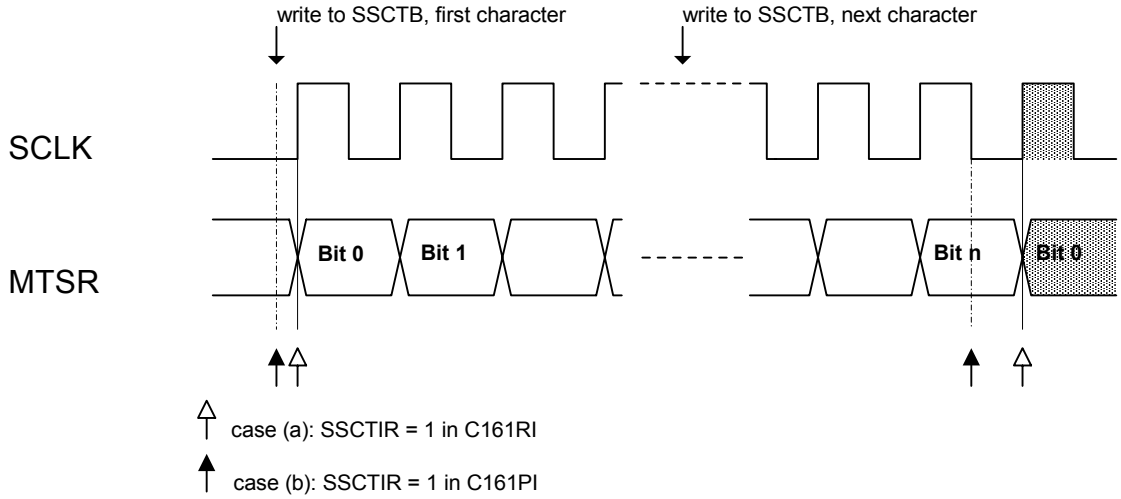
Timing of flag SSCTIR (SSC Transmit Interrupt Request)

In master mode, the timing of SSCTIR depends on the device type as follows:

(a) in the **C161RI**, flag SSCTIR is set to '1' synchronous to the shift clock SCLK 1/2 bit time before the first latching edge (= first shifting clock edge when SSCPH = 0). When SSCTB is written while the shift register is empty, the maximum delay between the time SSCTB has been written and flag SSCTIR=1 is up to 1/2 bit time.

(b) in the **C161PI**, when SSCTB has been written while the transmit shift register was empty (and the SSC is enabled), flag SSCTIR is set to '1' directly after completion of the write operation, independent of the selected baud rate. When the transmit shift register is not empty when SSCTB was written, SSCTIR is set to '1' after the last latching edge of SCLK (= 1/2 bit time before the first shifting edge of the next character). See also e.g. C167CR User's Manual V3.1, p. 12-5.

The following diagram shows these relations in an example for a data transfer in master mode with SSCPO = 0 and SSCPH = 0. It is assumed that the transmit shift register is empty at the time the first character is written to SSCTB:



Typically, in interrupt driven systems, no problems are expected from the modified timing of flag SSCTIR. However, when flag SSCTIR is polled by software in combination with other flags which are set/cleared at the end or at the beginning of a transfer (e.g. SSCBSY), the modified timing may have an effect.

Another situation where a different system behaviour may be noticed is the case when only one character is transferred by the PEC into the transmit buffer register SSCTB. In this case, 2 interrupt requests from SSCTIR are expected: the 'PEC COUNT = 0' interrupt, and the 'SSCTB empty' interrupt.

- in the **C161PI**, the second interrupt request ('SSCTB empty') is always **systematically** generated before the first one ('PEC COUNT = 0') has been acknowledged by the CPU, such that effectively only **one interrupt request** is generated for two different events.
- in the **C161RI**, when the PEC transfer is performed with sufficient margin to the next clock tick from the SSC baud rate generator, and no higher priority interrupt request has occurred in the meantime, the 'PEC COUNT = 0' interrupt will be acknowledged before the 'SSCTB empty' interrupt request is generated, i.e. **two interrupts** will occur based on these events. However, when the PEC transfer takes place relatively close before the next clock tick from the SSC baud rate generator, or a higher priority interrupt request has occurred while the PEC transfer is performed, the 'PEC COUNT = 0' interrupt may not be acknowledged before the 'SSCTB empty' interrupt request is generated, such that effectively only **one interrupt request** will be generated for two different events.

In order to achieve a defined and systematic behavior with all device steps of the C161PI/RI, the SSC receive interrupt, which is generated at the end of a character transmission, may be used instead of the SSC transmit interrupt.

Structure of P3.0/SCL0, P3.1/SDA0

The structure of P3.0 and P3.1 with the alternate functions SCL0 and SDA0 has been modified in the C161PI compared to the C161RI. P3.0/P3.1 in the C161PI now have the same structure as P6.5..7, both in the C161PI as well as in the C161RI (see C161PI User's Manual V1.0 1999-08, Figure 7-20 on page 7-32): the alternate output function and direction is directly enabled via the pin selection bits in register ICCFG, the alternate output signal is no longer ANDed with the contents of the port output latch. As a consequence, in order to modify P3.0/P3.1 in (open drain) output mode by software, the corresponding pin selection bits in register ICCFG must be set to '0'.