Errata Sheet

V 1.0, 2004-10-25

Device	**SAF-C161JI-32RF, SAK-C161JI-32RF, SAF-C161JC-32RF, SAK-C161JC-32RF, SAF-C161CS-32RF, SAK-C161CS-32RF**
Marking/Step	**Step (E)ES-CA, CA**
Package	**P-TQFP-128-2**

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.

Current Documentation

- C161CS/JC/JI-32R/-L Data Sheet, V3.0, Jan. 2001
- C161CS/JC/JI-32R/-L User's Manual, V3.0, Feb. 2001
- Instruction Set Manual: User's Manual, V2.0, Mar. 2001

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.

Contents

Section . . . . . Page
History List/Change Summary 2
Functional Problems 5
Deviations from Electrical- and Timing Specification 30
Application Hints 31
Documentation Update 38

1 History List/Change Summary

(from step AA, previous errata sheet: V1.3, step: AB)

Table 1 Functional Deviations

Functional Problem	Short Description	Fixed in step	Change
ADC.11	Modifications of ADM field while bit ADST = 0		
CAN.7	Unexpected Remote Frame Transmission		
CAN.9	Contents of Message Objects and Mask of Last Message Registers after Reset		
SDLM.3	IFR type2 byte retransmission		
SDLM.4	Arbitration loss at byte boundary		
SDLM.5	Arbitration loss during IFR frames will automatically retransmit IFR data		
SDLM.6	ARL status flag is not cleared after a successful message transmission		
SDLM.7	Automatic IFR Transmission after losing Message Arbitration (3 byte Header)		
SDLM.8	Reading the Bus Side Buffer with FIFO Mode Enabled		
SDLM.9	IFRs cannot be used in 4x Mode		
SDLM.10	TxRQ set during Start of Frame		
SDLM.11	Reset TxRQ - TxCPU		
SDLM.12	Header interrupt blocks other SDLM interrupts		
SDLM.13	Byte Access on Receive Buffer in Block Mode		
CAPCOM.4	SW Access to P1H Overwrites CAPCOM HW Settings		
RTC.6	RTC operation with Auxiliary Oscillator		
TCOMP.3	Pad Driver Temperature Compensation disabled		
BUS.19	Unlatched Chip Selects at Entry into Hold Mode		
X9	Read Access to XPERs in Visible Mode		
X19	Write Access to ASC1 or IIC Module		
CPU.21	BFLDL/BFLDH Instructions after Write Operation to internal IRAM		
CPU.22	Z Flag after PUSH and PCALL		

Table 1 Functional Deviations (cont'd)

Functional Problem	Short Description	Fixed in step	Change
PLL.3.1	Increased PLL Jitter caused by external Access		
PWRDN.1	Execution of PWRDN instruction while Pin $\overline{\text{NMI}}$ = high		
POWER.14	Wake Up from Sleep Mode not possible in PLL Mode		
POWER.15	Sleep Mode not possible in Prescaler Mode		
POWER.17	Sleep Mode with RTC on cannot be terminated		
POWER.19	SW- or WDT-Reset while SDD Mode is active	see RST.9	
RST.8	Clock failure detection during external Reset		
RST.9	Software/Watchdog/Short HW Reset during Slow Down operation with PLL off		
RST.17	Missing Clock at XTAL3 and SDD Mode		
RST.18.3	Undefined Clock Configuration after Power-On		

Table 2 AC/DC Deviations

AC/DC Deviations	Short Description	Fixed in step	Change
DC.HYS.250	Input Hysteresis(Special Threshold) 250 mV		
DC.VILS.1	Input low voltage (Special Threshold) 1.7 V		

Table 3 Application Hints

Hint	Short Description	Change
SSC.H1	Handling of the SSC Busy Flag (SSCBSY)	
SSC.H2	Timing of flag SSCTIR (SSC Transmit Interrupt Request)	
CAN.H1	Note on Interrupt Register behaviour of the CAN module	
IIC.H1	Maximum IIC Bus Data Rate at low f_{CPU}	
IIC.H2	Overload Protection Circuit for IIC Bus Pins P9.0 ... P9.4	
BSL.H1	Bootstrap Loader: Baudrate Detection in Single Chip Boot Mode	
PLL.H1	PLL holds Reset Mode while missing clock	
AuxOsc.H1	Rf for Type_RTC2 Auxiliary Oscillator necessary	
MainOsc.H1	Oscillator Type_LP2: Negative Resistance and Start-up Reliability	
MainOsc.H2	Maximum (Type_LP2) Oscillator Frequency = 16 MHz	
OWD.H1	Oscillator Watchdog and Auxiliary Oscillator	
OWD.H2	Oscillator Watchdog and Prescaler Mode	
EMUL.H1	Adapt Mode Setting	
ISNC.H1	Maintenance of ISNC Register	

Table 4 Documentation Update

Name	Short Description	Change
SDLM.D1	Register FLAGRST returns a random value at read access	new
IIC.D1	Output Pin Configuration	new
SPEC.D1	Values verified by Characterization	
PORT.D3	P3.15 and Frequency Output Signal Path	
ID-Registers	ID-Registers	new

2 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.

When no interrupt or PEC is servicing the A/D Conversion Complete Interrupt, interrupt request flag ADCIR will be set, and for n > 1 also the A/D Overrun Error interrupt request flag will be set, unless the wait for ADDAT read mode had been selected. When ADCON.ADWR = 1 (wait for ADDAT read), the converter will wait after 2 conversions until ADDAT is read.

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. C164CI User's Manual, V1.0, p.18-4)

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.

CAN.7 Unexpected Remote Frame Transmission

Symptom:

The on-chip CAN module may send an unexpected remote frame with the identifier = 0, when a pending transmit request of a message object is disabled by software.

Detailed Description:

There are three possibilities to disable a pending transmit request of a message object (n = 1..14):

- Set CPUUPDn element
- Reset TXRQn element
- Reset MSGVALn element

Either of these actions will prevent further transmissions of message object n.

The symptom described above occurs when the CPU accesses CPUUPD, TXRQ or MSGVAL, while the pending transmit request of the corresponding message object is transferred to the CAN state machine (just before start of frame transmission). At this particular time the transmit request is transferred to the CAN state machine before the CPU prevents transmission. In this case the transmit request is still accepted from the CAN state machine. However the transfer of the identifier, the data length code and the data of the corresponding message object is prevented. Then the pre-charge values of the internal "hidden buffer" are transmitted instead, this causes to a remote frame transmission with identifier = 0 (11 bit) and data length code = 0.

This behavior occurs only when the transmit request of message object n is pending and the transmit requests of other message objects are not active (single transmit request).

If this remote frame loses arbitration (to a data frame with identifier = 0) or if it is disturbed by an error frame, it is not retransmitted.

Effects to other CAN nodes in the network:

The effect leads to delays of other pending messages in the CAN network due to the high priority of the Remote Frame. Furthermore the unexpected remote frame can trigger other data frames depending on the CAN node's configuration.

Workarounds:

- The behavior can be avoided if a message object is not updated by software when a transmission of the corresponding message object is pending (TXRQ element is set) and the CAN module is active (INIT = 0). If a re-transmission of a message (e.g. after lost arbitration or after the occurrence of an error frame) needs to be cancelled, the TXRQ element should be cleared by software as soon as NEWDAT is reset from the CAN module.

- The nodes in the CAN system ignore the remote frame with the identifier = 0 and no data frame is triggered by this remote frame.

CAN.9 Contents of Message Objects and Mask of Last Message Registers after Reset

After any reset, the contents of the CAN Message Objects and the Mask of Last Message Registers may be undefined instead of unaffected (reset value 'X' instead of 'U').

This problem depends on temperature and the length of the reset, and differs from device to device. The problem is more likely to occur at high temperature and for long hardware resets (> 100 ms).

Workaround:

Re-initialize the CAN module after each reset.

SDLM.3 IFR type2 byte retransmission

The SDLM has transmitted the IFR type2 byte successfully. After the reception of an IFR byte of another node, the SDLM will retransmit its own IFR byte.

The behavior is taken into account if ARIFR bit (Automatic Retry of IFR) is set.

Workaround:

- The problem can be avoided if the SDLM IFR type2 byte value has the lowest priority (largest number) in the network. Note that this prohibits having more than one SDLM node on the bus.

SDLM.4 Arbitration loss at byte boundary

The SDLM transmits two passive bits when it loses arbitration at the last bit of a transmitted byte (any byte boundary). This leads to arbitration problems, if the application is not conforming to class 2 specification.

Workaround:

none.

SDLM.5 Arbitration loss during IFR frames will automatically retransmit IFR data

The problem occurs when the SDLM loses arbitration during the transmission of its IFR data. The SDLM will retry the transmission of its IFR data regardless of the state of the GLOBCON.ARIFR (Automatic Retry of IFR) control bit.

Workaround:

This only concerns the transmission of type1 or type3 IFRs. During the transmission of type1 or type3 IFRs use the arbitration loss interrupt to clear the TXIFR flag, aborting IFR transmission.

SDLM.6 ARL status flag is not cleared after a successful message transmission

The TRANSSTAT.ARL flag is not cleared after the SDLM module has successfully transmitted a frame or IFR.

Workaround:

Software should clear the TRANSSTAT.ARL flag after every occurrence of an arbitration loss.

SDLM.7 Automatic IFR Transmission after losing Message Arbitration (3 byte Header)

While transmitting a 3 byte consolidated header message and the SDLM has lost arbitration to another node that is requesting a type2 IFR, the SDLM will not automatically send its IFR data.

Workaround:

Do not use automatic IFR transmission. Use the header interrupt to determine the type of IFR and control the IFR transmission.

SDLM.8 Reading the Bus Side Buffer with FIFO Mode Enabled

Read of byte 1 of the bus side receive buffer using a word access is not possible if FIFO mode is enabled. If the FIFO mode is enabled for the receive buffer (block mode or RXINCE = '1') the second byte of the bus side receive buffer (address $xx51_H$) can't be read with a word access. The byte is always read as 00_H.

Workaround:

Use byte access to read address $xx51_H$.

SDLM.9 IFRs cannot be used in 4x Mode

IFRs are not possible in 4x diagnostic mode due to internal state machine constraints.

Workaround:

Do not use IFRs while in 4x Mode.

SDLM.10 TXRQ set during Start of Frame

In case a message is on the bus with Start of Frame and a Transmission Request is set for the SDLM module, then both messages get lost. This means, that the MSGREC interrupt is not generated.

Workaround:

Do not set TXRQ in case in register BUSSTAT the bits IDLE and SOF are low (SOF has to be reset by software) or IDLE and RIP (in register BUFFSTAT – the bit is reset by hardware) are low.

SDLM.11 Reset TXRQ - TxCPU

A software reset on bit TXRQ in register BUFFCON should reset the register TXCPU. This is the case, if blockmode (bit BMEN, register GLOBCON) is enabled.

Failure: In all other cases TXCPU is not reset, if TXRQ is reset by software.

Workaround:

Reset TXCPU by software.

SDLM.12 Header interrupt blocks other SDLM interrupts

The header interrupt keeps the interrupt line until an End Of Frame has been seen. After EOF and before the Interframe Separation Time, the SDLM interrupt node is released.

Workaround:

Inside the header interrupt, the header interrupt itself has to be disabled. The easiest way to reenable this interrupt is to use the EOF-Interrupt.

SDLM.13 Byte Access on Receive Buffer in Block Mode

According to the manual, an access to the receive buffer in block works like the following: RxD00 is accessed as long as RxCPU is smaller RxCNT. Automatically bytes are shifted to RxD00, so that the complete buffer is emptied.

Failure:

Accessing RxD00 gives two times the same byte and then the next but one byte.

Workaround:

To read out the complete sequence received, the following algorithm can be used:

```
// check if receive buffer is valid
 if ((!(SDLM_RIP)) && SDLM_RXCNT)
 {    // get number of received bytes
      ubLen = RXCNT_RXCNT;
      SDLM_RXCNTB=0;
       while ((SDLM_RXCPU=RXCPU_RXCPU) < (SDLM_RXCNT=RXCNT_RXCNT))
       { if (!(SDLM_RXCNTB%2)) // odd or even
          { *(ubBuffer++) = RXD00_RXDATA00; // normal access }
          else
           { *(ubBuffer++) = RXD00_RXDATA01; // access odd byte information
              buffer_inbetween = RXD00_RXDATA00; // increment RxCPU }
            SDLM_RXCNTB++;
       }
     // release receive buffer
     BUFFCON |= BUFFCON_DONE;
```

CAPCOM.4 SW Access to P1H Overwrites CAPCOM HW Settings

HW settings on P1H.7...4 by CAPCOM compare output functions (CC24 ... CC27) can be overwritten by SW accesses to P1H.7...0 on the same port.

Read modify write operations like BSET, BFLDx, OR, ... read the input or output latches respectively, modify the affected bits and write back the result to the output latches of the whole port (P1H7...0).

In case a compare event has occurred after the read phase, but before the write-back phase of such an instruction, the output signal change of the compare event is lost or

only a short pulse (>/=1 TCL) may appear. The bit protection mechanism to avoid these effects is out of function on P1H.

Workaround:

- Avoid the combination of HW and SW write accesses to P1H. HW access only to CC24 ... CC27 or SW access only to P1H.7 ... 0 works properly.
- Use "interrupt only" compare modes (CCMODx = 100 or 110) and modify the port pin in the interrupt service routine by software to avoid the combination of HW and SW accesses.

RTC.6 RTC operation with Auxiliary Oscillator

When the auxiliary oscillator (XTAL3/4) is selected as the clock source for the RTC (bit RCS/SYSCON2.6 = 1), it is not possible to switch back the RTC to main oscillator.

Workaround:

Do not change the selection of the clock source for the RTC after 1st setting after power on reset.

TCOMP.3 Pad Driver Temperature Compensation disabled

To avoid a malfunction and additional current consumption the on chip pad driver temperature compensation is disabled by default.

The description of temperature compensation has been removed from documentation.

BUS.19 Unlatched Chip Selects at Entry into Hold Mode

Unlike in standard (latched) configuration, the chip select lines in unlatched configuration (SYSCON.CSCFG = 1) are not driven high for 1 TCL after HLDA# is driven low, but start to float when HLDA# is driven low.

Workaround:

none

X9 Read Access to XPERs in Visible Mode

The data of a read access to an XBUS-Peripheral (CAN) in Visible Mode (SYSCON.1 = 1) 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.

X19 Write Access to ASC1 or IIC Module

When a **write** access to one of the registers of the **ASC1** module is directly followed by a **read** access to any register of the **ASC1 or IIC** module, incorrect data will be read.

When a **write** access to one of the registers of the **ASC1 or IIC** module is directly followed by a write access to any register of the **ASC1** module, incorrect data will be written (destination register of first write access may be overwritten with data of second write access, destination register of second write access may not be modified).

When a write access to one of the registers of the **IIC** module is directly followed by a **read** access to any register of the **ASC1** module, incorrect data will be read.

Workaround:

- for uninterruptible instruction sequences (e.g. during the initialization phase where interrupts are disabled) make sure that after a write access to the ASC1 or IIC module the following 3 instructions do not access the ASC1 or IIC module
- otherwise, insert an ATOMIC #4 instruction in front of any instruction which writes to the ASC1 or IIC module, and make sure that the following 3 instructions do not access the ASC1 or IIC module
- in case a PEC channel which writes to the ASC1 or IIC module is used, no other PEC transfers which read from or write to the ASC1 or IIC module must be used

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 $00'F200_H$ ... $00'FDFF_H$ for 3 Kbyte module, $00'F600_H$ ... $00'FDFF_H$ for 2 Kbyte module, or $00'FA00_H$ ... $00'FDFF_H$ 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 = $0F\mathbf{y}_H$ (**y** = $0..F_H$),
      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 = $\mathbf{00_H}$ **...** $\mathbf{0EF_H}$,
      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 $\mathbf{00_H} \leq \mathbf{v}_L \leq \mathbf{7F_H}$
      and the previous instruction or PEC transfer writes to the (low and/or high byte of) the specific bit-addressable IRAM location $\mathbf{00'FD00_H + 2\ v}_L$
      (i.e. the 8-bit offset address of the location in the bit-addressable IRAM area ($00'FD00_H$ ... $00'FDFF_H$) 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 $00'FCE0_H$ ... $00'FCFE_H$ 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 = $0Fy_H$ (**y** = $0..F_H$), and the range which is covered by the context pointer CP (partially or completely) overlaps the PEC source and destination pointer area ($00'FCE0_H$ ... $00'FCFE_H$), 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 ($00'FD00_H$ ... $00'FDFF_H$), 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 $\leq 20_H$).

Workaround 1:

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

- 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.
- 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

Table 5 Range of original ATOMIC/EXTEND statement

	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 do not 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: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8984**

AP1628 Description: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=18484&parent_oid=-8984**
AP1628 Software: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=18468&parent_oid=-8984**

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 00_H, and
- the contents of the low byte of the GPR or (E)SFR which is pushed is > 00_H, 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: F1 EC):
           ;  incorrect setting of Z flag if contents of R15 is odd,
           ;  and 00FFh ≥ contents of R1 ≥ 0001h
PUSH  DPP3 ;(coding: 03 EC):
           ;  incorrect setting of Z flag if contents of R0 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 SFR which is pushed is 00_H, and
- when the contents of the low byte of the GPR or 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: 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      ; 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
```

```
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:

www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8984

direct links:

AP1679 Description: **www.infineon.com/cmc_upload/documents/040/841/ap1679_v1.1_2002_05_scanning_cpu22.pdf** and
AP1679 Software: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=40840&parent_oid=-8984**

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:
  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:
  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 = 0000_H 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

PLL.3.1 Increased PLL Jitter caused by external Access

Problem description:

In systems where the PLL is used for generation of the CPU clock frequency, the PLL jitter can increase under certain circumstances and exceed the specified value.

The value of the additional jitter is not a fixed one but it depends on the kind of activity on the external bus or output pins. The additional jitter increases with the number of output/bus pins which are switched at the same time to a new voltage level because the problem is caused by noise on the on-chip power supply.

The capacitive load of the used pins has also an influence to the additional jitter. A high capacitive load can increase the additional jitter.

Effects to the system:

All PLL factors are affected (PLL factor 1.5 / 2 / 2.5 / 3 / 4 / 5). The PLL jitter increases when an access to the external bus or output pins is performed. This phenomenon has no influence to direct drive-, prescaler-, and SDD mode.

The problem does not affect the functionality of the CPU and the on-chip peripherals.

The additional jitter has the maximum effect if only one TCL is considered (period jitter). This can have an influence to the bus timings with one TCL.

The additional jitter decreases with the number of consecutive TCLs (accumulated jitter). The accumulated jitter has an effect on timings with more than one TCL (certain bus timings, CAN bus timing, serial interfaces).

New value of the additional jitter

For an accumulated period of N*TCL the new maximum jitter D_N[ns], which exceeds the specified value in the Data Sheet, is computed using the formula:

$$D_N[ns] = \pm[(50 / f_{CPU} + 164 / f_{CPU}^2 - 0.04) * (N / 2 - 1) + 1180 / f_{CPU}^2 + 0.2]$$

Were N = number of consecutive TCL, f_{CPU} = CPU frequency in MHz.
This approximated formula is valid for $1 \leq N \leq 12$ and $10\ MHz \leq f_{CPU} \leq 25\ MHz$.

For all accumulated periods longer than 12 TCL the N=12 value can be used; see figure below.

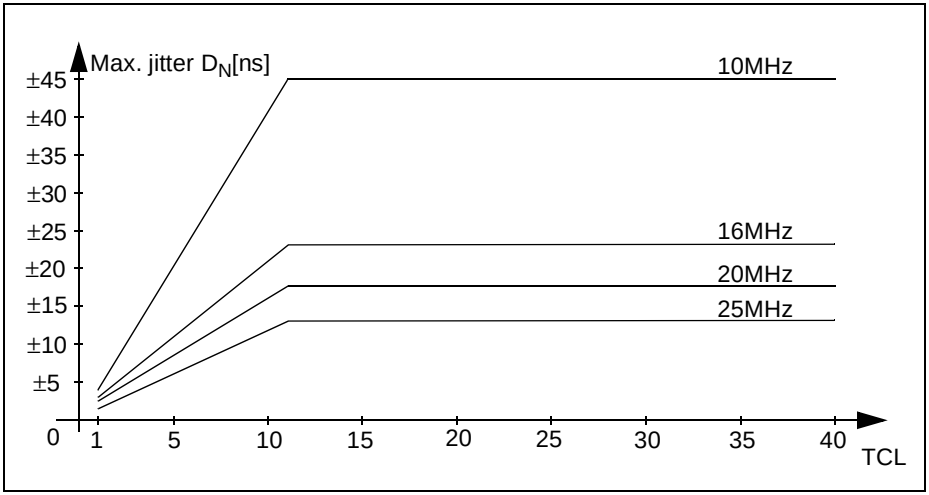


Figure 1 Additional Jitter

Workaround:

In case that the additional jitter causes problems in the system which cannot be relaxed with software adapted timing parameters, then direct drive or prescaler mode can be used. If the maximum frequency of the on-chip oscillator (16 MHz) is not sufficient then an external clock generator can be used.

PWRDN.1 Execution of PWRDN Instruction while pin $\overline{\text{NMI}}$ = high

When instruction PWRDN is executed while pin $\overline{\text{NMI}}$ 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 preceeding the PWRDN instruction writes to external memory or an XPeripheral (e.g. CAN or XRAM), 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 $\overline{NMI}$ 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 preceeds the PWRDN instruction, otherwise insert e.g. a NOP instruction in front of PWRDN. When a muliplexed bus with memory tristate waitstate is used, the PWRDN instruction should be executed out of internal RAM.

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

Sleep mode cannot be terminated by external interrupt (NMI or fast external interrupts including altermate 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 only: Switch to direct drive mode before selecting sleep mode.

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 only: Switch to direct drive mode before selecting sleep mode (but take care on max. f_{CPU})

POWER.17 Sleep Mode with RTC on cannot be terminated

After the wake up signal from sleep mode is recognised, 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 but the flash module (with approximately 20 mA current consumption) is already enabled.

Workarounds:

- Switch to direct drive mode before entering sleep mode (bitfield CLKCFG = 011b in register RSTCON), see User's Manual, 22.5 System Configuration via Software.
- Select sleep mode with RTC off.

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:

- 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.

- 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.

RST.9 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 SYSCON.2) then depending on the status of the main oscillator a critical and an uncritical behavior can occur:

Uncritical behavior (Main oscillator is switched off)

If the main oscillator is **switched off** (devices with Auxiliary Oscillator only: Bits RCS and SCS = 1 in register SYSCON2) then the internal reset condition is extended until the PLL is properly locked. The system starts after the oscillator supplies a clock at XTAL1 and the PLL is locked.

Critical behavior (Main oscillator is switched on)

If the main oscillator **remains active** (Bit RCS or SCS = 0 in register SYSCON2) then the internal reset condition is extended until the next long HW reset.
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:

- Switch **off** the main oscillator during slow down mode while PLL is switched off (select CLKCON = 10 in register SYSCON2 **and** auxiliary oscillator as clock source for RTC **and** auxiliary oscillator as clock source for SDD, i.e. set bits RCS = 1 **and** SCS = 1). Using the auxiliary oscillator as a clock source requires a good blocking of the power supply pins to reduce noise on V_{DD} and V_{SS}.
- 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.

RST.17 Missing Clock at XTAL3 and SDD Mode

The bits SCS and RCS in register SYSCON2 are described as not affected by a reset - consequently they are undefined after power on.

To change the configuration of SCS and RCS in register SYSCON2 a clock pulse from XTAL3 is required.

Under certain circumstances (e.g. after short voltage drops) in some of the devices the internal setting of SCS and RCS in register SYSCON2 can be changed.

After such an erroneous setting even SW access to these bits does not change the internal configuration as long as no clock is provided at XTAL3. This will cause the following problems:

- the CPU clock will be stopped when SDD mode is entered (internal setting of SCS = 1)
- the RTC will be stopped (internal setting of RCS = 1)

Workarounds:

- Using the auxiliary oscillator with an external quartz crystal.

- Connection between XTAL2 (output main osc.) and XTAL3 (input auxiliary osc.) is recommended to activate the auxiliary oscillator (in this case with an undefined frequency).
- Switch power off for a duration that Vcc and $\overline{\text{RSTIN}}$ reaches nearly 0 V at the microcontroller. Now supply voltage has to be switched on again.

RST.18.3 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 to set the clock configuration in register SYSCON2 (SCS and RCS) to a defined state and to clear the real time clock registers (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}) $\overline{\text{RSTIN}}$ has to be active and is required at least until the clock at XTAL1 is stable. The input level of $\overline{\text{RSTIN}}$ has to be
  $V_{\overline{\text{RSTIN}}} \leq 0.3 * V_{DD}$ for $0\text{ V} \leq V_{DD} < 2.7$ V and
  $V_{\overline{\text{RSTIN}}} \leq 0.8$ 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 bits SCS, RCS in register SYSCON2 and the RTC registers are set to the following values:

- SYSCON2.7 (SCS) is cleared to select main oscillator for slow down divider clock source.
- SYSCON2.6 (RCS) is cleared to select main oscillator for real time clock source.
- 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: Clock @ XTAL1, NO Clock @ XTAL3 and Power-on Detection OK

This is only an application hint for systems with no clock at XTAL3:

- When the system is running in direct drive or PLL clock mode and the software switches the clock path to slow down divider (SDD) mode, it is not allowed to switch the clock source from main oscillator to auxiliary oscillator. Because when bit RSTCON2.SCS is set to ’1’ while no clock is at XTAL3 the system stops.
- If bit RSTCON2.RCS is set to ’1’ then the RTC stops.

Case 2: Clock @ XTAL1, NO Clock @ XTAL3 and Power-on Detection FAIL

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

For the modification of the internal clock multiplexer controlled via SCS or RCS in register SYSCON2 a clock pulse from XTAL3 is required. The register contents SCS and RCS can be modified without a clock pulse from XTAL3 but this has no effect on the system because SCS and RCS are synchronized with XTAL3.

- When the system is running in direct drive or PLL clock mode, it is not allowed to switch the clock path to SDD mode via software because if SCS = 1 (modification via software has no effect on the clock multiplexer) then the system gets no clock.
- When SYSCON2.RCS = 1 the RTC will stop (modification via software has no effect on the clock multiplexer).

Workarounds:

- Using the auxiliary oscillator with an external crystal.
  Despite this workaround the case 3 "Clock @ XTAL1, Clock @ XTAL3 and Power-on Detection FAIL" has to be considered too.
- Connection between XTAL2 (output main osc.) and XTAL3 (input auxiliary osc.) is recommended to activate the auxiliary oscillator (in this case with an undefined frequency).
  Despite this workaround the case 3 "Clock @ XTAL1, Clock @ XTAL3 and Power-on Detection FAIL" has to be considered too.
- Consider the conditions to get a defined clock configuration at power-on

Case 3: Clock @ XTAL1, Clock @ XTAL3 and Power-on Detection FAIL

This is only an application hint:

When the power-on detection fails during power-on because of not considering the necessary conditions, the register bits SCS, RCS and the RTC registers are undefined.

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).
- Bit RCS in register SYSCON2 is undefined (i.e. main or auxiliary oscillator may be selected as real time clock source). If main oscillator is selected (RCS = 0), the RTC stops (NO clock @ XTAL1).
- Bit SCS in register SYSCON2 is undefined (i.e. main or auxiliary oscillator may be selected as slow down divider clock source). If main oscillator is selected (SCS = 0), the slow down divider stops (NO clock @ XTAL1) - see SDD selection below.
- The slow down divider (SDD) may be selected as f_{CPU} clock source (no oscillator watchdog function possible - the oscillator selected in SCS has to be active otherwise the device will stop).

Workaround:

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

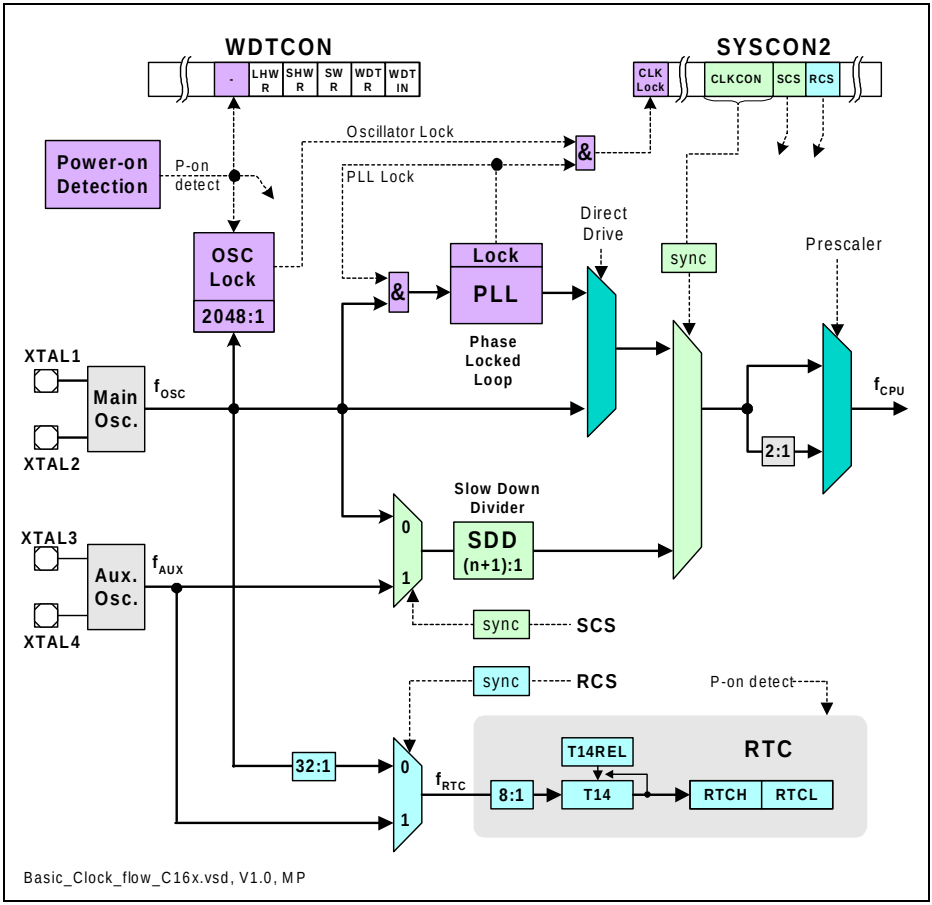


Figure 2 C161JI/JC/CS-32RF Clock Configuration

3 Deviations from Electrical- and Timing Specification

Problem Short Name	Parameter	Symbol	Limit Values		Unit	Test Condition
			min.	max.		
DC.HYS.250	Input Hysteresis (Special Threshold)	HYS	**250** instead of 300	–	mV	–
DC.VILS.1	Input low voltage (Special Threshold)	V_{ILS}	-0.5	**1.7** instead of 2.0	V	–

Note: Pin **$\overline{\text{READY}}$** *has an internal pull-up (all C16x derivatives). This is documented in the Data Sheet now.*

Note: Timing t_{28}: Parameter description and test changed from 'Address hold after $\overline{RD}$/$\overline{WR}$' to 'Address hold after $\overline{WR}$'. It is guaranteed by design that read data are internally latched by the controller before the address changes.

Note: During reset and adapt mode (in external bus mode - pin $\overline{EA}$ = LOW), the ***internal pull-ups on P6.[4:0]*** *are active, independent whether the respective pins are used for $\overline{CS}$ function after reset or not.*

4 Application Hints

SSC.H1 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:

- 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)
```

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

SSC.H2 Timing of flag SSCTIR (SSC Transmit Interrupt Request)

In master mode, the timing of SSCTIR is as follows:

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:

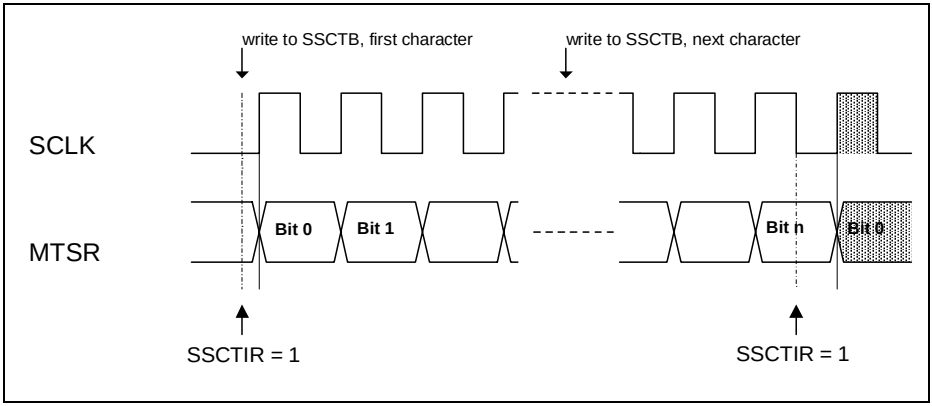


Figure 3 SSCTB Timing

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:

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, the SSC receive interrupt, which is generated at the end of a character transmission, may be used instead of the SSC transmit interrupt.

CAN.H1 Note on Interrupt Register Behaviour of the CAN module

Due to the internal state machine of the CAN module, a specific delay has to be considered between resetting INTPND and reading the updated value of INTID. See Application Note AP2924 "Interrupt Register behaviour of the CAN module ..." on:

Infineon Microcontroller Products: **www.infineon.com/microcontrollers**

AP 2924 V01:

http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/public_download.jsp?oid=10042&parent_oid=-8984

IIC.H1 Maximum IIC Bus Data Rate at low f_{CPU}

The IIC 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.

More details are under evaluation.

IIC.H2 Overload Protection Circuit for IIC Bus Pins P9.0 ... P9.4

Since these pins are used for alternate functions by the IIC Bus module, they have a different internal protection circuit than the other I/O pins in order to avoid interference with the IIC 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).

BSL.H1 Bootstrap Loader: Baudrate Detection in Single Chip Boot Mode

In single chip boot mode (Pin $\overline{EA}$ = 1) the bootstrap loader mode is detected during hardware reset when pin $\overline{RD}$ is tied to low.

In this mode $f_{CPU} = f_{OSC} / 2$ (default configuration) and can be controlled via register RSTCON. In case of a desired PLL mode (e.g. with a PLL factor of 5), f_{CPU} in bootstrap loader mode is 10 times slower than in normal running mode. So a communication with an external host starts with a baudrate related to $f_{CPU} = f_{OSC} / 2$. It is recommended to establish a 2nd level loader which adapts f_{CPU} and baudrate to a convenient transmission rate.

PLL.H1 PLL holds Reset Mode while missing Clock

In case of a missing clock signal at input XTAL1 during reset in PLL modes (reset configuration P0H.7-5 is set to $f_{CPU} = f_{XTAL} * x$; x = 1.5, 2, 2.5, 3, 4 or 5) the PLL circuit holds the device in reset mode instead of providing a PLL base frequency clock signal.

AuxOsc.H1 Rf for Type_RTC2 Auxiliary Oscillator necessary

This auxiliary oscillator-inverter is also a Real Time Clock oscillator with a very low power consumption and it is optimized for a frequency range of 32 kHz ± 50%. The feedback resistor Rf of the Type_RTC2 oscillator is not integrated on chip. Rf has to be connected externally between pin XTAL3 and XTAL4. The recommended values for Rf and the load capacitors are Rf = 6.8 MOhm, CX3 = CX4 = 2.7 pF. The value of Rf should not be smaller than Rf_min = 5 MOhm or start-up problems can occur. Note, that the value of Rf can be reduced by condensed humidity or dirt particles on the PCB .

How to check the Safety Factor is described in ApNote 2420xx:

www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_cat.jsp?oid=-8984

AP2420: www.infineon.com/cgi/ecrm.dll/ecrm/scripts/
public_download.jsp?oid=9746&parent_oid=-8984

MainOsc.H1 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'.
Infineon Microcontroller Products: **www.infineon.com/microcontrollers**

AP2420: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/**
public_download.jsp?oid=9746&parent_oid=-8984
AP2424: **www.infineon.com/cgi/ecrm.dll/ecrm/scripts/**
public_download.jsp?oid=9745&parent_oid=-8984

MainOsc.H2 Maximum Oscillator Frequency = 16 MHz (Main: Type_LP2)

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 (see Data Sheet, AC Characteristics, tables 'Clock Generation Modes' and 'External Clock Drive Characteristics'). Operation at lower input frequencies is possible but is guaranteed by design only (not 100% tested) (see Data Sheet, AC Characteristics, table 'External Clock Drive Characteristics').

OWD.H1 Oscillator Watchdog and Auxiliary Oscillator

If oscillator watchdog is enabled (SYSCON bit OWDDIS = 0) the oscillator frequency is compared with the PLL output frequency. In direct drive or prescaler mode (reset configuration on port 0) a missing oscillator clock leads to the OWD interrupt (SFR ISNC, bit PLLIR = 1) and the free running PLL is used as clock source until next hardware reset.

The main oscillator is automatically switched off in a configuration that uses the auxiliary oscillator only (e.g. SYSCON3 = 01C0h: Slow Down Mode, SDD clock source and RTC clock source is the auxiliary oscillator).

If the main oscillator is switched off and the oscillator watchdog (OWD) is enabled bit PLLIR (ISNC.2) will be set and PLL frequency used after switch back to basic clock.

Recommendation:

disable OWD (bit OWDDIS (SYSCON.4) = 1) before execution of EINIT.

OWD.H2 Oscillator Watchdog and Prescaler Mode

The OWD replaces a missing oscillator clock signal with the PLL clock signal (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).

EMUL.H1 Adapt Mode setting for emulation of devices in Single Chip Mode

In Adapt Mode all target device pins are in high impedance state. This mode is mainly used for deactivation of the soldered device while an emulator probe is connected. In single chip mode ($\overline{EA}$ = 1) Adapt Mode can not be selected via PORT0 (P0L.1) during reset, because PORT0 is not evaluated in this mode.

Recommendation:

- use no target device or a dummy device (package without silicon on the PCB) for emuation **or**
- use the target device on the PCB **and**:
  - provide a possibility to set target device pins $\overline{EA}$ and P0L.1 to "0" during reset (now the adapt mode is enabled on the target device) **and**
  - disconnect pin $\overline{EA}$ from the emulator probe **and**
  - set the $\overline{EA}$ pin on the emulator-probe to "1" to configure the emulator in single chip mode.

ISNC.H1 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 same source it is recommended to clear the request and the enable bit first and then to set the enable bit again.

Example for an 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
}
...
```

Example for an XP3 interrupt service routine (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:
...
```

5 Documentation Update

SDLM.D1 Register FLAGRST returns a random value at read access

C161CS/JC/JI-32F / -16F User's Manual, V2.0 Jan. 2001 and C161CS/JC/JI-32R/-L User's Manual, V3.0, Feb. 2001, Chapter 20 - The Serial Data Link Module - SDLM Register Description (page 20-38) states in remark 1 for FLAGRST: All bits return '0' when read.

This sentence will be removed.

FLAGRST: This register is for write purposes only. In case this register is accessed either by read-modify-write-functions or by direct reads, it will return a random value.

IIC.D1 IIC - Output Pin Configuration

The following sentences in chapter "The IIC Bus Module", paragraph "Output Pin Configuration":

"All pins of the C161JI/JC/CS-32RF that are to be used for IIC bus communication must be switched to output and their alternate function must be enabled (by setting the respective port output latch to '1'), before any communication can be established.

If not driven by the IIC module (i.e. the corresponding enable bit in register ICCFG is '0') they then switch off their drivers (i.e. driving '1' to an open drain output). Due to the external pullup devices the respective bus levels will then be '1' which is idle."

have to be replaced by:

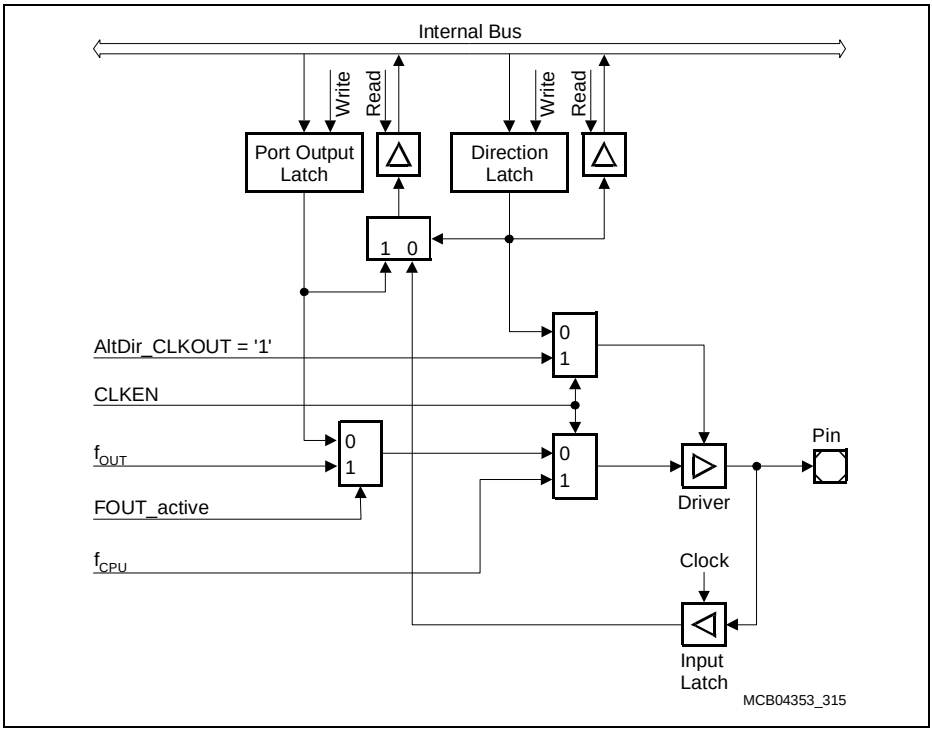
All IIC-pins of the C161JI/JC/CS-32RF that are not used for IIC bus communication (i.e. the corresponding enable bit in register ICCFG is '0') are then controlled by their port control registers. The pin drivers can be disabled either by switching them to input (default after reset) or by driving '1' to the open drain outputs. Due to the external pullup devices the respective bus levels will then be '1' which is idle.

SPEC.D1 Values verified by Characterization

The values for I_{IDOM}, I_{IDOA} and I_{PDRM} (Data Sheet, V3.0, Jan. 2001 page 53) and all timings of external clock drive XTAL3 (Data Sheet, V3.0, Jan. 2001 page 61) are not 100% tested but verified by means of system characterization.

PORT.D3 P3.15 and Frequency Output Signal Path

To enable the frequency output signal f_{OUT} bit CLKEN in register SYSCON must be cleared (before EINIT) and f_{OUT} must be enabled in register FOCON. In contrast to signal CLKOUT the direction latch has to be set by software to enable f_{OUT}.



ID-Registers

		Register:	IDMANUF	IDCHIP	IDMEM	IDPROG	IDMEM2
Device	**Step**	***Address:***	**$F07F_H$**	**$F07C_H$**	**$F07A_H$**	**$F078_H$**	**$F076_H$**
C161JI-32R	-CA		1820_H	1703_H	1040_H	0000_H	0000_H
C161JC-32R	-CA		1820_H	1503_H	1040_H	0000_H	0000_H
C161CS-32R	-CA		1820_H	1403_H	1040_H	0000_H	0000_H

Product and Test Engineering Group, Munich