Errata Sheet

V2.2, 2000-11-08

Devices	**C164CI-8R, C164SI-8R, C164CL-8R, C164SL-8R, C164xy-L**
Marking/Step	**Step ES-AB, AB**
Package	**P-MQFP-80-1 (C-MQFP-80: EES-AB only)**

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

- C164 Data Sheet 1999-08
- C164 User's Manual V2.0 1999-09
- Instruction Set Manual 12.97 Version 1.2

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

Microcontrollers

1 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.

ADC.13 ADC Channel Injection Trigger

Channel injection mode activated by ADCIN = 1 (and ADWR = 1) in register ADCON can be triggered by setting bit ADCRQ in register ADCON or by capture event on CAPCOM channel 27 or by compare event on CAPCOM channel 27 depending on the selected mode.

The erroneous function does not trigger channel injection by any function of CAPCOM channel 27.

Workaround:

Use an CC27 interrupt service routine to set bit ADCRQ to trigger channel injection.

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.

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.4 Undefined Results after write to T14

A write operation to T14 leads to undefined results depending on contents of T14REL.

Workaround:

Load T14REL (1st) and T14 (2nd) with the desired value.

BUS.17 Spikes on CS# Lines after access with RDCS# and/or WRCS#

Spikes of about 5 ns width (measured at VOH = 0.9 Vcc) from Vcc to Vss may occur on Port 6 lines configured as CS# signals. The spikes occur on one CSx# line at a time for the first external bus access which is performed via a specific BUSCONx/ADDRSELx register pair (x=1..4) or via BUSCON0 (x=0) when the following two conditions are met:

- the previous bus cycle was performed in a non-multiplexed bus mode without tristate waitstate via a different BUSCONy/ADDRSELy register pair (y=1..4, y≠x) or BUSCON0 (y=0, y≠x) **and**
- the previous bus cycle was a read cycle with RDCSy# (bit BUSCONy.CSRENy = 1) or a write cycle with WRCS# (bit BUSCONy.CSWENy = 1).

The position of the spikes is at the beginning of the new bus cycle which is performed via CSx#, synchronous with the rising edge of ALE and synchronous with the rising edge of RD#/WR# of the previous bus cycle.

Potential effects on applications:

- when CS# lines are used as CE# signals for external memories, typically no problems are expected, since the spikes occur after the rising edge of the RD# or WR# signal.
- when CS# lines configured as RDCS# and/or WRCS# are used e.g. as OE# signals for external devices or as clock input for shift registers, problems may occur (temporary bus contention for read cycles, unexpected shift operations, etc.). When CS# lines configured as WRCS# are used as WE# signals for external devices, no problems are expected, since a tristate waitstate should be used anyway due to the negative address hold time after WRCS# (t55) without tristate WS.

Workarounds:

- Use a memory tristate WS (i.e. leave bit BUSCONy.5 = 0) in all active BUSCON registers where RD/WR-CS# is used (i.e. bit BUSCONy.CSRENy = 1 and/or bit BUSCONy.CSWENy = 1), **or**
- Use Address-CS# instead of RD/WR-CS# (i.e. leave bits BUSCONy[15:14] = 00b) for all BUSCONy registers where a non-multiplexed bus without tristate WS is configured (i.e. bit BUSCONy.5 = 1).

BUS.18 PEC Transfers after JMPR instruction

Problems may occur when a PEC transfer immediately follows a taken JMPR instruction when the following sequence of 4 conditions is met (labels refer to following examples):

1. in an instruction sequence which represents a loop, a jump instruction (Label_B) which is capable of loading the jump cache (JMPR, JMPA, JB/JNB/JBC/JNBS) is taken
2. the target of this jump instruction directly is a JMPR instruction (Label_C) which is also taken and whose target is at address A (Label_A)
3. a PEC transfer occurs immediately after this JMPR instruction (Label_C)
4. in the following program flow, the JMPR instruction (Label_C) is taken a second time, and no other JMPR, JMPA, JB/JNB/JBC/JNBS or instruction which has branched to a different code segment (JMPS/CALLS) or interrupt has been processed in the meantime (i.e. the condition for a jump cache hit for the JMPR instruction (Label_C) is true)

In this case, when the JMPR instruction (Label_C) is taken for the second time (as described in condition 4 above), and the 2 words stored in the jump cache (word address A and A+2) have been processed, the word at address A+2 is erroneously fetched and executed instead of the word at address A+4.

Note: the problem does not occur when

- the jump instruction (Label_C) is a JMPA instruction
- the program sequence is executed from internal Flash/ROM/OTP

Example1:

```
Label_A:  instruction x                ; Begin of Loop
          instruction x+1
.....
Label_B:  JMP Label_C; JMP may be any of the following jump instructions:
             ; JMPR cc_zz, JMPA cc_zz, JB/JNB/JBC/JNBS
             ; jump must be taken in loop iteration n
             ; jump must not be taken in loop iteration n+1
.....
Label_C:  JMPR cc_xx, Label_A          ; End of Loop
             ; instruction must be JMPR (single word instruction)
             ; jump must be taken in loop iteration n and n+1
             ; PEC transfer must occur in loop iteration n
```

Example2:

```
Label_A:  instruction x                ; Begin of Loop1
          instruction x+1
.....
Label_C:  JMPR cc_xx, Label_A          ; End of Loop1, Begin of Loop2
             ; instruction must be JMPR (single word instruction)
            ; jump not taken in loop iteration n-1, i.e. Loop2 is entered
             ; jump must be taken in loop iteration n and n+1
             ; PEC transfer must occur in loop iteration n
.....
Label_B:  JMP Label_C                  ; End of Loop2
             ; JMP may be any of the following jump instructions:
             ;   JMPR cc_zz, JMPA cc_zz, JB/JNB/JBC/JNBS
             ; jump taken in loop iteration n-1
```

A code sequence with the basic structure of Example1 was generated e.g. by a compiler for comparison of double words (long variables).

Workarounds:

- use a JMPA instruction instead of a JMPR instruction when this instruction can be the direct target of a preceding JMPR, JMPA, JB/JNB/JBC/JNBS instruction, **or**
- insert another instruction (e.g. NOP) as branch target when a JMPR instruction would be the direct target of a preceding JMPR, JMPA, JB/JNB/JBC/JNBS instruction, **or**
- change the loop structure such that instead of jumping from Label_B to Label_C and then to Label_A, the jump from Label_B directly goes to Label_A.

Notes on compilers:

In the **Hightec** compiler beginning with version Gcc 2.7.2.1 for SAB C16x - V3.1 Rel. 1.1, patchlevel 5, a switch -m bus18 is implemented as workaround for this problem. In addition, optimization has to be set at least to level 1 with -u1.

The **Keil** C compiler versions < V4.0 and run time libraries do not generate or use instruction sequences where a JMPR instruction can be the target of another jump instruction, i.e. the conditions for this problem do not occur.

In V4.0, the problem may occur when optimize (size or speed, 7) is selected. Lower optimization levels than 7 are not affected.

In V4.01, a new directive FIXPEC is implemented which avoids this problem.

In the **TASKING** C166 Software Development Tools, the code sequence related to problem BUS.18 can be generated in Assembly. The problem can also be reproduced in C-language by using a particular sequence of GOTOs.

With V6.0r3, TASKING tested all the Libraries, C-startup code and the extensive set of internal test-suite sources and the BUS.18 related code sequence appeared to be NOT GENERATED.

To prevent introduction of this erroneous code sequence, the TASKING Assembler V6.0r3 has been extended with the CHECKBUS18 control which generates a WARNING in the case the described code sequence appears. When called from within EDE, the Assembler control CHECKBUS18 is automatically 'activated'.

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.

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 AP2924xx "Interrupt Register behaviour of the CAN module in Siemens 16-bit Microcontrollers" on

http://www.infineon.com/products/ics/34/index3.htm

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

A problem with BFLDL and BFLDH instructions which follow a write operation to internal IRAM has been detected on Infineon 16-bit microcontrollers.

see: Early Problem Notification

PLL.3 Increased PLL Jitter caused by external Access

Problem description:

In systems were the PLL is used for generation of the CPU clock frequency, the PLL jitter can increase in 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 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 interface).

Value of additional jitter:

Under evaluation

Workaround:

In the case that the additional jitter causes problems in the system which cannot be relaxed with software adapted timing parameters the following workarounds are possible:

- If the clock generation is done with the on-chip oscillator then direct drive- or prescaler mode can be used.
- If the maximum frequency of the on-chip oscillator in 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 (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 $\overline{\text{NMI}}$ 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.7 Wake Up from Sleep while RTC switched off

After wake up from Sleep Mode the CPU is immediately started unless waiting for correct clock function. During the first phase of the oscillator startup time the input clock can be disturbed and in worst case f_{CPU} exceeds the specified value - resulting in CPU malfunctions.

Workaround:

Leave RTC on while in Sleep Mode (SYSCON1.1-0 = 01b)

POWER.9 ASC0 and SSC Output Latches disabled when PCDDIS = 1

When bit PCDDIS in register SYSCON3 is set the output latches of pin 3.10 ASC0 TxD, 3.8 SSC MRST (Slave Mode), 3.9 SSC MTSR (Master Mode) and 3.13 SSC SCLK (Master Mode) are disconnected from internal clock. Data and clock transmissions are stopped.

Workaround:

Keep bit PCDDIS in register SYSCON3 enabled in power saving modes.

POWER.10 Sleep Mode: CAN and ADC Modules Enabled

In sleep mode the peripheral modules CAN and ADC are still enabled.

Workaround:

Before entering in sleep mode disable CAN & ADC in SYSCON3
(e.g. Unlock Sequence + SCXT SYSCON3,#0FFFFh)

Note: CANDIS = 1 stops the CAN state machine i.e.: a running transmission is terminated immediately.

After return from sleep mode enable CAN & ADC in SYSCON3
(e.g. Unlock Sequence + POP SYSCON3)

Example Program:

```
    MOV    SYSCON2, ZEROS; reset unlock state machine
    EXTR   #4
    BFLDL  SYSCON2, #0Fh, #09h
    MOV    SYSCON2, #0003h
    BSET   SYSCON2.2
enable_sleep_mode:
    MOV    SYSCON1, #00x1b; x = 0: RTC on, x = 1: RTC off
WORKAROUND:
    EXTR#4
    BFLDL  SYSCON2, #0Fh, #09h
    MOV    SYSCON2, #0003h
    BSET   SYSCON2.2
    SCXT   SYSCON3, ONES;   disable all peripherals
enter_sleep_mode:
    ATOMIC #3
    PUSH   PSW;             save current ILVL
    BFLDH  PSW, #0F0h, 0F0h; ILVL = 15
    IDLE
return_from_sleep:
    EXTR#4
    BFLDL  SYSCON2, #0Fh, #09h
    MOV    SYSCON2, #0003h
    BSET   SYSCON2.2
    POP    PSW;            restore ILVL
    POP    SYSCON3;        restore peripheral release setting
```

POWER.12 Sleep Mode: RTC cannot be switched off

Bit SYSCON1.1 is not connected to the internal oscillator circuit: Even in sleep mode with RTC off the real time clock and the oscillator (main and auxiliary - if implemented) keep running and the power consumption is like Power Down mode with RTC running (Main Osc.).

Workaround:

Not possible

POWER.13 Sleep Mode cannot be terminated by HW Reset

Workaround:

NMI interrupt wakes up the device from sleep mode and allows HW reset:

- Combine pins NMI and RSTIN.

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 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 (fCPU = fOSC/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. fCPU)

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.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 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:

None

2 Deviations from Electrical- and Timing Specification

Problem Short Name	Parameter	Symbol	Limit Values		Unit	Test Condition
			min.	max.		
DC.HYS.1[1)]	Input Hysteresis (Special Threshold)	HYS	**300** [1)] instead of 400	–	mV	–
DC.IPDR5.2	Power-down mode supply current with RTC running	I_{PDR5}	–	**200** + 25 x f_{OSC} instead of 100 + 25 x f_{OSC}	µA	V_{DD} = 5.5V f_{OSC} in [MHz]
DC.IPDR5.4	Sleep Mode Supply Current with RTC enabled	I_{PDR5}	–	0.6 @ f_{OSC} = 4 MHz instead of 100 + 25 x f_{OSC}	mA µA	V_{DD}= V_{DDMAX} f_{OSC} in [MHz]
DC.IPDO5.3	Sleep Mode Supply Current with RTC disabled	I_{PDO5}	–	**0.6** instead of 50	mA µA	V_{DD}= V_{DDMAX}
DC.VILS.1[1)]	Input low voltage (Special Threshold)	V_{ILS}	- 0.5	**1.7** [1)] instead of 2.0	V	–

1) If $f_{CPU} \leq 16$ MHz the specified value is guaranteed by design.

Note: Timing t28: 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 P4[3:0] are active, independent whether the respective pins are used for CS# function after reset or not.

3 History List

(no previous device step)

Table 1 Functional Deviations

Functional Problem	Short Description	Fixed in step
ADC.11	Modifications of ADM field while bit ADST = 0	
ADC.13	ADC Channel Injection Trigger	
CAN.7	Unexpected Remote Frame Transmission	
CAPCOM.4	SW Access to P1H Overwrites CAPCOM HW Settings	
RTC.4	Undefined Results after write to T14	
BUS.17	Spikes on CS# lines after access with RDCS# and/or WRCS#	
BUS.18	PEC transfers after JMPR	
X9	Read Access to XPERs in Visible Mode	
CPU.21	BFLDL/BFLDH Instructions after Write Operation to internal IRAM	
PLL.3	Increased PLL Jitter caused by external Access	
OSC.1	Internal Oscillator Circuit does not work	AB
OSC.2	Clock generation: Prescaler mode does not work	AB
PWRDN.1	Execution of PWRDN instruction while Pin NMI# = high	
POWER.1	internal Reset generation when PLL is switched off	AB
POWER.7	Wake Up from Sleep while RTC switched off	
POWER.9	ASC0 and SSC Output Latches disabled when PCDDIS = 1	
POWER.10	Sleep Mode: CAN and ADC Modules enabled	
POWER.12	Sleep Mode: RTC cannot be switched off	
POWER.13	Sleep Mode cannot be terminated by HW Reset	
POWER.14	Wake Up from Sleep Mode not possible in PLL Mode	
POWER.15	Wake Up from Sleep Mode not possible in Prescaler Mode	
RST.8	Clock failure detection during external Reset	
RST.13	Power Up with missing Clock	

Table 2 AC/DC Deviations

AC/DC Deviations	Short Description	Fixed in step
DC.HYS.1	Input Hysteresis	
DC.IPDR5.2	Power-down mode supply current with RTC enabled	
DC.IPDR5.4	Sleep Mode Supply Current with RTC **en**abled	
DC.IPDO5.3	Sleep Mode Supply Current with RTC **dis**abled	
DC.VDD.2	see C164 Data Sheet 1999-08, Operating Condition Parameters	
DC.VILS.1	Input low voltage (Special Threshold) 1.7 V	

4 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 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:
...
```

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. Operation at lower input frequencies is possible but is guaranteed by design only (not 100% tested).

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 AP2924xx "Interrupt Register Behaviour of the CAN Module in Siemens 16-bit Microcontrollers" on:

http://www.infineon.com/products/micro/applicat/3461.htm

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).

5 Documentation Update

C164 User's Manual Version 2.0 is available

Single Chip Reset Configuration Byte for Single Chip Reset

The Reset Configuration Byte (see description below) used in single-chip mode reset must be stored at location 00'003EH within the user ROM code. The Target Specification V1.0 (page 17) erroneously defines the highest ROM location.

Single-Chip Mode Reset

In contrast to the user's manual in the C164CI-8R the register RSTCON is not available. After single-chip mode reset the register RP0H is not set to the described default value but shows the configuration defined in the ROM mask (Configuration Byte) as follows.

For a single-chip mode reset (indicated by $\overline{EA}$ = 1) the configuration via PORT0 can be replaced by a fixed configuration value. This fixed configuration value is hardwired and is selectable for each ROM mask. In this case PORT0 needs no external circuitry (pullups/pulldowns) and also the internal configuration pullups are not activated.

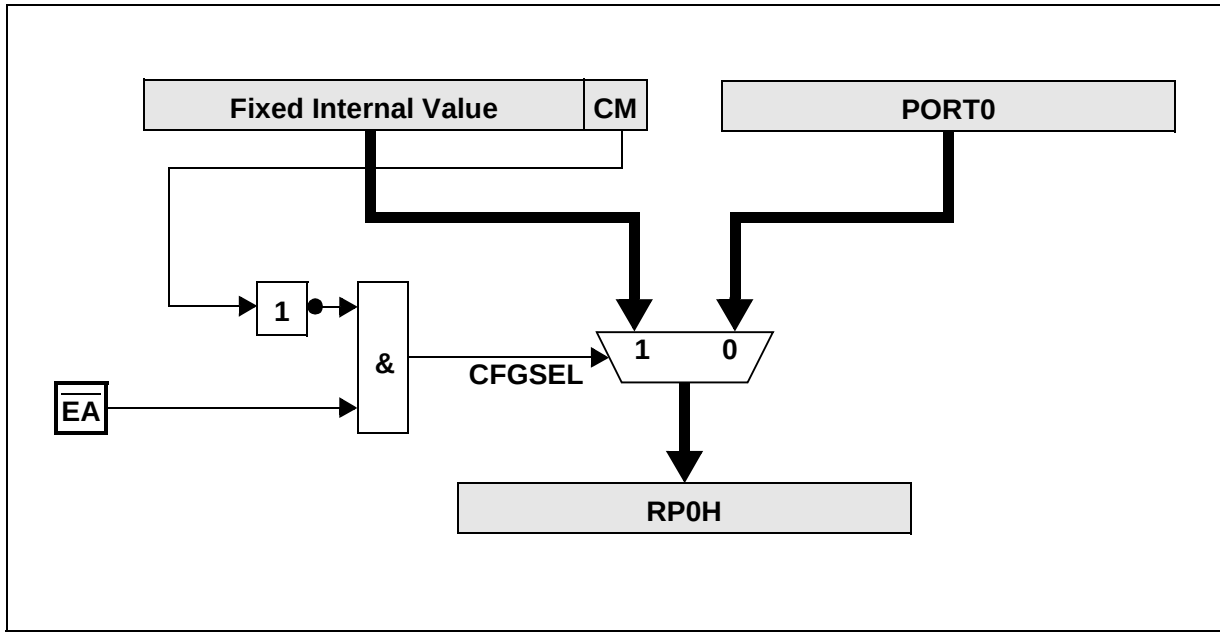


Figure 1 Reset Configuration Source Selection

The logic described in the figure above controls the reset configuration source (see signal CFGSEL) in the following way:

External configuration via PORT0 is selected
if $\overline{EA}$ = 0 *or* if compatibility mode bit CM = 1 (even if $\overline{EA}$=1)

Internal configuration via fixed value (Configuration Byte) is selected
if $\overline{EA}$ = 1 *and* compatibility mode bit CM = 0

This provides two possibilities (selectable with each ROM mask) for single-chip reset ($\overline{EA}$ = 1):

- Compatible mode, configuration via PORT0
- Internal mode, configuration via fixed value

The fixed internal value and the compatibility mode control bit CM for the Single-Chip reset are specified by a byte value described below. This byte value shall be stored at location 00'003EH within the user ROM code.

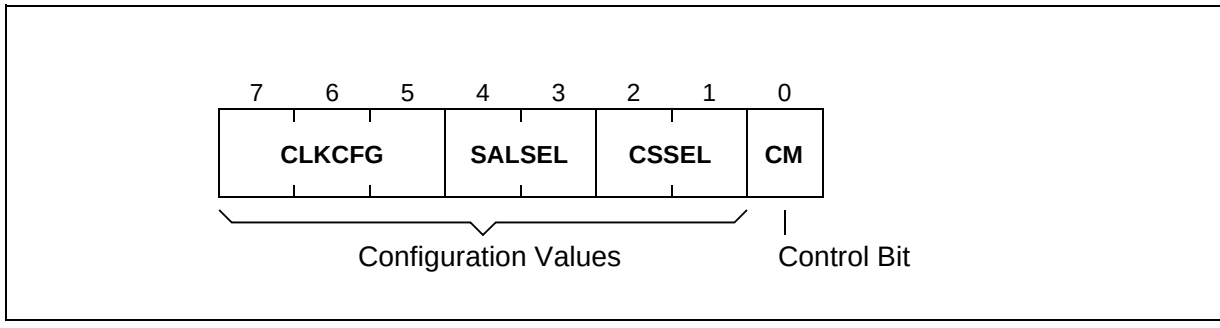


Figure 2 Internal Reset Configuration Byte

Output Driver Control

In contrast to the user's manual in the C164CI-8R the registers POCONx are not available. The output drivers edge characteristic can be configured by register PDCR instead.

Edge Characteristic

This defines the rise/fall time for the respective output, i.e. the output transition time. Slow edges reduce the peak currents that are drawn when changing the voltage level of an external capacitive load. For a bus interface, however, fast edges may still be required. Edge characteristic effects the pre-driver which controls the final output driver stage.

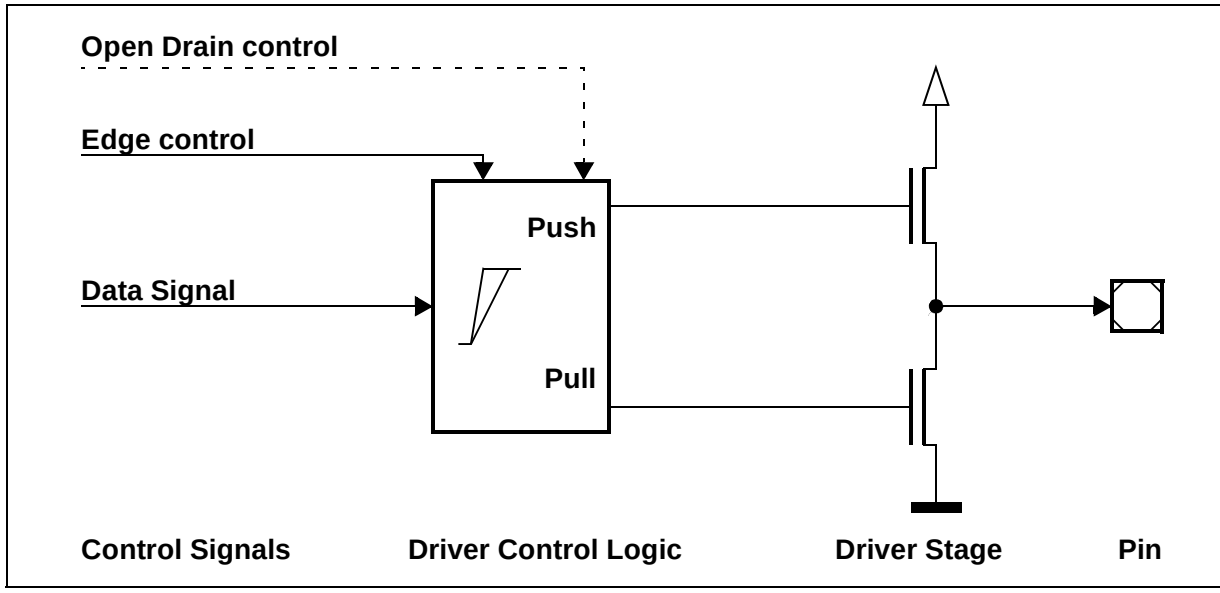


Figure 1 Structure of Output Driver with Edge Control

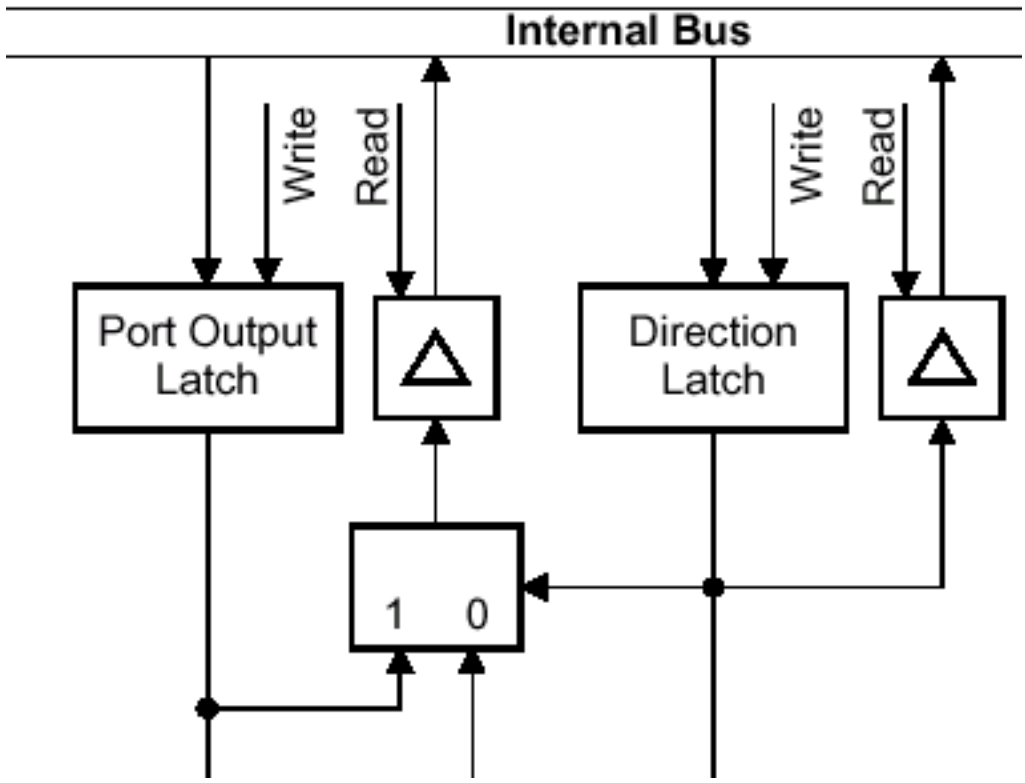
The **Port Driver Control Register** PDCR provides the corresponding control bits. A separate control bit is provided for bus pins as well as for non-bus pins.

PDCR
Port Driver Control Reg. **ESFR ($F0AA_H/55_H$)** **Reset value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	**NBPEC**	-	-	-	**BIPEC**
-	-	-	-	-	-	-	-	-	-	-	rw	-	-	-	rw

Bit	Function
BIPEC	**Bus Interface Pins Edge Characteristic** (Defines the outp. rise/fall time t_{RF}) 0: Fast edge mode, rise/fall times depend on the driver's dimensioning. 1: Reduced edge mode. BIPEC controls: PORT0, PORT1, Port 4, $\overline{RD}$, $\overline{WR}$, ALE, CLKOUT, $\overline{BHE}$/$\overline{WRH}$, $\overline{RSTOUT}$, $\overline{RSTIN}$ (bidirectional reset mode only).
NBPEC	**Non-Bus Pins Edge Characteristic** (Defines the output rise/fall time t_{RF}) 0: Fast edge mode, rise/fall times depend on the driver's dimensioning. 1: Reduced edge mode. NBPEC controls: Port 3, Port 8

Correct Port Output Latch read multiplexer description:



Application Support Group, Munich