Errata Sheet

V1.4, 2005-09-02

Device	**C164CI-8R, C164SI-8R, C164CL-8R, C164SL-8R, C164xy-L**
Marking/Step	**(E)ES-CA, CA, CA+**
Package	**P-MQFP-80**

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

- **C164CI/CL, SI/SL Data Sheet, V2.0, May 2001**
- **C164CI/CL, SI/SL User's Manual, V3.1, Feb. 2002**
- **Instruction Set 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.

Note: For simplicity all versions are referred to by the term C164CI-8R throughout this document.

Contents

1 History List/Change Summary

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

Table 1 Functional Deviations

Functional Problem	Short Description	Fixed in step	Change
ADC.11	Modifications of ADM field while bit ADST = 0		
ADC.13	ADC Channel Injection Trigger	AC	
SSC.9	Data Transmission in Slave Mode		
CAN.7	Unexpected Remote Frame Transmission		
CAN.9	Contents of Message Object and Mask of Last Message Registers after Reset		
CAPCOM.4	SW Access to P1H Overwrites CAPCOM HW Settings		
CAPCOM6.4	Spike when enabling alternate Port Function		
RTC.4	Undefined Results after write to T14	AC	
BUS.17	Spikes on $\overline{CS}$ lines after access with $\overline{RDCS}$ and/or $\overline{WRCS}$	AC	
BUS.18	PEC transfers after JMPR	AC	
X9	Read Access to XPERs in Visible Mode		
CPU.21	BFLDL/BFLDH Instructions after Write Operation to internal IRAM	CA	
CPU.22	Z Flag after PUSH and PCALL		
PLL.3.x	Increased PLL Jitter caused by external Access		
PLL.4	PLL base frequency	see AC.PLL.1	
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 $\overline{NMI}$ = high		
POWER.1	internal Reset generation when PLL is switched off	AB	
POWER.7	Wake Up from Sleep while RTC switched off	AC	
POWER.9	ASC0 and SSC Output Latches disabled when PCDDIS = 1		
POWER.10	Sleep Mode: CAN and ADC Modules enabled	AC	
POWER.12	Sleep Mode: RTC cannot be switched off	AC	

Table 1 Functional Deviations

Functional Problem	Short Description	Fixed in step	Change
POWER.13	Sleep Mode cannot be terminated by HW Reset	AC	
POWER.14	Wake Up from Sleep Mode not possible in PLL Mode		
POWER.15	Wake Up from Sleep Mode not possible in Prescaler Mode		
POWER.22	Wake up from Sleep Mode in Oscillator Decay Time		
RST.8	Clock failure detection during external Reset		
RST.13	Power Up with missing Clock	AC	

Table 2 AC/DC Deviations

AC/DC Deviations	Short Description	Fixed in step	Change
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	AC *)	
DC.IPDO5.3	Sleep Mode Supply Current with RTC **dis**abled	AC	
DC.VDD.2	see C164 Data Sheet 1999-08, Operating Condition Parameters and Documentation Uptdate (DOC_ROM.D1)	AC	
DC.VILS.1	Input low voltage (Special Threshold) 1.7 V		
AC.PLL.1	PLL Base Frequency 2...6 MHz (instead of 2...5)		

*) Specification in C164 Data Sheet 1999-08 (and V2.0, May 2001) has been changed

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	
MainOsc.H1	Oscillator Type_LP2: Negative Resistance and Start-up Reliability	
MainOsc.H2	Maximum (Type_LP2) Oscillator Frequency = 16 MHz	
POWER.H1	Reduced Power Consumption	
OWD.H2	Oscillator Watchdog and Prescaler Mode	
ISNC.H1	Maintenance of ISNC Register	
EMUL.H1	Adapt Mode Setting	

Table 4 Documentation Update

Hint	Short Description	Change
DOC_ROM.D1	Differences to Data Sheet Version 2.0, May 2001 Updated: Ports: Output high voltage	updated
DOC_ROM.D2	Differences to User's Manual Version 3.1, Feb. 2002	
RESET.D1	Single Chip Mode Reset: Configuration Byte	
RESET.D2	Single-Chip Mode Reset	
PORT.D2	Output Driver Control	
CAN.D1	On Chip CAN - Bitfield IPC is not available	
VDD.D2	Data Sheet V2.0, 2001-05 - Operating Conditions	
CAPCOM2.D1	CAPCOM2 Channel Port Connections	
AC.t20_t21.D1	Typo in Data Sheet	
ID-Registers	ID-Registers	

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 ADCH (ADCON.3-0) = n during this conversion)

2. the A/D converter is idle (i.e. ADBSY (ADCON.8) = 0)

3. then the conversion mode in the ADC Mode Selection field ADM (ADCON.5-4) is changed to Auto Scan Single (ADM = 10_B) or Continuous (ADM = 11_B) 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 (ADCIC.7) 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 ADWR (ADCON.9) = 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.

SSC.9 Data Transmission in Slave Mode

During data reception in slave mode of the SSC module, sporadically the shift clock supplied by the external master on pin SCLK may not be properly recognized due to a synchronization problem when all of the following conditions are true:

1. the latching edge for the serial data is the **falling** edge of SCLK (i.e. both bit SSCPO = 1 and bit SSCPH = 1, or SSCPO = 0 and SSCPH = 0 in register SSCCON), and
2. the transmit buffer **SSCTB** of the slave has **not** been **written** prior to the start of the reception (initiated by the master asserting the shift clock SCLK), and
3. a specific time window (phase delay) is hit by the serial shift clock SCLK in relation to the internal system clock of the slave. Therefore, this synchronization problem will occur in particular when the slave device is clocked (on XTAL1) by an external clock generation circuit which is independent from the clock generation circuit of the master (i.e. slave and master clocks are **asynchronous**).

When the problem occurs, this results in missing bits in the character received in SSCTB, and in duplicated bits in the character transmitted on pin MRST of the slave. As a consequence, interrupt generation in the slave is delayed by the number of missed bits.

Workaround

For systems using the falling edge of SCLK as latching edge (see condition 1. above), always write to the transmit buffer SSCTB prior to any reception in slave mode of the SSC module. For the second and all following characters, e.g. write a (dummy) character to SSCTB in the receive interrupt routine, or use a PEC transfer triggered by the transmit interrupt request to write to SSCTB. In this case, the critical synchronization path is not used, and the problem will not occur.

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.

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.

CAPCOM6.4 Spike when enabling alternate Port Function

When switching a port from I/O mode to compare mode during the low state of the PWM by setting bits in register CC6MSEL, a spike is generated for one TCL before the PMW signal starts.

Workaround:

When the port configuration is changed during the high phase of the compare output the spike is suppressed. Two different interrupts can be used for this synchronization depending on the initial value bits in the register CC6MCON (CCxI and COUTxI represent the passive output level for the enabled compare channels). The first high phase of the PWM signal is delayed by the corresponding interrupt latency and the execution time of the necessary instructions:

- If the initial value is 0, the compare interrupt service routine switches the port to compare output. So the spike is pushed in the high phase of the PWM.
- If the initial value is 1, the Timer 12 period match interrupt service routine switches the port to compare mode with the same result.

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.

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/16-bit-microcontrollers

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.x Increased PLL Jitter caused by external Access

When external bus mode is used and the CPU frequency is in the range from 10 to 15 MHz the PLL jitter is higher than specified.

Details are under evaluation.

The problem does not occur in single chip mode whithout external bus.

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 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.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.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.22 Wake up from Sleep Mode in Oscillator Decay Time

Due to a synchronization problem, the device may not correctly wake up from sleep mode if an asynchronous external wake up trigger occurs already in the oscillator decay time, i.e. in a particular time window before the device has actually entered sleep mode (see figure)

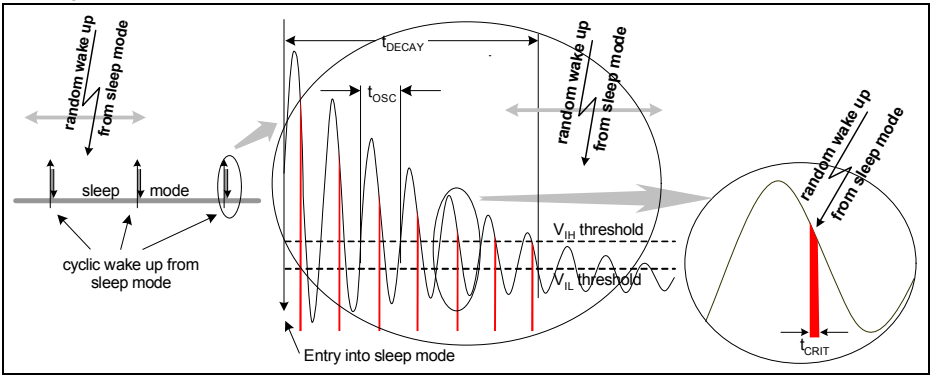


Figure 1 Critical Event

This problem may occur when all of the following conditions are met at the same time:

1. the device is configured for sleep mode with the following settings where the main oscillator (input XTAL1) will be switched off, i.e.
   a) with RTC and oscillator to be stopped (SYSCON1 bitfield SLEEPCON = 11_B), or
   b) only for devices with XTAL3 input:
   with RTC running and supplied by the clock from XTAL3 (SYSCON1 bitfield SLEEPCON = 01_B and SYSCON2 bit RCS = 1)
2. the IDLE instruction to enter sleep mode has been executed, i.e. the main oscillator is in the oscillator decay time (typically ~ 160 µs decay time for a 16 MHz crystal, ~ 2 µs for a 16 MHz ceramic resonator)

3. the external wake up trigger signal (fast external interrupt or signal on pin $\overline{\text{NMI}}$) coincides with the critical time window (t_{CRIT}) of each oscillator cycle during the decay time of the clock signal at pin XTAL1 (critical times shown in red in figure)

In this case the clock system may get blocked but the internal program memory module becomes enabled. I_{DDI} rises up to I_{PDR} ... 3 mA with main oscillator.

The watchdog timer - stopped in sleep mode - cannot restart the system. Only HW reset restarts the system properly.

The problem occurs only in the decay time of the main oscillator and in a very small window around the clock edge at XTAL1 (t_{CRIT} is about 5 ps wide).

The following formula shows a risk calculation for an asynchronous (random) wake up event at any time within a 100 ms sleep period (16 MHz crystal/resonator):

$$P_{\text{Lock-Up}} = P_{\text{Sensitive Device}} * P_{\text{Critical Wake-Up-Event}}$$

$P_{\text{Lock-Up}}$ = 5 ps (critical window) / 62500 ps (one 16 MHz oscillator period)
* 160 µs (typical **crystal** decay time) / 100000 µs (wake up after 100 ms)

$P_{\text{Lock-Up}}$ = 5 / 62500 * 160 / 100000 = 0.000'000'13
= 0.13 failure events per 1 million random wake up events (for a 16 MHz crystal)

The decay time of a **resonator** circuit is below 2 µs typically:

$P_{\text{Lock-Up (resonator)}}$ = 5 / 62500 * 2 / 100000 = 0.000'000'001'6
= 0.0016 failure events per 1 million random wake up events
(for a 16 MHz resonator)

Note: A cyclic wake up event would never cause the problem as it happens well after entry into sleep mode and after the decay time.

Workarounds:

- Select RTC on in sleep mode (SYSCON1.SLEEPCON = 01_B).
  For devices with auxiliary oscillator (e.g. C161CS), select main oscillator as clock source (SYSCON2.RCS = 0).
- Avoid a wake up during the main oscillator decay time

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.

3 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.VILS.1[1)]	Input low voltage (Special Threshold)	V_{ILS}	- 0.5	**1.7** [1)] instead of 2.0	V	–
AC.PLL.1	PLL base frequency		2	**6** instead of 5	MHz	–

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 $\overline{CS}$ 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:

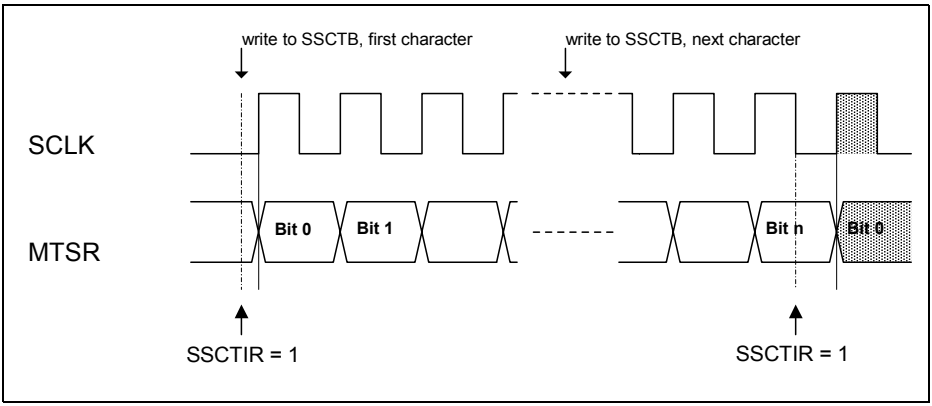


Figure 2 SSCTB Timing

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

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').

POWER.H1 Reduced Power Consumption

Due to design optimization in step CA the power consumption in active and idle mode has been reduced. The typical value is about 10% lower than in previous steps.

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).

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:
...
```

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.

5 Documentation Update

DOC_ROM.D1 Differences to Data Sheet Version 2.0, May 2001

To merge the documentation of all C164CI derivatives the Data Sheet Version 2.0, May 2001 will be used as the standard description. The following table describes the differences of the C164CI-8R Steps AC and CA to this document.

Table 5 Documentation Update

Function	Short Description	Data Sheet V2.0, 2001-05	
		affected section	page
Extension RAM	External XRAM Access (this mode was not described before)	AC Characteristics	73
CAN pin assignment	CAN RxD and TxD are fixed assigned to Port 4 (P4.5 and P4.6), CAN Register IR is implemented instead of PCIR (no bitfield IPC) - see **CAN.D1**	• Figure 2 • Table 2 Pin Definitions and Functions • CAN-Module	4 10 27
External Bus Controller	Demultiplexed Bus Mode available (this mode was not described before)	• Ext. Bus Controller • AC Characteristics	13 65 ff
CAPCOM6	Version 4 implemented	The Capture/ Compare Unit CAPCOM6	21
Watchdog Timer	No input prescaler implemented only f_{CPU} / 2 or f_{CPU} / 128 available time interval 26 µs ... 422 ms @ 20 MHz default interval 6.55 ms @ 20 MHz	Watchdog Timer	27
Port Driver	Port driver characteristics controlled by register PDCR (for 2 groups: ext. bus ports and others) instead of registers POCONx - see **PORT.D2**	• Parallel Ports • Operating Conditions (footnote 5)	28 42

Table 5 Documentation Update (cont'd)

Function	Short Description	Data Sheet V2.0, 2001-05	
		affected section	page
Reset configuration via Pin $\overline{RD}$	• In **bus mode** (pin $\overline{EA}$ = 0) or single chip mode (pin $\overline{EA}$ = 1) while CM = 1 at the end of a reset bit OWDDIS in register SYSCON reflects the inverted level of pin $\overline{RD}$ at that time. • In **single chip mode** (pin $\overline{EA}$ = 1) while bit **CM = 0** (configuration byte, address $003E_H$ see **RESET.D1**) a low level at pin $\overline{RD}$ starts the bootstrap loader additionally.	Oscillator Watchdog	29
Digital supply voltage	Minimum 4.5 V	Operating Conditions	42
Ports	Output low voltage $V_{OL\ max}$ = 0.45 V @ I_{OL} = 2.4 mA ($\overline{PORT0}$, PORT1, Port 4, ALE, $\overline{RD}$, $\overline{WR}$, $\overline{BHE}$, CLKOUT, $\overline{RSTOUT}$)	DC Characteristics	43
Ports	Output low voltage $V_{OL1\ max}$ = 0.45 V @ I_{OL} = 1.6 mA (all other outputs)	DC Characteristics	43
Ports	Output high voltage $V_{OH\ min}$ = 2.4 V @ I_{OL} = -2.4 mA $V_{OH\ min}$ = 0.9 V_{DD} @ I_{OH} = -0.5 mA ($\overline{PORT0}$, PORT1, Port 4, ALE, $\overline{RD}$, $\overline{WR}$, $\overline{BHE}$, CLKOUT, $\overline{RSTOUT}$)	DC Characteristics	43
Ports	Output high voltage $V_{OH\ min}$ = 2.4 V @ I_{OL} = -1.6 mA $V_{OH\ min}$ = 0.9 V_{DD} @ I_{OH} = -0.5 mA (all other outputs)	DC Characteristics	43
$\overline{RSTIN}$ inactive / active current	To meet the real test condition the pullup resistor definition is replaced by "$\overline{RSTIN}$ inactive current" and "$\overline{RSTIN}$ active current"	DC Characteristics footnote 6	44

Table 5 Documentation Update (cont'd)

Function	Short Description	Data Sheet V2.0, 2001-05	
		affected section	page
Port drivers	The Maximum Output Current for port output drivers is 5 mA per pin (nominal 2.4 mA for PORT0, PORT1, PORT4, ALE, RD, WR, BHE, CLKOUT, RSTOUT and 1.6 mA for all remaining ports). The pin drivers have a fixed configuration (control register PDCR is implemented instead of POCONx)	DC Characteristics The absolute sum of input overload currents on **all** port pins may not exceed 50 mA.	45
Power Consumption	(Sleep and) Power-down mode supply current with RTC running: I_{PDR} = **200** + 25 * f_{OSC}	Power Consumption	45
Package	P-MQFP-80-7 vs -1 in previous data sheet	Package Outlines	74

CAPCOM6

- The Block Commutation sequence table for left rotation is not a standard commutation table, but rather is shifted against the standard pattern by 60 degrees.
- Cout63 is switched to P1L.6 while CTRAP = low.
- In case of a trap event (CTRAP = low) CAPCOM6 outputs are not switched to their initial values (CC6MCON, bits 0..5 and bit 7), but to port register P1L instead.
- switches **TT12DIS and TT13DIS are not available**

For more details see application note AP1673: Technical details on CAPCOM6 module:

http://www.infineon.com/cmc_upload/documents/042/818/ap1607310_.pdf

DOC_ROM.D2 Differences to User's Manual Version 3.1, Feb. 2002

To merge the documentation of all C164CI derivatives the User's Manual Version 3.1, Feb. 2002 will be used as the standard description. The following table describes the differences of the C164CI-8R Steps AC and CA to this document.

Table 6 Documentation Update

Function	Short Description	User's Manual Version 3.1, Feb. 2002	
		affected section	page
Register EXISEL	External Interrupt Source Control Only EXI0SS (CAN1_RxD) is implemented. ASC0 and SSC can **NOT** be used as alternate interrupt source.	5.8 External Interrupts	5-29
Parallel port characteristic	The output drivers edge characteristic is controlled by register PDCR - see 'PORT.D2 Output Driver Control' for details.	7.2 Output Driver Control	7-5 ff
Temperature Compensation	The Port Driver Temperature Compensation is **NOT** implemented.	7.2 Output Driver Control	7-8 ff
Single Chip Mode Reset Configuration	The register RSTCON is **NOT** implemented. See 'RESET.D2 Single-Chip Mode Reset' for details.	20.4.2 System Startup Configuration at Single-Chip Mode Reset 20.5 System Configuration via Software	20-20 ff 20-22 ff

RESET.D1 Single-Chip Mode 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 00'003E$_H$ within the user ROM code. The Target Specification V1.0 (page 17) erroneously defines the highest ROM location.

RESET.D2 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.

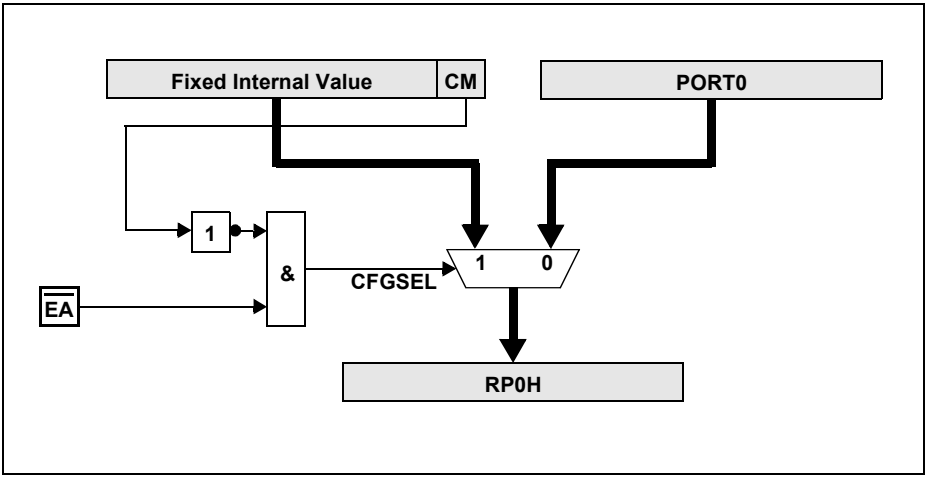


Figure 3 Reset Configuration Source Selection

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 00'003E$_H$ within the user ROM code.

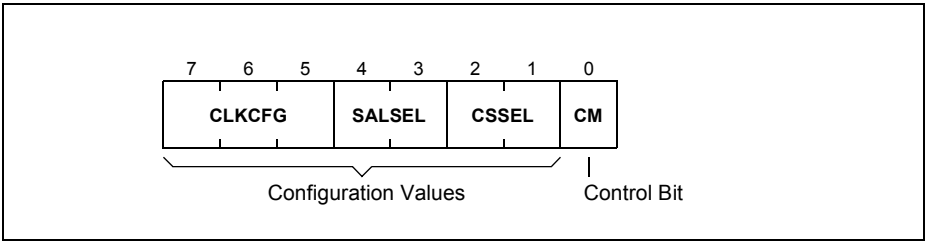


Figure 4 Internal Reset Configuration Byte

PORT.D2 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.

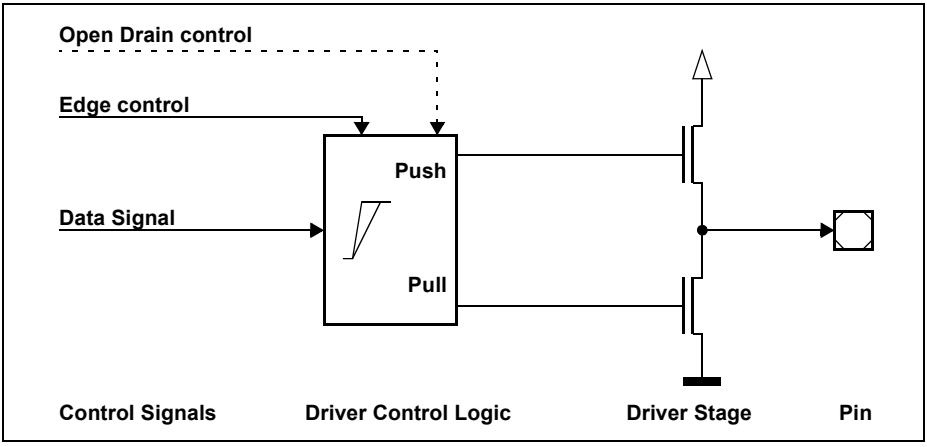


Figure 5 Structure of Output Driver with Edge Control

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
-	-	-	-	-	-	-	-	-	-	-	NBP EC	-	-	-	BIP EC
-	-	-	-	-	-	-	-	-	-	-	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

The figure below summarizes the effects of the driver characteristics:

Edge characteristic generally influences the output signal's **shape**.

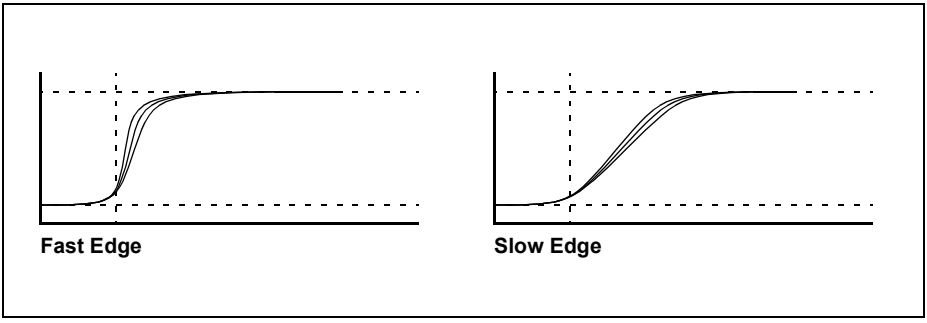


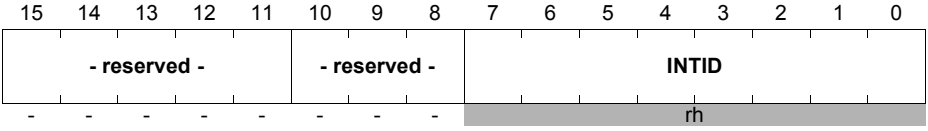
Figure 6 General Output Signal Waveforms

CAN.D1 On Chip CAN - Bitfield IPC is not available !

In contrast to the user's manual in the C164CI-8R no Interface Port Control is implemented. Thus the register at address $EF02_H$ is the Interrupt Register only (user's manual V2.0 page 19-10 ff.). The Port Control function (user's manual V2.0 page 19-36 ff.) is not implemented.

The function of the interrupt register and the CAN signal lines is as follows.

PCIR
Port Control / Interrupt Register XReg ($EF02_H$) **Reset value: $XXXX_H$**



Bit	Function	
INTID	**Interrupt Identifier** This number indicates the cause of the interrupt (if pending).	
	00_H	**Interrupt Idle:** There is no interrupt request pending.
	01_H	**Status Change Interrupt:** The CAN controller has updated (not necessarily changed) the status in the Control Register. This can refer to a change of the error status of the CAN controller (EIE is set and BOFF or EWRN change) or to a CAN transfer incident (SIE must be set), like reception or transmission of a message (RXOK or TXOK is set) or the occurrence of a CAN bus error (LEC is updated). The CPU may clear RXOK, TXOK, and LEC, however, writing to the status partition of the Control Register can never generate or reset an interrupt. To update the INTID value the status partition of the Control Register must be read.
	02_H	**Message 15 Interrupt:** Bit INTPND in the Message Control Register of message object 15 (last message) has been set. The last message object has the highest interrupt priority of all message objects. [1)]
	(02 + N)	**Message N Interrupt:** Bit INTPND in the Message Control Register of message object 'N' has been set (N = 1 … 14). Note that a message interrupt code is only displayed, if there is no other interrupt request with a higher priority. [1)] Example: message 1: INTID = 03_H, message 14: INTID = 10_H

1) Bit INTPND of the corresponding message object has to be cleared to give messages with a lower priority the possibility to update INTID or to reset INTID to "00_H" (idle state).

The CAN Application Interface

The on-chip CAN module of the C164CI-8R is connected to the (external) physical layer (i.e. the CAN bus) via two signals:

Table 5-1 CAN Interface Signals

CAN Signal	Port Pin	Function
CAN1_RXD	Port 4.5	Receive data from the physical layer of the CAN bus.
CAN1_TXD	Port 4.6	Transmit data to the physical layer of the CAN bus.

A logic low level (‘0’) is interpreted as the dominant CAN bus level, a logic high level (‘1’) is interpreted as the recessive CAN bus level.

Port Control

This function is not implemented in the C164CI-8R (i.e. the CAN module has a fixed connection to Port 4.5...6 when bit XPEN in register SYSCON is set). See also register PCIR above.

Device Specification

*) Only Port 4 of the marked pins can have CAN interface lines assigned to them (see User’s Manual V3.0, page 25-2).

VDD.D2 Data Sheet V2.0, 2001-05 - Operating Conditions

In contrast to Table 9 (Data Sheet V2.0, page 42) $V_{DD\ min.}$ is 4.5 V for all devices listed in Table 1 (Data Sheet V2.0, page 2).

CAPCOM2.D1 CAPCOM2 Channel Port Connections

In contrast to the user's manual in the C164CI-8R only CC19IO...CC16IO pins of CAPCOM2 have the output function in compare mode:

Table 16-1 CAPCOM2 Channel Port Connections

Unit	Channel	Port	Capture	Compare
CAPCOM2	CC16IO … CC19IO	P8.3 … P8.0	Input	Output
	CC20IO … CC23IO	–	–	–
	CC24IO … CC27IO	P1H.7 … P1H.4	Input	**–**
	CC28IO … CC31IO	–	–	–
	Σ = 16	Σ = 8	Σ = 8	**Σ = 4**

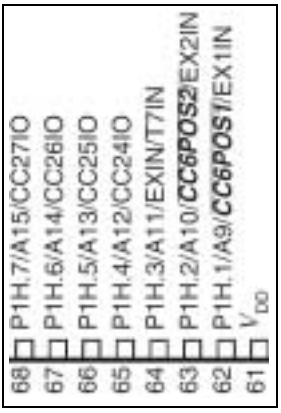


Figure 25-1 Pin Description for C164, P-MQFP-80 Package

AC.t20_t21.D1 Typo in Data Sheet

Parameter	Symbol	Max. CPU Clock = 25 MHz		Variable CPU Clock 1 / 2TCL = 1 to 25 MHz		Unit
		min.	max.	min.	max.	
Data float after $\overline{RD}$ rising edge (with RW-delay)	t_{20} SR	–	26 + $2t_A + t_F$	–	2TCL - 14 + **2**$t_A + t_F$ **instead of** ...+ 22t_A ...	ns
Data float after $\overline{RD}$ rising edge (no RW-delay)	t_{21} SR	–	10 + $2t_A + t_F$	–	TCL - 10 + **2**$t_A + t_F$ **instead of** ...+ 22t_A ...	ns

ID-Registers

		Register:	IDMANUF	IDCHIP	IDMEM	IDPROG	IDMEM2
Device	**Step**	**Address:**	**F07E$_H$**	**F07C$_H$**	**F07A$_H$**	**F078$_H$**	**F076$_H$**
C164CI-8R	-AC		1820$_H$	0A02$_H$	1010$_H$	0000$_H$	0000$_H$
C164CI-8R	-CA		1820$_H$	0A03$_H$	1010$_H$	0000$_H$	0000$_H$
C164SI-8R	-AC		1820$_H$	1B02$_H$	1010$_H$	0000$_H$	0000$_H$
C164SI-8R	-CA		1820$_H$	1B03$_H$	1010$_H$	0000$_H$	0000$_H$

Product and Test Engineering Group, Munich