SIEMENS

Since April 1, 1999, Siemens Semiconductor is Infineon Technologies. The next revision of this document will be updated accordingly.

Errata Sheet

November 23, 1998 / Release 1.2

Device: **SAK-C167CR-4RM**

Stepping Code / Marking: **ES-DA, DA**

Package: **MQFP-144**

This Errata Sheet describes the deviations from the current user documentation. The classification and numbering system is module oriented in a continual ascending sequence over several derivatives, as well already solved deviations are included. So gaps inside this enumeration could occur.

The current documentation is: Data Sheet: C167CR-4RM Data Sheet 07.97,
User's Manual: C167 Derivatives User's Manual V2.0 03.96
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.

Change summary to Errata Sheet Rel.1.1 for devices with stepping code/marking ES-DA, DA:

- ADC Overload Current (ADCC.2.2): description modified (step specific)
- Spikes on CS# lines after access with RDCS# and/or WRCS# (BUS.17)
- PEC Transfers after JMPR (BUS.18)
- Note about workaround for Tasking compiler added for problem CPU.16
- Note on Interrupt Register Behaviour of CAN module added
- Modifications of ADM field while bit ADST = 0 (ADC.11): reference to single channel continuous mode in condition (1) eliminated
- Read Access to XPERs in Visible Mode (X9): description modified
- new naming convention for DC/AC specification deviations used
- Output low voltage V_{OL} (bus pins) tested according to specification (DC.VOL.1)
- Input hysteresis HYS (special threshold) tested according to specification (DC.HYS.1)

Functional Problems:

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

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

a) the instructions following the PWRDN instruction are located in external memory, and a **multiplexed bus** configuration **with memory tristate waitstate** (bit MTTCx = 0) is used, or

b) the instruction preceding the PWRDN instruction **writes** to external memory or an XPeripheral (XRAM, CAN), and the instructions following the PWRDN instruction are located in external memory. In this case, the problem will occur for any bus configuration.

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

Workaround:

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

CPU.16: Data read access with MOVB [Rn], mem instruction to internal ROM/Flash/OTP

When the *MOVB [Rn], mem* instruction (opcode 0A4h) is executed, where

1. mem specifies a direct 16-bit byte operand address in the internal ROM/Flash memory,

AND

2. *[Rn]* points to an **even** byte address, while the contents of the word which includes the byte addressed by *mem* is **odd**,

OR

[Rn] points to an **odd** byte address, while the contents of the word which includes the byte addressed by *mem* is **even**

the following problem occurs:

a) when *[Rn]* points to **external** memory or to the **X-Peripheral** (XRAM, CAN, etc.) address space, the data value which is written back is always 00h

b) when *[Rn]* points to the **internal** RAM or SFR/ESFR address space,

- the (correct) data value [*mem*] is written to *[Rn]***+1**, i.e. to the **odd** byte address of the selected word in case *[Rn]* points to an **even** byte address,
- the (correct) data value [mem] is written to *[Rn]***-1**, i.e. to the **even** byte address of the selected word in case *[Rn]* points to an **odd** byte address.

Workaround:

When *mem* is an address in internal ROM/Flash/OTP memory, substitute instruction

MOVB [Rn], mem e.g. by

```
MOV   Rm, #mem
MOVB  [Rn], [Rm]
```

Notes on compilers:
- the **Keil** C166 Compiler V3.10 has been extended by the directive FIXROM which avoids accesses to 'const' objects via the instruction MOVB [Rn], mem.
- the **Tasking** compiler provides a workaround for this problem from version V6.0r2 on

CPU.17: Arithmetic Overflow by DIVLU instruction

For specific combinations of the values of the dividend (MDH,MDL) and divisor (Rn), the Overflow (V) flag in the PSW may not be set for unsigned divide operations, although an overflow occurred.

E.g.:

```
MDH   MDL      Rn          MDH  MDL
F0F0  0F0Fh : F0F0h  = FFFF FFFFh, but no Overflow indicated!
                          (result with 32-bit precision: 1 0000h)
```

The same malfunction appears for the following combinations:

```
n0n0 0n0n : n0n0
n00n 0nn0 : n00n
n000 000n : n000
n0nn 0nnn : n0nn        where n means any Hex Digit between 8 ...  F
```

i.e. all operand combinations where at least the most significant bit of the dividend (MDH) and the divisor (Rn) is set.

In the cases where an overflow occurred after DIVLU, but the V flag is not set, the result in MDL is equal to FFFFh.

Workaround:

Skip execution of DIVLU in case an overflow would occur, and explicitly set V = 1.

```
E.g.:       CMP Rn, MDH
            JMPR cc_ugt, NoOverflow     ; no overflow if Rn > MDH
            BSET V                      ; set V = 1 if overflow would occur
            JMPR cc_uc, NoDivide        ; and skip DIVLU
NoOverflow: DIVLU Rn
NoDivide:   ...                         ; next instruction, may evaluate correct V flag
```

Note:
- the KEIL C compiler, run time libraries and operating system RTX166 do not generate or use instruction sequences where the V flag in the PSW is tested after a DIVLU instruction.

- with the TASKING C166 compiler, for the following intrinsic functions code is generated which uses the overflow flag for minimizing or maximizing the function result after a division with a DIVLU:

```
_div_u32u16_u16()
_div_s32u16_s16()
_div_s32u16_s32()
```

Consequently, an incorrect overflow flag (when clear instead of set) might affect the result of one of the above intrinsic functions but only in a situation where no correct result could be calculated anyway. These intrinsics first appeared in version 5.1r1 of the toolchain.

Libraries: not affected

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

Spikes of about 5 ns width (measured at V_{OH} = 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:

1. 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**
2. 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:

1. 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
2. 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 ROM/Flash

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:

1. 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
2. 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
3. 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 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 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'.

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

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

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

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

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

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

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

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

Workaround:

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

X9: Read Access to XPERs in Visible Mode

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

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

X12: P0H spikes after XPER write access and external 8-bit Non-multiplexed bus

When an external 8-bit non-multiplexed bus mode is selected and P0H is used for general purpose I/O, and an internal (byte or word) write access to an XBUS peripheral (e.g. XRAM, CAN, or I²C module) is performed, and an **external** bus cycle is directly following the internal XBUS write cycle, then P0H is actively driven with the write data for approx. 7ns (spikes on P0H).

The spikes also occur if P0H is configured as input. However, read operations from P0H are not affected and will always return the correct logical state.

The spikes have the following position and shape in a typical application:

spikes occur after the rising edge of CLKOUT which follows the rising edge of ALE for the external bus cycle

P0H.x = low --> output low voltage rises to approx. 2.5V, spike width approx. 7ns (@ 0.2 Vcc)
P0H.x = high --> output high voltage drops to approx. 2.0V, spike width approx. 7ns (@ 0.8 Vcc)

Referring to a worst case simulation the maximum width of the spikes may be 15ns with full amplitude (Vcc/Vss). But this might not be seen on application level.

Note that if any of the other bus modes is selected in addition to the 8-bit non-multiplexed mode, P0H can not be used for I/O per default.

Workaround:

- use a different port instead of P0H for I/O when (only) an external 8-bit non-multiplexed bus mode is selected
- or use a different bus type (e.g. 8-bit multiplexed, where P1H may be used for I/O instead of P0H)
- or the spikes on P0H may be filtered with an application specific RC element,
- or do not perform an external bus access directly after an XBUS write access:
  this may be achieved by an instruction sequence which is executed in internal ROM/Flash/OTP, or internal RAM, or internal XRAM
  e.g. ATOMIC #3 ; to prevent PEC transfers which may access external memory
  instruction which writes to XBUS peripheral
  NOP
  NOP

PINS.1: Output Signal Rise Time

When a **non-multiplexed external bus without memory tristate waitstate** is used, and an external access (A) has been performed to a memory location whose address contains many 0s, followed by a memory access (B) to a location whose address contains many 1s, the following problem may occur:
1. the rising edge of RD# or WR# which is generated at the end of access (A) is relatively slow

2. the rising edge of ALE which is generated at the beginning of access (B) is relatively slow
In an application, we see the following potential problem, in particular at low temperatures:
Timing **t28/Address Hold after WR#** can no longer be guaranteed, since the address lines (P4, P1) may already fall below the high reference level before the WR# signal has reached its inactive high signal level. The high reference level used for timing measurements is 0.2Vcc+0.9 V (= 1.9V @ Vcc=5.0V).
Depending on the characteristic (e.g. input threshold) of an external peripheral (e.g. SRAM), write cycles may unintentionally overwrite external data.
For RD# cycles, there is no problem since data are always already latched internally by the controller before the RD# signal is externally driven high.

Workaround:
use a memory tristate waitstate (BUSCONx.MTTCx = 0) for all BUSCON registers which are used for external write cycles.

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 in Siemens 16-bit Microcontrollers" on

http://www.siemens.de/semiconductor/products/ics/34/pdf/ap292401.pdf

Deviation from Electrical- and Timing Specification:

The following table lists the deviations of the DC/AC characteristics from the specification in the C167CR-4RM Data Sheet 7.97 and C167SR/CR-L25M Data Sheet Addendum 1998-03

Problem short name	Parameter	Symbol	Limit Values		Unit	Test Condition
			min.	max.		
DC.IALEH.1	ALE active current	I_{ALEH}	**1000** instead of 500	-	µA	V_{OUT} = 2.4 V
DC.IRWL.1	RD#/WR# active current	I_{RWL}	**-600** instead of -500	-	µA	$V_{OUT} = V_{OLmax}$
DC.IP6L.1	Port 6 active current	I_{P6L}	**-600** instead of -500	-	µA	$V_{OUT} = V_{OLmax}$
DC.IP0L.1	Port 0 configuration current	I_{P0L}	**-110** instead of - 100	-	µA	$V_{OUT} = V_{OLmax}$

Problem short name	Parameter	Symbol	CPU Clock = 20 MHz		Variable CPU Clock 1/2TCL = 1 to 20 MHz		Unit
			min.	max.	min.	max.	
AC.t5.1	ALE high time	t5	**10**+ta instead of 15+ta	-	TCL-**15**+ta instead of TCL-10+ta	-	ns
AC.t12.1	WR#/WRH# low time (with RW-delay)	t12	**38**+tc instead of 40+tc	-	2TCL-**12**+tc instead of 2TCL -10+tc	-	ns
AC.t13.1	WR#/WRH# low time (no RW-delay)	t13	**63**+tc instead of 65+tc	-	3TCL-**12**+tc instead of 3TCL -10+tc	-	ns
AC.t38.1	ALE falling edge to CS#	t38	-**7**-ta instead of -4-ta	10-ta	-**7**-ta instead of -4-ta	10-ta	ns
AC.t48.1	RDCS#/WRCS# low time (with RW-delay)	t48	**38**+tc instead of 40+tc	-	2TCL-**12**+tc instead of 2TCL -10+tc	-	ns
AC.t49.1	RDCS#/WRCS# low time (without RW-delay)	t49	**63**+tc instead of 65+tc	-	3TCL-**12**+tc instead of 3TCL -10+tc	-	ns

Notes:

1) Pin **READY**# has an internal pull-up (all C167xx derivatives). This will be documented in the next revision of the Data Sheet.

2) Timing **t28**: Parameter description and test changed from 'Address hold after RD#/WR#' to 'Address hold after WR#'. It is guaranteed by design that read data are internally latched by the controller before the address changes.

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

- A/D Converter Characteristics:

ADCC.2.2: ADC Overload Current

During exceptional conditions in the application system an overload current I_{OV} can occur on the analog inputs of the A/D converter when $V_{AIN} > V_{dd}$ or $V_{AIN} < V_{ss}$. For this case, the following conditions are specified in the Data Sheet:

I_{OVmax} = | ±5 mA |

The specified total unadjusted error TUE_{max} = | ±2 LSB | is only guaranteed if overload conditions occur on maximum 2 not selected analog input pins and the absolute sum of input overload currents on all analog input pins does not exceed 10 mA. (It is also allowed to distribute the overload to more than 2 not selected analog input pins).

Due to an internal problem, the specified TUE value is only met for a **positive** overload current 0 mA ≤ I_{OV} ≤ +5 mA (all currents flowing into the microcontroller are defined as positive and all currents flowing out of it are defined as negative).

If the exceptional conditions in the application system cause a **negative** overload current, then the maximum TUE can be exceeded (depending on value of I_{OV}, R_{AIN}, R_{AREF} and R_{AGND}):

Problem Description in Detail:

1. **Overload Current at analog Channel AN0 - AN9 and AN11 - AN15**
   If a **negative** overload current I_{OV} occurs on analog input channel ANn (n≠10) then an additional current I_{AN} (crosstalk current) is caused at the neighbour channels ANn-1 and ANn+1. This behavior causes an additional unadjusted error AUE to the ADC result.

   Relation between I_{AN} and I_{OV}:

   $I_{ANn+1} = ovf\_1 * I_{OVn}$ (n≠10)

   $I_{ANn-1} = ovf\_1 * I_{OVn}$ (n≠10)

2. **Overload Current at digital Channel P7.7**
   A negative overload current I_{OV} at digital Port P7.7 causes an additional current I_{AN} at the analog input AN0. The relation between both channels is also defined by ovf_1:

   $I_{AN0} = ovf\_1 * I_{OVP7.7}$

3. **Overload Current at analog Channel AN10 and Influence to V_{AREF}**
   If an overload current I_{OV} occurs on analog input channel AN10 then an additional current I_{AREF} (crosstalk current) is caused at pin V_{AREF}.
   Depending on R_{AREF}, the internal resistance of the reference voltage, the crosstalk current I_{AREF} at pin V_{AREF} can cause an additional unadjusted error AUE to all other analog channels.
   In case $R_{AREF} \leq$ 195 Ohm [$R_{AREF} \leq$ ((LSB/2) / (I_{OVmax} * ovf_3)] the maximum possible additional error to all other channels is smaller than 0.5 LSB with the condition of I_{OVmax} = | ±5 mA| at AN10.

   Relation between I_{AREF} and I_{OV} at AN10:

   $I_{AREF} = ovf\_3 * I_{OV10}$ (I_{OV10} : overload current at AN10)

Note: The influence to the reference voltage V_{AREF} caused by I_{OV10} (shift of V_{AREF}) is maximum for $V_{AINn} = V_{AREF}$ and the influence is minimum for V_{AINn} = 0V (n ≠10). The condition $R_{AREF} \leq$ 195 Ohm and 0.5 LSB is calculated for the worst case at $V_{AINn} = V_{AREF}$.

4. Overload Current at analog Channel AN10 and Influence to V_{AGND}

If an overload current I_{OV} occurs on analog input channel AN10 then an additional current I_{AGND} (crosstalk current) is caused at pin V_{AGND}.
Depending on R_{AGND}, the resistance between V_{AGND} pin of the microcontroller and analog ground of the system, the crosstalk current I_{AGND} at pin V_{AGND} can cause an additional unadjusted error AUE to all other analog channels. In case $R_{AGND} \leq 72$ Ohm [$R_{AGND} \leq$ ((LSB/2) / (I_{OVmax} * ovf_2)] the possible additional error to all other channels is smaller than 0.5 LSB with the condition of I_{Ovmax} = | ±5 mA | at AN10.

Relation between I_{AGND} and I_{OV10}:

$$I_{AGND} = ovf\_2 * I_{OV10} \qquad (I_{OV10}: \text{overload current at AN10})$$

Note: The influence to the reference voltage caused by I_{OV10} (shifting the potential of V_{AGND} relative to system ground) is maximum for V_{AINn} = 0V and the influence is minimum for $V_{AINn} = V_{AREF}$ (n ≠10). The condition $R_{AGND} \leq 72$ Ohm and 0.5 LSB is calculated for the worst case at V_{AINn} = 0V. In standard systems the typical value for R_{AGND} = 0.1Ohm. In that case the ground shift error is **negligible**!

5. Overload Current at analog Channel AN10 (P5.10) and Influence to AN11

If $R_{AGND} \leq 72$ Ohm and $R_{AREF} \leq 195$ Ohm then the influence of a negative overload current I_{OV} to the analog input channel AN11 is:

$$I_{AN11} = ovf\_1 * I_{OV10}$$

6. Values of ovf_1, ovf_2 and ovf_3

Parameter	Symbol	Min	Max
Overload factor_1	ovf_1	- 0.0034	0
Overload factor_2	ovf_2	- 0.0068	0
Overload factor_3	ovf_3	- 0.0025	0

7. Effects on the Conversion Result and TUE

The effect on the conversion result and the TUE has to be calculated based on the crosstalk current and the impedance of the analog source R_{ASRC}. I_{ANn} causes an external voltage $U_{\Delta n}$ at the analog channel ANn (same principle for V_{AREF} and V_{AGND}) which is the reason for an additional unadjusted error **AUE** of the conversion result. This AUE can increase the specified total unadjusted error TUE (Specified value: TUE = ± 2 LSB). The voltage $U_{\Delta n}$ is nearly independent on the voltage value of the analog source.

$U_{\Delta n} = I_{ANn} * R_{ASRC}$
$U_{\Delta REF} = I_{AREF} * R_{AREF}$
$U_{\Delta GND} = I_{AGND} * R_{AGND}$
AUE = $U_{\Delta n}$ / 1 LSB [$U_{\Delta n}$ in mV and LSB in mV]
TUE = (± 2 LSB) ± AUE

Note: A negative overload current I_{OVn} decreases the analog signal voltage V_{AINn} (n≠10) and causes a negative AUE.
A negative overload current I_{OV10} (@ AN10) decreases the absolute value of V_{AREF} and causes a positive AUE.

8. Calculation Example

Assumed system values:

I_{OV4} = -300 µA negative overload current at P5.4
R_{ASRC} = 20 kOhm resistance of the external sensor at P5.5
V_{AREF} = 5 V 1 LSB = 4.9 mV
R_{AREF} and R_{AGND} has only to be considered for I_{OV} at AN10
R_{AGND} = 0.1 Ohm ➔ GND shift error is negligible
R_{AREF} = 500 Ohm ➔ V_{AREF} shift error is negligible

I_{AN5} = ovf_1 * I_{OV4}
I_{AN5} = (-3.4*E-3) * (- 300 µA)
I_{AN5} = 1.02 µA

$U_{\Delta 5}$ = I_{AN5} * R_{ASRC}
$U_{\Delta 5}$ = 1.02 µA * 20 kOhm
$U_{\Delta 5}$ = 20.4 mV

AUE = $U_{\Delta n}$ / 1 LSB
AUE = 20.4 mV / 4.9 mV
AUE = 4.2 LSB

Result: The negative overload current I_{OV4} of this system example can distort the real result of AN5 by an additional unadjusted error, -4.2 LSB ≤ AUE ≤ 0 LSB.
The TUE is in the range of -6.2 LSB ≤ TUE ≤ 2 LSB.

History List (since device step AB)

Functional Problems

Functional Problem	Short Description	Fixed in step
PWRDN.1	Execution of PWRDN Instruction while pin NMI# = high	
CPU.8	Jump instruction in EXTEND sequence	AC
CPU.9	PEC Transfers during instruction execution from Internal RAM	AC
CPU.11	Stack Underflow during Restart of Interrupted Multiply	AC
CPU.16	Data read access with MOVB [Rn], mem instruction to internal ROM	
CPU.17	Arithmetic Overflow by DIVLU instruction	
BUS.17	Spikes on CS# Lines after access with RDCS# and/or WRCS#	
BUS.18	PEC Transfers after JMPR Instruction	
RST.1	System Configuration via P0L.0 during Software/Watchdog Timer Reset	AC
RST.3	Bidirectional Hardware Reset	DA
ADC.8	CC31/ADC Interference	AC
ADC.10	Start of Standard Conversion at End of Injected Conversion	AC
ADC.11	Modifications of ADM field while bit ADST = 0	
X9	Read Access to XPERs in Visible Mode	
X12	P0H spikes after XPER write access and external 8-bit Non-multiplexed bus	
PINS.1	OUTPUT Signal Rise Time (DA-step only)	

AC/DC Deviations

AC/DC Deviation	Short Description	Fixed in step
DC.VOL.1	Output low voltage (Port0/1/4, ALE, RD#, WR#, ...) test condition 1.6mA (AC-step only)	DA
DC.IALEH.1	ALE active current 1000µA	
DC.IRWL.1	RD#/WR# active current –600µA	
DC.IP6L.1	Port 6 active current –600 µA	
DC.IP0L.1	Port 0 configuration current –110µA	1)
DC.HYS.1	Input Hysteresis 300mV(restriction not effective in production test)	
AC.t5.1	ALE high time TCL-15ns	
AC.t12.1	WR#/WRH# low time (with RW-delay) 2TCL-12ns	
AC.t13.1	WR#/WRH# low time (no RW-delay) 3TCL-12ns	
AC.t38.1	ALE falling edge to CS# -7ns	
AC.t48.1	RDCS#/WRCS# low time (with RW-delay) 2TCL-12ns	
AC.t49.1	RDCS#/WRCS# low time (no RW-delay) 3TCL-12ns	
ADCC.2.2	ADC Overload Current	

Note 1): problem not included in AC-step

In addition to the description in the C167 Derivatives User's Manual V2.0, the following feature enhancements have been implemented in the C167CR-4RM:

Incremental position sensor interface

For each of the GPT1 timers T2, T3, T4 of the GPT1 unit, an additional operating mode has been implemented which allows to interface to incremental position sensors (A, B, Top0). This mode is selected for a timer Tx via TxM = 110b in register TxCON, x = (2, 3, 4). Optionally, the contents of T5 may be captured into register CAPREL upon an event on T3. This feature is selected via bit CT3 = 1 in register T5CON.10

Compatibility with previous versions:

In previous versions (e.g. C167CR-LM BA-step), both of the settings (TxM = 110b, T5CON.10 = 1) were reserved and should not be used. Therefore, systems designed for previous versions will also work without problems with the C167CR-4RM.

Oscillator Watchdog

The C167CR-4RM provides an Oscillator Watchdog (OWD) which monitors the clock at XTAL1 in direct drive mode. In case of clock failure, the PLL Unlock/OWD Interrupt Request Flag (XP3IR) is set and the internal CPU clock is supplied with the PLL basic frequency. This feature can be disabled by a low level on pin Vpp/OWE. See also C167CR-4RM Data Sheet 7.97.

Bidirectional Reset

The C167CR-4RM allows to indicate an internal watchdog timer or software reset on the RSTIN# pin which will be driven low for the duration of the internal reset sequence. This option is selectable by software via bit BDRSTEN/SYSCON.3. After reset, the bidirectional reset option is disabled (BDRSTEN/SYSCON.3 = 0). See also C167CR-4RM Data Sheet 7.97. Beginning with the **AC-step** of the C167CR-4RM, RSTIN# will also be driven low for the duration of the internal reset sequence when this reset was initiated by an **external HW reset** signal on pin RSTIN#.

Please note also the following functional difference to the C167CR-LM BA-step:

XBUS Peripheral Enable Bit XPEN/SYSCON.2

In the C167CR-4RM, bit SYSCON.2 is a general XBUS Peripheral Enable bit, i.e. it controls both the XRAM **and the CAN module**.

Compatibility with previous versions:

When bit SYSCON.2 = 0 (default after reset) in the C167CR-4RM, and an access to an address in the range EF00h ... EFFFh is made, either an external bus access is performed (if an external bus is enabled), or the Illegal Bus Trap is entered. In previous versions (e.g. C167CR-LM BA-step), the CAN module was accessed in this case.

Systems where bit SYSCON.2 was set to '1' before an access to the CAN module in the address range EF00h ... EFFFh was made will also work without problems with the C167CR-4RM.

Application Support Group, Munich