Errata Sheet

July 16, 1999 / Release 1.6

Device: **SAB-C164CI-8E**

Stepping Code / Marking: **ES-BC**

Package: **MQFP-80**

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: C164CI Data Sheet 02.98
User's Manual: C164CI User's Manual V1.1 1998-08
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.1.5 for devices with stepping code/marking ES-BC:

- The following problem has been included in the Errata Sheet:

- Programming Voltage V_{PP} and Supply Voltage V_{DD} (OTP.8) – *updated version*

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 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 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.

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** (e.g. 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]

Note:

- the Keil C166 Compiler from V3.10 on has been extended by the directive FIXROM which avoids accesses to 'const' objectes via the instruction MOVB [Rn], mem.
- the Tasking Compiler provides a workaround for this problem from release 6.0r2 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 occured.

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 significat 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.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:

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 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'.

CAPCOM.2: Wakeup from Idle Mode by CAPCOM2 channels CC19 or CC27

When the CPU is in idle mode, and an interrupt request from CAPCOM2 channels CC19 or CC27 is used to wake the CPU up from idle mode, interrupt request flag CC18IR may be set erroneously in addition to CC19IR, or CC26IR may be set erroneously in addition to CC27IR.

Workarounds:

1) when using CC19 or CC27 for wakeup from idle mode, do not use the interrupts of CC18 or CC26.

2) for systems with a lower interrupt rate, also the following procedure may be applicable:
always assign a higer interrupt priority to channel CC19 (CC27) than to CC18 (CC26). After wakeup from idle, the interrupt routine of channel CC19 (CC27) should check whether flag CC18IR (CC26IR) is set. Depending on the contents of register CC18 (CC26) and the contents of the associated timer, the validity of the CC18 (CC26) interrupt request may then be determined by software, and in case CC18IR (CC26IR) has been set erroneously it should be cleared.

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.

CAPCOM6.3: Write Access to CAPCOM6 Registers

While the contents of T12 or T13 is 0 or equal to the programmed period value, write accesses to registers with shadow latches should be avoided if bit STE12 (dedicated to T12) or STE13 (dedicated to T13) are set.

Such write access can lead to a second data transfer to the corresponding register itself.

Workaround:

Write accesses should be performed after the corresponding period interrupt (T12IR or T13IR) has occurred.

CAPCOM6 Application Hint: Compare value lower than Offset value

The compare value of all three channels should not be lower than the selected offset value in order to avoid undesired switching of the outputs into active state. This is particularly important for motor control applications.

See Application Note AP0823 'C504 Important applications hints for dead time generation with the Capture/Compare Unit' (the peripheral CCU discussed in this ApNote is functionally identical to CAPCOM6 of the C164CI):

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

RTC.2: Read RTC Count Registers

Due to a wrong read protection of the RTC registers during counting state, an undefined RTC value can be read. The RTC value is only valid if the registers are not toggling during read access. There are two possible procedures for reading a correct RTC value:

Workarounds:

1. Compare of 4 RTC values
   - Read RTC registers four times
   - Compare these values
   - If two successive read values are equal, this value is valid
2. Read RTC registers during RTC interrupt
   - After two NOPs on the beginning of the RTC interrupt service routine, the RTC value is valid and can be read.
   - The RTC value is stable for at least 240 oscillator cycles.

OTP.7: Verify in CHM when Bootstrap Loader Mode is selected

The verification of programmed data is not possible during programming of the OTP in CPU host mode (CHM) when using CHM in combination with the bootstrap loader mode (selected with P0L.4 low during reset). The OTP´s programming is not influenced by this problem.

Workaround:

The OTP contents can be verified after leaving the CPU host mode by reading the contents via the normal ROM-bus access.

The OTP contents can be verified in CHM when booting from external memory.

OTP.8: Programming Voltage V_{PP} and Supply Voltage V_{DD}

To guarantee a proper internal OTP programming the tolerances for supply voltage V_{DD} and programming voltage V_{PP} have to be reduced.

The following values have been successfully tested and can be recommended:

Supply Voltage V_{DD}	Programming Voltage V_{PP}
4.35V – 4.65V	11.35V - 11.65V
4.85V – 5.15V	11.8 – 12.1V

Application Hints:

- Devices formerly programmed with these recommended values can be used without restriction.
- Software for OTP programming tools should be adapted to one of these recommended values (V_{DD} = 4.85V – 5.15V and V_{PP} = 11.8 – 12.1V can still be used after the problem is fixed in the future).

ADP.1: Oscillator in Adapt Mode

In adapt mode the real time clock (RTC) and the main oscillator is running. This behaviour should not present an a problem, since during emulation via a clip-on adapter the oscillator output pin XTAL2 of the target and the emulation devices (bondout chip) should be disconnected anyway.

The main difference in the behaviour of XTAL2 during adapt mode of different C167 derivatives is as follows:

C167/S/SR/CR:	XTAL1 = low	-->	XTAL2 = high
	XTAL1 = high	-->	XTAL2 = Z (high impedance)
C164CI:	XTAL1 = low	-->	XTAL2 = high
	XTAL1 = high	-->	XTAL2 = low

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 AP2924 " Interrupt Register behaviour of the CAN module in Siemens 16-bit Microcontrollers" on

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

Deviations from Electrical- and Timing Specification:

The following table lists the deviations of the DC/AC characteristics from the specification in the C164CI Data Sheet 2.98

- DC.VDD.1 Vcc = **4.4** .. 5.5 V instead of 4.25 .. 5.5 V

Problem short name	**Parameter**	**Symbol**	**Limit Values**		**Unit**	**Test Condition**
			min.	max.		
DCAH.2	ALE active current	I_{ALEH}	**650** instead of 500	-	µA	V_{OUT} = 2.4 V
DCRL.2	RD#/WR# active current	I_{RWL}	**-650** instead of -500	-	µA	$V_{OUT} = V_{OLmax}$
DCP4L.2	Port 4 active current (during Reset or Adapt mode)	I_{P4L}	**-650** instead of -500	-	µA	$V_{OUT} = V_{OLmax}$
DCP0L.2	Port 0 configuration current	I_{P0L}	**-120** 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.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 (with RW-delay)	t49	**63**+tc instead of 65+tc	-	3TCL-**12**+tc instead of 3TCL -10+tc	-	ns

.

ADCC.2.3: 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:
$I_{OVmax} = | \pm 5 \text{ mA} |$
The specified total unadjusted error $TUE_{max} = | \pm 2 \text{ LSB} |$ 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 can be exceeded if an overload current occurs at P5.3 (AN3). If the exceptional conditions in the application system cause an overload current, then the maximum TUE depends on value of I_{OV}, R_{AREF} and R_{AGND}.

Problem Description in Detail:

1. Overload Current at analog Channel AN3 and Influence to V_{AREF}

If an overload current I_{OV} occurs on analog input channel AN3 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 $R_{AREF} \leq 40$ 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 $I_{OVmax} = | \pm 5 \text{ mA} |$ at AN3.

Relation between I_{AREF} and I_{OV} at AN3

$$I_{AREF} = ovf\_3 * I_{OV3} \qquad (I_{OV3:} \text{ overload current at AN3})$$

Note: The influence to the reference voltage V_{AREF} caused by I_{OV3} (shift of V_{AREF}) is maximum for $V_{AINn} = V_{AREF}$ and the influence is minimum for $V_{AINn} = 0V$ ($n \neq 3$). The condition $R_{AREF} \leq 40$ Ohm and 0.5 LSB is calculated for the worst case at $V_{AINn} = V_{AREF}$.

2. Overload Current at analog Channel AN3 and Influence to V_{AGND}

If an overload current I_{OV} occurs on analog input channel AN3 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 $R_{AGND} \leq 15$ 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 $I_{OVmax} = | \pm 5 \text{ mA} |$ at AN3.

Relation between I_{AGND} and I_{OV3}:

$$I_{AGND} = ovf\_2 * I_{OV3} \qquad (I_{OV3}: \text{ overload current at AN3})$$

Note: The influence to the reference voltage caused by I_{OV3} (shifting the potential of V_{AGND} relative to system ground) is maximum for $V_{AINn} = 0V$ and the influence is minimum for $V_{AINn} = V_{AREF}$ ($n \neq 3$). The condition $R_{AGND} \leq 15$ Ohm and 0.5 LSB is calculated for the worst case at $V_{AINn} = 0V$. In standard systems the typical value for $R_{AGND} = 0.1$Ohm. In that case the ground shift error is **negligible**!

3. Values of ovf_2 and ovf_3

Parameter	Symbol	Min	Max
Overload factor_2	ovf_2	- 0.03	0
Overload factor_3	ovf_3	- 0.01	0

These Values are the absolute **maximum values measured in the lab and not tested!**

4. 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 ($I_{AREF\ and}\ I_{AGND}$) and the impedance of the sources R_{AGND} and R_{AREF}. $I_{AREF\ and}\ I_{AGND}$ cause an external voltage $U\Delta$ at the reference voltage 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).

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

5. Calculation Example

Assumed system values:

I_{OV3} = 300 µA positive overload current at AN3 (P5.3)
R_{AREF} = 3400 Ohm resistance of the reference voltage V_{AREF}
V_{AREF} = 5 V 1 LSB = 4.9 mV
R_{AGND} = 0.1 Ohm ➔ GND shift error is negligible
V_{AINn} : 5V, 2.5V, 0V analog input voltage at ANn with n ≠ 3

I_{AREF} = ovf_3 * I_{OV3} crosstalk current at reference voltage
I_{AREF} = - 0.01 * 300 µA
I_{AREF} = - 3 µA

$U\Delta REF$ = I_{AREF} * R_{AREF} reference voltage shift
$U\Delta REF$ = - 3 µA * 3400 Ohm
$U\Delta REF$ = -10.2 mV

$U\Delta REF@VAIN$ = (V_{AIN} * $U\Delta REF$) / V_{AREF} relative reference voltage shift at V_{AIN}
AUE = $U\Delta REF@VAIN$ / 1 LSB additional unadjusted error
AUE = (V_{AIN} * $U\Delta REF$) / (V_{AREF} * 1 LSB)

➔ V_{AINn} = 5 V:
AUE = (V_{AIN} * $U\Delta REF$) / (V_{AREF} * 1 LSB)
AUE = (5V * (-10.2mV)) / (5V * 4.9 mV)
AUE = -2.1 LSB

➔ V_{AINn} = 2.5 V:
AUE = (V_{AIN} * $U\Delta REF$) / (V_{AREF} * 1 LSB)
AUE = (2.5V * (-10.2mV)) / (5V * 4.9 mV)
AUE = -1 LSB

➔ V_{AINn} = 0 V:
AUE = 0LSB

Result:

The overload current I_{OV3} of this system example can distort the real result of AN0 – AN2 and AN4 – AN7 by an additional unadjusted error,
-2.1 LSB ≤ AUE ≤ 0 LSB. The TUE is in the range of -4.1LSB ≤ TUE ≤ 2LSB.

History List (since device step ES-BB)

Functional Problems

Functional Problem	Short Description	Fixed in step
PWRDN.1	Execution of PWRDN Instruction while pin NMI# = high	
CPU.16	Data read access with MOVB [Rn], mem instruction to internal ROM/Flash/OTP	
CPU.17	Arithmetic Overflow by DIVLU instruction	
BUS.18	PEC transfers after JMPR	
RST.4	Power-on Reset	ES-BC
ADC.11	Modifications of ADM field while bit ADST = 0	
ADP.1	Oscillator in Adapt Mode	
CAPCOM6.3	Write Access to CAPCOM6 Registers	
RTC.2	Read RTC count registers	
OTP.7	Verify in CHM when Bootstrap Loader Mode is selected	
OTP.8	Programming Voltage V_{PP} and Supply Voltage V_{DD}	
RST.4	Power on Reset	
X9	Read Access to XPERs in Visible Mode	

AC/DC Deviations

AC/DC Deviation	Short Description	Fixed in step
DC.VDD.1	Device Supply Voltage VDD min. 4.4 V	
DCAH.2	ALE active current 650 µA	
DCRL.2	RD#/WR# active current – 650 µA	
DCP4L.2	P4 active current – 650 µA	
DCP0L.2	P0 configuration current –120 µA	
t48	RDCS#/WRCS# low time (with RW-delay) 2TCL-12	
t49	RDCS#/WRCS# low time (without RW-delay) 3TCL-12	
ADCC.2.3	ADC Overload Current	

Application Support Group, Munich