SPRU189C Errata -

1998 TMS320C62x/C67x CPU and Instruction Set Reference Guide

Last Update: July 24, 1998

Page 2-11 A note should be added about the Data Cache Control (DCC) field in the CSR: Some C62x/C67x devices do not have a data cache (i.e. C6201 and C6701).

Page 2-15 The Rmode.L2 and Rmode.L1 Value 00 description for the **FADCR** should read: Round toward nearest.

Page 2-19 The Rmode.L2 and Rmode.L1 Value 00 description for the **FMCR** should read: Round toward nearest.

Page 2-20 The following should be added to section 2.6 and 2.7: The SAT, OVER, UNDER, INEX, INVAL, DENn, NANn, INFO, UNORD, and DIV0 status bits will not be modified by an instruction if the condition of the instruction is false.

Page 3-2 The description for slsb16 and smsb16 should read:

slsb16	Signed 16 LSB of **register**
smsb16	Signed 16 MSB of **register**

Page 3-3 The description for ulsb16, umsb16, xslsb16 and xulsb16 should read:

ulsb16	Signed 16 LSB of **register** that can optionally use cross path
umsb16	Unsigned 16 LSB of **register** that can optionally use cross path
xslsb16	Signed 16 LSB of **register** that can optionally use cross path
xulsb16	Unsigned 16 LSB of **register** that can optionally use cross path

Page 3-4 In table 3-2, under the .S unit, the SHRL instruction should read **SSHL**.

Page 3-4 In table 3-2, the **ADDU** instruction should not be listed for the .D or .S units; the **ADDU** is only available on the .L units.

Page 3-5 In table 3-3, the **ADDU** instruction is only available on the .L units; the table should not state that it is available on the .D or .S units.

Page 3-12 A note should be added to the load instruction type in table 3-5 stating: The write on cycle i+4 uses a separate write port than the write on other .D unit instructions.

Page 3-34 The opcode diagram is incorrect for the **ADDAB/ADDAH/ADDAW** instruction. It should be the 3rd diagram on page 3-10 (Operations on the .D unit) rather than the 6th diagram on page 3-10 (Operations on the .S unit).

Page 3-35 In example 2, the **AMR** value should be **0002 0001h** for “before instruction” and “1 cycle after instruction”.

Page 3-38 The correct opcode for the first version of the **AND** instruction should be **1111011** rather than 111001.

Page 3-40 The second note for the **B label** instruction should be changed to read as follows: Execute packets in the delay slots of a branch cannot be interrupted. This is true regardless of whether the branch is taken.

Page 3-42 The second note for the **B src2** instruction should be changed to read as follows:
Execute packets in the delay slots of a branch cannot be interrupted. This is true regardless of whether the branch is taken.

Page 3-44 The third note for the **B IRP** instruction should be changed to read as follows:
Execute packets in the delay slots of a branch cannot be interrupted. This is true regardless of whether the branch is taken.

Page 3-46 The third note for the **B NRP** instruction should be changed to read as follows:
Execute packets in the delay slots of a branch cannot be interrupted. This is true regardless of whether the branch is taken.

Page 3-49 A note should be added to the **CLR** instruction stating:
Only the 10 LSBs are valid for the register version. If the 22 MSBs are non-zero, the result is invalid.

Page 3-55 A note should be added to the **CMPGT(U)** instruction stating:
Only the 4 LSBs are valid in the 5-bit cst field when the ucst4 operand is used. If the MSB of the cst field is non-zero, the result is invalid.

Page 3-56 The first column of the table is incorrect for the first four opfields. The first operand should be called src1 and the second operand should be called src2 for the opfields 1010111, 1010110, 1010101, and 1010100. All other columns should remain the same.

Page 3-59 A note should be added to the **EXT** instruction stating:
Only the 10 LSBs are valid for the register version. If the 22 MSBs are non-zero, the result is invalid.

Page 3-62 The first opcode diagram for the **EXTU** instruction (constant) has a misprint; the field labeled *src1* should read *csta*.

Page 3-62 A note should be added to the **EXTU** instruction stating:
Only the 10 LSBs are valid for the register version. If the 22 MSBs are non-zero, the result is invalid.

Page 3-71 The second paragraph of the **LDB(U)/LDH(U)/LDW** description should read:
The offset, ucst15**,** is scaled by a left shift of 0, 1, or 2 for LDB(U), LDH(U), and LDW, respectively. After scaling, ucst15 is added to **baseR (no subtraction is supported).** The result …

Page 3-107 A note should be added to the **SET** instruction stating:
Only the 10 LSBs are valid for the register version. If the 22 MSBs are non-zero, the result is invalid.

Page 3-109 The operand types for the third (opfield 010011) and sixth (opfield 010010) versions of the **SHL** instruction should be changed to xuint, uint, and ulong for the src2, src1, and dst operands, respectively.

Page 3-109 A note should be added to the **SHL** instruction stating:
Only the 6 LSBs are valid for the register version. If the 26 MSBs are non-zero, the result is invalid.

Page 3-111 A note should be added to the **SHR** instruction stating:
Only the 6 LSBs are valid for the register version. If the 26 MSBs are non-zero, the result is invalid.

Page 3-113 A note should be added to the **SHRU** instruction stating:
Only the 6 LSBs are valid for the register version. If the 26 MSBs are non-zero, the result is invalid.

Page 3-118 A note should be added to the **SSHL** instruction stating:

For the register version, the result is invalid if the shift amount is greater than 31.

Page 3-125 In example 4 of the **STB/STH/STW** instruction the value of A10 for “1 cycle after instruction” and “3 cycles after instruction” should be **0000 00F8h**, and the label mem 9Ch should read **mem F8h**.

Page 3-126 The second paragraph of the **STB/STH/STW** instruction description should read:
The offset, ucst15, is scaled by a left shift of 0, 1, or 2 for STB, STH, and STW, respectively. After scaling, ucst15 is added to **baseR (no subtraction is supported).** The result …

Page 3-135 The header for the **SUB2** instruction should read:
Two 16-Bit Integer **Subtractions** on Upper and Lower Register Halves

Page 4-3 The meaning for **sp** in table 4-1 should read:
Single-precision floating point register value.

Page 4-3 The meaning for **xsp** in table 4-1 should read:
Single-precision floating point register value that can optionally use cross path.

Page 4-6 The last sentence of the first paragraph should be changed to read as follows:
The register pair syntax always places the odd register first, a colon, followed by the even register (that is, A1:A0, B1:B0, A3:A2, B3:B2, etc).

Page 4-11 A note should be added to the load instruction type in table 4-9 stating:
The write on cycle i+4 uses a separate write port than other .D unit instructions.

Page 4-11 The second paragraph should be changed as follows:
The double-precision floating-point addition, subtraction, multiplication, compare, and the 32-bit integer multiplication instructions also have a functional unit latency that is greater than 1. **Functional unit latency is equivalent to the number of cycles that the instruction uses the functional unit read ports**. For example, the ADDDP instruction …

Page 4-12 The first paragraph should read as follows:
If an instruction has a multicycle functional unit latency, it locks the functional unit for the necessary number of cycles. Any new instruction dispatched to that functional unit during this locking **period will cause undefined results**. If an instruction with a multicycle functional unit latency has a condition that is evaluated as false during E1, it **still locks** the functional unit for subsequent cycles.

Page 4-12 The second paragraph should read as follows:
An instruction of the following types scheduled on cycle i has the following constraints:

Page 4-12 The forth paragraph should read as follows:
An instruction of the following types scheduled on cycle i and using a cross path to read a source, has the following constraints:

Page 4-12 The fifth paragraph should read as follows:
Other hazards exist because instructions have **varying numbers of delay slots, and because they need the functional unit read and write ports for varying numbers of cycles**. A read or write or write hazard occurs when two instructions on the same functional unit attempt to read or write, respectively, to the register file on the same cycle.

Page 4-13 The first paragraph should read as follows:
An instruction scheduled on cycle i has the following constraints:

Page 4-51 The Opcode should show the dst/src opfield as the dst field.

Page 4-51 A sentence should be added to the end of the second paragraph stating the following:
The dst field must always be an even value since LDDW loads register pairs. Therefore, bit 23 must always be zero.

Page 4-52 The last sentence of the first paragraph should be changed to read as follows:
The register pair syntax always places the odd register first, a colon, followed by the even register.

Page 4-62 The 7th note should be removed since **RCPDP** will never perform rounding.

Page 4-64 The 7th note should be removed since **RCPSP** will never perform rounding.

Page 4-66 The 3rd note for **RSQRDP** should be changed to read:
If src2 is a negative, nonzero, nondenormalized number, NaN_out is placed in dst and the **INVAL bit is** set.

Page 4-66 The 6th note for **RSQRDP** should be changed to read:
If src2 is **positive** infinity, then **positive** zero is placed in dst.

Page 4-66 The 7th note should be removed since **RSQRDP** will never perform rounding.

Page 4-69 The 3rd note for **RSQRSP** should be changed to read:
If src2 is a negative, nonzero, or nondenormalized number, then NaN_out is placed in dst and the **INVAL bit is** set.

Page 4-69 The 6th note for **RSQRSP** should be changed to read:
If src2 is **positive** infinity, then **positive** zero is placed in dst.

Page 4-69 The 7th note should be removed since **RSQRSP** will never perform rounding.

Page 6-16 The last sentence of the first paragraph should read as follows:
If an instruction has a multicycle functional unit latency, it locks the functional unit for the necessary number of cycles. Any new instruction dispatched to that functional unit during this locking **period will cause undefined results**. If an instruction with a multicycle functional unit latency has a condition that is evaluated as false during E1, it **still locks** the functional unit for subsequent cycles.

Page 6-16 The second paragraph should read as follows:
An instruction of the following types scheduled on cycle i has the following constraints:

Page 6-17 The first paragraph should read as follows:
An instruction of the following types scheduled on cycle i and using a cross path to read a source, has the following constraints:

Page 6-17 The second paragraph should read as follows:
Other hazards exist because instructions have **varying numbers of delay slots, and because they need the functional unit read and write ports for varying numbers of cycles**. A read or write or write hazard occurs when two instructions on the same functional unit attempt to read or write, respectively, to the register file on the same cycle.

Page 6-17 The third paragraph should read as follows:
An instruction scheduled on cycle i has the following constraints:

Page 7-28 The following section should be added:
It is good practice to disable interrupts when modifying a control register since an ISR may modify the control register value between the time that the program loads the old value and stores the new value. The below method will ensure that an ISR does not modify the CSR between the 2 MVC instructions since interrupts are disabled during the 5 delay slots of the branch instruction.

```
        b       label
        mvc     CSR, B0         ; B0 = CSR
        or      1, B0, B0       ; modify B0
        mvc     B0, CSR         ; CSR = B0
        nop     2
label
```

Without disabling the interrupts the following could occur:

```
        mvc     CSR, B0         ; B0 = CSR (Assume PCC = 0)
        or      1, B0, B0       ; modify B0
                                ; interrupt occurs and the ISR sets PCC = 1 in the CSR
        mvc     B0, CSR         ; CSR = B0
                                ; writing the old value of PCC = 0 to the CSR
```

Page 7-28 The following section should be added:
If the SAT bit in the CSR is zero, the MVC instruction can not set it to one. This could be a problem if an ISR saves a CSR value with SAT=1, sets SAT=0, and later tries to restore the saved CSR value.

```
; assume SAT = 1 before CSR value was saved to B0
; assume that SAT = 0 in the CSR
        mvc     B0,CSR          ; restore previous CSR value
```

The above code will not set SAT back to one. The following is a work-around.

```
; assume SAT = 1 before CSR value was saved to B0
; assume that SAT = 0 in the CSR
        extu    B0,22,31,B1     ; b1 = previous SAT value
        shl     B1,30,B1        ; if prev SAT bit set, B1 = 0x40000000
        sadd    B1,B1,B1        ; if(b1) set SAT bit
        mvc     B0,CSR          ; restore previous CSR value
```