Manual Update Sheet

DATE: August 1, 1997

Document Being Updated: ***TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set***

Literature Number Being Updated: SPRU172

Manual Included in a Kit: Yes

This Manual Update Sheet (SPRZ118) ships with the *TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set*.

Updates within paragraphs appear in a **bold typeface**.

Page: **Change or Add:**

viii Deleted *Trademarks* from page header.

1–3 Changed hi (A) meaning in Table 1–1, *Instruction Set Symbols and Abbreviations*:

Symbol	Meaning
hi(A)	High part of accumulator A (bits **31**–16)

1–5 Added following symbols to Table 1–1, *Instruction Set Symbols and Abbreviations*:

Symbol	Meaning
– – SP	Stack pointer value is decremented by 1
+ + SP	Stack pointer value is incremented by 1
+ + PC	Program counter value is incremented by 1

1–8 Added following symbol to Table 1–4, *Operators Used in Instruction Set*:

Symbols	Operators	Evaluation
< < <	Logical left shift	Left to right

2–4 Changed following Expression in Table 2–4, *Multiply-Accumulate and Multiply-Subtract Instructions*:

Syntax	Expression
MACR Smem, src	**src** = rnd(src + T * Smem)

2–7 Changed following Expressions in Table 2–6, *Application-Specific Instructions*:

Syntax	Expression
LMS Xmem, Ymem	B = B + Xmem * Ymem A = **A + Xmem << 16** + 2^{15}
POLY Smem	B = Smem << 16 A = rnd(A**(32–16)** * T + B)
SQDST Xmem, Ymem	B = B + A(32–16) * A(32–16) A = (Xmem **–** Ymem) << 16

Page: **Change or Add:**

2–11 Changed Expressions in Table 2–13, *Call Instructions*:

Syntax	Expression
CALA[D] src	– –SP, **PC + 1[3¶] = TOS**, PC = src(15–0)
CALL[D] pmad	– –SP, **PC + 2[4¶] = TOS**, PC = pmad(15–0)
CC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then – –SP, **PC + 2[4¶] = TOS**, PC = pmad(15–0)
FCALA[D] src	– –SP **,PC + 1[3¶] = TOS**, PC = src(15–0), XPC = src(22–16)
FCALL[D] extpmad	– –SP, **PC + 2[4¶] = TOS**, PC = pmad(15–0), XPC = pmad(22–16)

Changed Expressions in Table 2–14, *Interrupt Instructions*:

Syntax	Expression
INTR K	– –SP, **++ PC = TOS**, PC = IPTR(15–7) + K << 2, INTM = 1
TRAP K	– –SP, **++ PC = TOS**, PC = IPTR(15–7) + K << 2

2–12 Changed Expressions in Table 2–15, *Return Instructions*:

Syntax	Expression
FRET[D]	XPC = **TOS, ++ SP, PC = TOS, ++SP**
FRETE[D]	XPC = **TOS, ++ SP, PC = TOS, ++SP**, INTM = 0
RC[D] cond [, cond [, cond]]	if (cond(s)) then PC = **TOS, ++SP**
RET[D]	PC = **TOS, ++SP**
RETE[D]	PC = **TOS, ++SP**, INTM = 0
RETF[D]	PC = RTN, **++SP**, INTM = 0

Changed following Expression in Table 2–16, *Repeat Instructions*:

Syntax	Expression
RPTB[D] pmad	Repeat block, RSA = PC + 2[4¶], REA = pmad, **BRAF = 1**

Page: **Change or Add:**

2–13 Changed following Expressions in Table 2–17, *Stack-Manipulating Instructions*:

Syntax	Expression
POPD Smem	Smem = **TOS, ++SP**
POPM MMR	MMR = **TOS, ++SP**
PSHD Smem	– –SP, **Smem = TOS**
PSHM MMR	– –SP, **MMR = TOS**

3–2 Deleted following phrase from *n* definition:

(where n $>$ 2 to fill the pipeline)

4–7 Added Example 4:

Example 4

```
ADD *AR2+, *AR2–, A    ;after accessing the operands, AR2
                       ;is incremented by one.
```

Example 4 shows the same auxiliary register (AR2) with different addressing modes specified for both operands. The mode defined by the Xmod field (*AR2+) is used for addressing.

4–13 Changed syntax in Example 2:

Example 2 `ANDM #0101h, 4; DP = 0`

4–17 Changed contents of PC (after instruction) in Example 3 to 1003.

4–18 Removed underscore character from Syntax.

4–18/19 Changed Description:

This instruction passes control to the program-memory address (*pmad*) if the specified condition(s) is met. The two 1-word instructions or the one 2-word instruction following the branch instruction is fetched from program memory. If the condition(s) is met, the two words following the instruction are flushed from the pipeline and execution begins at *pmad*. If the condition(s) is not met, the PC is incremented by 2 and the two words following the instruction are executed.

If the branch is delayed (specified by the D suffix), the two 1-word instructions or the one 2-word instruction is fetched from program memory and executed. The two words following the delayed instruction have no effect on the conditions being tested. If the condition(s) is met, execution continues at *pmad*. If the condition(s) is not met, the PC is incremented by 2 and the two words following the delayed instruction are executed.

4–22 Changed syntaxes in Examples 1 and 2:

Example 1 `BITF 5, 00FFh`

Example 2 `BITF 5, 0800h`

Page:	Change or Add:
4–28	Added following paragraph after Example 2: After the memory location has been ANDed with 4444h, the program continues executing from location 1000h.
4–30	Changed Description: This instruction passes control to the program-memory address (*pmad*) if the specified condition(s) is met. The two 1-word instructions or the one 2-word instruction following the call instruction is fetched from program memory. If the condition(s) is met, the two words following the instruction are flushed from the pipeline and execution begins at *pmad*. If the condition(s) is not met, the PC is incremented by 2 and the two words following the instruction are executed. If the call is delayed (specified by the D suffix), the two 1-word instructions or the one 2-word instruction is fetched from program memory and executed. The two words following the delayed instruction have no effect on the conditions being tested. If the condition(s) is met, execution continues at *pmad*. If the condition(s) is not met, the PC is incremented by 2 and the two words following the delayed instruction are executed.
4–31	Added following paragraph after Example 2: After the memory location has been ANDed with 4444h, the program continues executing from location 1000h.
4–37	Changed Words and Cycles descriptions from Smem to Lmem.
4–38	Changed Example 3 contents of AR3 (after instruction) to 00FF.
4–39	Changed Words and Cycles descriptions from Smem to Lmem.
4–42	Changed Words and Cycles descriptions from Smem to Lmem.
4–43	Changed Words and Cycles descriptions from Smem to Lmem.
4–45	Changed Words and Cycles descriptions from Smem to Lmem.
4–46	Changed Example 1 contents of C (after instruction) to 0.
4–47	Changed Words and Cycles descriptions from Smem to Lmem.
4–48	Changed Words and Cycles descriptions from Smem to Lmem.
4–49	Changed Example 2 contents of C (after instruction) to 1.
4–50	Changed Words and Cycles descriptions from Smem to Lmem.
4–56	Changed Example 1 contents of SP (after instruction) to 110F and Data Memory address 1109h to 110Fh. Changed Example 2 contents of PC (after instruction) to 2000, contents of SP (after instruction) to 110F, and Data Memory address 1109h to 110Fh.

Page:	Change or Add:
4–58	Changed Example 1 contents of SP (after instruction) to 110F and Data Memory address 1109h to 110Fh. Changed Example 2 contents of SP (after instruction) to 110F, contents of Data Memory 1110h (after instruction) to 3005, and Data Memory address 1109h to 110Fh. Added following paragraph after Example 2: After the memory location has been ANDed with 4444h, the program continues executing from location 1000h.
4–72	Changed syntax in Example 6: Example 6 `LD **0**, ASM`
4–73	Changed Execution: (MMR) → dst**(15–0)** **00 0000h → dst(39–16)**
4–78	Changed Description: This instruction loads the data-memory value *Smem* shifted left 16 bits into the high part of *dst* (bits 31–16). *Smem* is rounded by adding 2^{15} to this value and clearing the **15** LSBs (**14**–0) of the accumulator to 0. Bit 15 of the accumulator is set to 1.
4–80	Changed Execution: **(A)** + **(Xmem)** << 16 + 2^{15} → A (B) + (Xmem) × (Ymem) → B Changed Description: This instruction executes the least mean square (LMS) algorithm. The dual data-memory operand *Xmem* is **shifted left 16 bits and added to accumulator A**. The result is rounded by adding 2^{15} to the high part of the accumulator (bits 31–16). The result is stored in accumulator A. In parallel, *Xmem* and *Ymem* are multiplied and the result is added to accumulator B. *Xmem* does not overwrite T; therefore, T always contains the error value used to update coefficients.
4–85	Changed syntax 2 Execution: 2: (T) × (A(32–16)) + (src) → **dst**
4–93	Changed syntax in Example 2: Example 2 `MAR **\*AR0–**` Changed Example 3 contents of AR3 (after instruction) to 0100. Changed Example 4 contents of AR3 (after instruction) to 0101. Changed Example 5 contents of AR3 (after instruction) to 00FF and syntax: Example 5 `MAR **\*AR3–**`

Page:	Change or Add:

4–94 Changed syntax 2 Execution:

2: (src) − (Xmem) × (Ymem) →**dst**
(Xmem) → T

Changed first paragraph in Description:

This instruction multiplies an operand by the content of T or multiplies two operands, subtracts the result from *src* **unless dst is specified**, and stores the result in *src* or *dst*. *Xmem* is loaded into T in the read phase.

4–102 Changed syntax in Example 1:

Example 1 `MPY` **`13`**`, A`

4–103 Changed syntax in Example 4:

Example 4 `MPYR` **`0`**`, B`

4–108 Changed syntax in Example 1:

Example 1 `MVDK` **`10`**`, 8000h`

4–112 Changed syntax in Example:

Example `MVDP` **`0`**`, 0FE00h`

4–118 Changed syntax in Example 1:

Example 1 `MVPD 0FE00h,` **`5`**

4–119 Changed second paragraph in Description:

If the accumulator equals FF 8000 0000h, the negate operation causes an overflow because the 2s complement of FF 8000 0000h exceeds the lower 32 bits of the accumulator. If OVM = 1, *dst* is assigned 00 7FFF FFFFh. If OVM = 0, *dst* is assigned 00 8000 0000h. The OV bit for *dst* is set to indicate overflow in either case.

4–122 Changed Example 2 contents of T (after instruction) to 0FF9.

4–127 Changed syntax in Example.

Example `POPD` **`10`**

4–135 Changed Example contents of PC (after instruction) to 2002.

4–137 Changed syntax in Example:

Example `READA` **`6`**

Page:	Change or Add:
4–142	Changed Example 1 contents of A (after instruction) to FF FFFF FFFF. Changed Example 2 contents of A (after instruction) to 00 7FFF FFFF.
4–149	Changed Example 1 contents of PC (after instruction) to 1002 and contents of REA (after instruction) to end_block – 1. Changed Example 2:

	Before Instruction		After Instruction
PC	1000	PC	1004
BRC	1234	BRC	0063
RSA	5678	RSA	1004
REA	9ABC	REA	end_block – 1

4–160	Deleted *INTM* from Status Bits.
4–161	Changed syntax in Example 1: Example 1 `SQUR **30**, B`
4–163	Changed syntax in Example 1: Example 1 `SQURA **30**, B`
4–164	Changed syntax in Example 1: Example 1 `SQURS **9**, A`
4–168	Changed syntax in Examples 1 and 2: Example 1 `ST FFFFh, **0**` Example 2 `ST TRN, **5**`
4–170	Changed syntax in Example 1: Example 1 `STH A, **10**`
4–171	Changed syntax in Example 3: Example 3 `STH A, –4, **10**`
4–173	Changed syntax in Example 1: Example 1 `STL A, **11**`
4–174	Changed syntax in Example 3: Example 3 `STL A, 7, **11**`
4–176	Changed Example 2 contents of AR7 (after instruction) to 0011.
4–177	Changed Example contents of AR3 (after instruction) to 0200.

Page: **Change or Add:**

4–179 Added Example 3:

Example 3

```
ST A, *AR2+
||LD *AR2–, A
```

In Example 3, the LD reads the source operand at the memory location pointed to by AR2 before the ST writes to the same location. The ST reads the source operand of accumulator A before LD loads accumulator A.

4–180 Changed Execution:

(**src** << (ASM – 16)) → Ymem
If (Rounding)
Then
Round ((Xmem) × (T) + (dst)) → dst
Else
(Xmem) × (T) + (dst) → dst

Changed first sentence in Description:

This instruction stores *src* shifted by (ASM – 16) in data-memory location *Ymem*.

4–181 Changed Example 1 contents of AR4 (after instruction) to 00FF and contents of AR5 (after instruction) to 0200.

Changed Example 2 contents of Data Memory 100h (after instruction) to 0001.

4–182 Changed Execution:

(**src** << (ASM − 16)) → Ymem
If (Rounding)
Then
Round ((dst) – (Xmem) × (T))→ dst
Else
(dst) – (Xmem) × (T) → dst

Changed first sentence in Description:

This instruction stores *src* shifted by (ASM – 16) in data-memory location *Ymem*.

4–183 Changed Example 1 contents of AR5 (after instruction) to 0200.

Changed Example 2 contents of ASM (before instruction) to 0001.

4–184 Changed Execution:

(**src** << (ASM − 16)) → Ymem
(T) × (Xmem) → dst

Changed first sentence in Description:

This instruction stores *src* shifted by (ASM – 16) in data-memory location *Ymem*.

Page:	Change or Add:
4–185	Changed Execution: (**src** << (ASM − 16)) → Ymem (Xmem) << 16 – (dst_) → dst Changed first sentence of Description: This instruction stores *src* shifted by (ASM – 16) in data-memory location *Ymem*.
4–188	Changed Execution for syntaxes 9 and 10: 9: (**dst**) – (src) << SHIFT → dst 10: (**dst**) – (src) << ASM → dst
4–190	Changed Example 2 contents of C (after instruction) to 1.
4–191	Changed syntax in Example 1: Example 1 `SUBB 5, A`
4–193	Changed syntax in Example 1: Example 1 `SUBC 2, A`
4–194	Changed Execution: (src) – unsigned(Smem) → src
4–197	Changed Data Memory address 1905h to 1005h and syntax: Example `WRITA 5`
D–3	Deleted *CNF* definition.
D–4	Deleted *configuration control bit (CNF)* definition. Changed *DAB address register (DAR)* definition: A register that holds the address to be put on the DAB to address data memory for reads via the DB. Changed *DARAM* definition: Memory that can be **accessed twice** in the same clock cycle. Changed *data address bus* definition: A group of connections used to route data memory addresses. The **'54x** has three 16-bit buses that carry data memory addresses: CAB, DAB, and EAB. Changed *data bus* definition: A group of connections used to route data. The **'54x** has three 16-bit data buses: CB, DB, and EB.

Page:	Change or Add:

D–5 Changed *EAB address register (EAR)* definition:

A register that holds the address to be put on the EAB to address data memory for reads via the EB.

Changed *exponent (EXP) encoder* definition:

A hardware device that computes the exponent value of the accumulator.

Changed *interrupt* definition:

A condition caused by **internal hardware**, an event external to the CPU, or by a previously executed instruction that forces the current program to be suspended and causes the processor to execute an interrupt service routine corresponding to the interrupt.

D–7 Changed *overflow mode bit (OVM)* definition:

A bit in status register **ST1** that specifies how the ALU handles an overflow after an operation.

Changed *pmad* definition:

A 16-bit immediate program-memory address.

D–8 Changed *program address register (PAR)* definition:

A register that holds the address to be put on the PAB to address memory for reads via the PB.

D–9 Changed *software interrupt* definition:

An interrupt caused by the execution of an INTR **or TRAP** instruction.

Changed *temporary register (T)* definition:

A 16-bit register that holds one of the operands for multiply and store instructions, the dynamic shift count for the add and subtract instructions, or the dynamic bit position for the bit test instructions.

Reference Card 1, page 2

Changed following Expression in *Multiply-Accumulate and Multiply-Subtract Instructions*:

Syntax	Expression
MACR Smem, src	**src** = rnd(src + T * Smem)

Page: **Change or Add:**

Reference Card 1, page 3

Changed following Expressions in *Application-Specific Instructions*:

Syntax	Expression
POLY Smem	B = Smem << 16 A = rnd(A**(32–16)** * T + B)
SQDST Xmem, Ymem	B = B + A(32–16) * A(32–16) A = (Xmem **–** Ymem) << 16

Reference Card 1, page 5

Changed Expressions in *Call Instructions*:

Syntax	Expression
CALA[D] src	– –SP, **PC + 1[3] = TOS**, PC = src(15–0)
CALL[D] pmad	– –SP, **PC + 2[4] = TOS**, PC = pmad(15–0)
CC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then – –SP, **PC + 2[4] = TOS**, PC = pmad(15–0)
FCALA[D] src	– –SP **,PC + 1[3] = TOS**, PC = src(15–0), XPC = src(22–16)
FCALL[D] extpmad	– –SP, **PC + 2[4] = TOS**, PC = pmad(15–0), XPC = pmad(22–16)

Changed Expressions in *Interrupt Instructions*.

Syntax	Expression
INTR K	– –SP, **++ PC = TOS**, PC = IPTR(15–7) + K << 2, INTM = 1
TRAP K	– –SP, **++ PC = TOS**, PC = IPTR(15–7) + K << 2

Page: **Change or Add:**

Reference Card 1, page 5

Changed Expressions in *Return Instructions*:

Syntax	Expression
FRET[D]	XPC = **TOS, ++ SP, PC = TOS, ++SP**
FRETE[D]	XPC = **TOS, ++ SP, PC = TOS, ++SP**, INTM = 0
RC[D] cond [, cond [, cond]]	if (cond(s)) then PC = **TOS, ++SP**
RET[D]	PC = **TOS, ++SP**
RETE[D]	PC = **TOS, ++SP**, INTM = 0
RETF[D]	PC = RTN, **++SP**, INTM = 0

Changed following Expression in *Repeat Instructions*:

Syntax	Expression
RPTB[D] pmad	Repeat block, RSA = PC + 2[4], REA = pmad, **BRAF = 1**

Reference Card 1, page 6

Changed following Expressions in *Stack-Manipulating Instructions*:

Syntax	Expression
POPD Smem	Smem = **TOS, ++SP**
POPM MMR	MMR = **TOS, ++SP**
PSHD Smem	– –SP, **Smem = TOS**
PSHM MMR	– –SP, **MMR = TOS**

Reference Card 2, page 1

Changed *Indirect Addressing Types With a Dual Data-Memory Operand*:

*ARx	***ARx+**
*ARx–	***ARx+ 0%**

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.