

DSP56374E Digital Signal Processor

Mask 1L60W

General remark: In order to prevent the use of instructions or sequences of instructions that do not operate correctly, we encourage you to use the "lint563" program to identify such cases and use alternative sequences of instructions. This program is available as part of the Motorola DSP Tools CLAS package.

Silicon Errata

Errata Number: ES131

Applies to Mask: 1L60W

Description (added 11/28/00):

When all of the following conditions are true:

1. Executing a conditional change of flow (branch, jump) instruction and the branch or jump is *not* taken, and
2. One of the two program memory words following that instruction includes an address or displacement (as a whole word, or as a field), and
3. DMA transactions *are* occurring simultaneously,

the address or displacement might not be calculated correctly.

Workaround: Do one of the following:

- A. If only the second word after the instruction includes the address or displacement, add one NOP after the conditional branch, or
- B. If the first word after the instruction includes the address or displacement, add two NOPs after the conditional branch.

Errata Number: ED1

Applies to Mask: 1L60W

Description (added 11/9/98)

XY memory data move does not work properly under one of the following two situations:

1. The X-memory move destination is internal I/O and the Y-memory move source is a register used as destination in the previous adjacent move from non Y-memory
2. The Y-memory move destination is a register used as source in the next adjacent move to non Y-memory.

Here are examples of the two cases (where x:(r1) is a peripheral)

Example 1:

```
move #512,y0
move x0,x:(r7) y0,y:(r3) ;(while x:(r7) is a peripheral).
```

Example 2:

```
mac      x1,y0,a x1,x:(r1)+      y:(r6)+,y0
move     y0,y1
```

Any of the following alternatives can be used:

- A. Separate these two consecutive moves by any other instruction.
- B. Split XY Data Move to two moves.

Pertains to: DSP56300 Family Manual, Section B-5 "Peripheral pipeline restrictions.

Errata Number: ED4

Applies to Mask: 1L60W

Description (added 10/31/1997)

The following instructions should not start at address LA:

MOVE to/from Program space {MOVEM, MOVEP (only the P space options)}

This is not a bug but a documentation update (Appendix B, DSP56300 Family Manual).

Errata Number: ED28

Applies to Mask: 1L60W

Description (added 1/7/1997; identified as Documentation Errata 2/1/99)

When two consecutive LAs have a conditional branch instruction at LA-1 of the internal loop, the part does not operate properly. For example, the following sequence may generate incorrect results:

```

DO #5, LABEL1
NOP
DO #4, LABEL2
NOP
MOVE (R0) +
BSCC _DEST;conditional branch at LA-1 of
;internal loop
NOP;internal LA
LABEL2
NOP;external LA
LABEL1
NOP
NOP
_DEST NOP
NOP
RTS
```

Workaround: Put an additional NOP between LABEL2 and LABEL1.

Pertains to: DSP56300 Family Manual, Appendix B, Section B-4.1.3, "At LA-1."

Errata Number: ED32

Applies to Mask: 1L60W

Description (added 11/9/98; identified as a Documentation errata 2/1/99)

When returning from a long interrupt (by RTI instruction), and the first instruction after the RTI is a move to a DALU register (A, B, X, Y), the move may not be correct, if the 16-bit arithmetic mode bit (bit 17 of SR) is changed due to the restoring of SR after RTI.

Workaround: Replace the RTI with the following sequence:

```
movec    ssl, sr
nop
rti
```

Pertains to: DSP56300 Family Manual. Add a new section to Appendix B that is entitled “Sixteen-Bit Compatibility Mode Restrictions.”

Errata Number: ED33

Applies to Mask: 1L60W

Description (added 12/16/98; identified as a Documentation errata 2/1/99):

When Stack Extension mode is enabled, a use of the instructions BRKcc or ENDDO inside do loops might cause an improper operation.

If the loop is non nested and has no nested loop inside it, the errata is relevant only if LA or LC values are being used outside the loop.

Workaround: If Stack Extension is used, emulate the BRKcc or ENDDO as in the following examples. We split between two cases, finite loops and do forever loops.

1. Finite DO loops (i.e. not DO FOREVER loops)

=====

BRKcc

Original code:

```
do #N, label1
.....
.....
do #M, label2
.....
.....
BRKcc
.....
.....
label2
.....
.....
label1
```

Will be replaced by:

```
do #N, label1
.....
.....
do #M, label2
.....
```

```

        .....
        Jcc    fix_brk_routine
        .....
        .....
nop_before_label2
        nop    ;This instruction must be NOP.
label2
        .....
        .....
label1
.....
.....

fix_brk_routine
    move #1,lc
    jmp  nop_before_label2

```

ENDDO

Original code:

```

        do #M,label1
        .....
        .....
            do #N,label2
            .....
            .....
            ENDDO
            .....
            .....
label2
        .....
        .....
label1

```

Will be replaced by:

```

        do #M,label1
        .....
        .....
            do #N, label2
            .....
            .....
            JMP    fix_enddo_routine
nop_after_jump
        NOP    ; This instruction must be NOP.
        .....
        .....
label2
        .....
        .....
label1
.....
.....

```

```
fix_enddo_routine
    move #1,lc
    move #nop_after_jump,la
    jmp  nop_after_jump
```

2. DO FOREVER loops

=====

BRKcc

Original code:

```
        do #M,label1
        .....
        .....
            do forever,label2
            .....
            .....
            BRKcc
            .....
            .....
label2
    .....
    .....
label1
```

Will be replaced by:

```
        do #M,label1
        .....
        .....
            do forever,label2
            .....
            .....
            JScC    fix_brk_forever_routine ; <---
note: JScC and not Jcc
            .....
            .....
nop_before_label2
        nop        ; This instruction must be NOP.
label2
    .....
    .....
label1
    ....
    ....
```

```
fix_brk_forever_routine
    move ssh,x:<..> ; <..> is some reserved not used
address (for temporary data)
    move #nop_before_label2,ssh
    bclr #16,ssl    ;
    move #1,lc
    rti            ; <----- note: "rti" and not "rts" !
```

ENDDO

Original code:

```
do #M,label1
    .....
    .....
    do forever,label2
    .....
    .....
    ENDDO
    .....
    .....
label2
    .....
    .....
label1
```

Will be replaced by:

```
do #M,label1
    .....
    .....
    do forever,label2
    .....
    .....
    JSR    fix_enddo_routine    ; <--- note:
```

JSR and not JMP

```
nop_after_jump
    NOP ; This instruction should be NOP
    .....
    .....
label2
    .....
    .....
label1
    ....
    ....
```

```
fix_enddo_routine
    nop
    move #1,lc
    bclr #16,ssl
    move #nop_after_jump,la
    rti    ; <--- note: "rti" and not "rts"
```

Pertains to: DSP56300 Family Manual, Section B-4.2, "General Do Restrictions."

Errata Number: ED34**Applies to Mask: 1L60W**

Description (added 1/5/99; identified as a Documentation errata 2/1/99)

When stack extension is enabled, the read result from stack may be improper if two previous executed instructions cause sequential read and write operations with SSH. Two cases are possible:

Case 1:

For the first executed instruction: move from SSH or bit manipulation on SSH (i.e. jclr, brclr, jset, brset, btst, bset, jset, bsclr, jsclr).

For the second executed instruction: move to SSH or bit manipulation on SSH (i.e. jsr, bsr, jscc, bscc).

For the third executed instruction: an SSL or SSH read from the stack result may be improper - move from SSH or SSL or bit manipulation on SSH or SSL (i.e., bset, bclr, bchg, jclr, brclr, jset, brset, btst, bset, jset, bsclr, jsclr).

Workaround: Add two NOP instructions before the third executed instruction.

Case 2:

For the first executed instruction: bit manipulation on SSH (i.e. bset, bclr, bchg).

For the second executed instruction: an SSL or SSH read from the stack result may be improper - move from SSH or SSL or bit manipulation on SSH or SSL (i.e., bset, bclr, bchg, jclr, brclr, jset, brset, btst, bset, jset, bsclr, jsclr).

Workaround: Add two NOP instructions before the second executed instruction.

Pertains to: DSP56300 Family Manual, Appendix B, add a new section called "Stack Extension Enable Restrictions." Cover all cases. Also, in Section 6.3.11.15, add a cross reference to this new section.

Errata Number: ED58**Applies to Mask: 1L60W**

Description (Added 10/29/04):

The DSP56374UM fails to note that when the SHI is set in master mode and the SHI clock is an output, Fosc must be greater than 8 x sck. i.e. the DSP system clock (Fosc) must be 8 times greater than the SHI clock.

Workaround: Do not use the SHI in master mode as an output when the system clock/shi clock ratio is set to 8 or less.

Pertains to: DSP56374 User Manual, Section 7, Serial Host Interface.

NOTES

1. An over-bar (i.e., $\overline{\text{xxxx}}$) indicates an active-low signal.
2. The letters in the right column tell which DSP56371 mask numbers apply.
3. The Motorola DSP website has additional documentation updates that can be accessed at the following URL:

<http://www.freescale.com/>

How to Reach Us:

USA/Europe/Locations Not Listed:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

Home Page:

www.freescale.com



Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Learn More: For more information about Freescale products, please visit www.freescale.com

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2004