TMS320 DSP
DESIGNER'S NOTEBOOK

Improved Context Save/Restore Performance and Interrupt Latency for ISRs Written in C

APPLICATION BRIEF: SPRA232

Alex Tessarolo
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
May 1994

TEXAS INSTRUMENTS

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 application 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Abstract 7
Design Problem 8
Solution 8

Examples

Example 1. Interrupt Vector File 8
Example 2. Assembly Language File with the Interrupt Service Context Save and Restore Functions for INT1 and INT2: 8
Example 3. C Code File with the Interrupt Service Routines: 10

Improved Context Save/Restore Performance and Interrupt Latency for ISRs Written in C

Abstract

In some control operations, the interrupt context save time must be minimized as much as possible and higher-priority interrupts must be delayed as little as possible in order to minimize interrupt latency. In the standard C interrupt service routines for the ’C2x/’C5x/’C3x/’C4x, the context save and restore functions are not fully optimized. This document shows both assembly language code and C code examples to optimize performance.

Design Problem

In some control operations, the interrupt context save time must be minimized as much as possible and higher-priority interrupts must be delayed as little as possible, to minimize interrupt latency. In the standard C interrupt service routines for the ’C2x/’C5x/’C3x/’C4x, the context save and restore functions are not fully optimized.

Solution

The example below shows how the interrupt service context save time can be reduced by 34% and the interrupt latency minimized by replacing the standard I$$SAVE and I$$REST context save/restore functions with optimized versions. The ’C2X is shown as an example, but similar techniques can be applied to the ’C5x, ’C3x, and ’C4x compilers to improve ISR performance in C.

For this example, we assume external INT1 is the highest priority interrupt and INT2 is a lower priority interrupt. We require that INT2 disable interrupts and enable INT1 to occur with minimum interrupt latency:

Example 1. Interrupt Vector File

```
        .ref _c_int0, Int1CTXT, Int2CTXT


SP      .set AR1
        .sect   “vectors”
RESET   b  _c_int0       ; External Reset.
INT1    b  Int1CTXT,*,SP ; External H/W Interrupt 1.
INT2    b  Int2CTXT,*,SP ; External H/W Interrupt 2.
        .
        .
```

Example 2. Assembly Language File with the Interrupt Service Context Save and Restore Functions for INT1 and INT2:

```
     .ref    _Int1, _Int2
     .def    Int1CTXT, Int2CTXT
SP   .set    AR1
     .text
;
; External H/W Interrupt 1, ISR context save/restore:
;
; Benchmark = 46 cycles (includes branch, call & ret,
;                                  ret)
; Notes: - Interrupts disabled for duration of ISR.
;        - The above benchmark is 24 cycles (34%)
;            faster than the standard I$$SAVE,
;            I$$REST functions.
```

```
Int1CTXT:           ; Assumed ARP -> SP.
      mar   *+      ; Increment stack pointer.
      sst1  *+      ; Save ST1.
      sst   *+      ; Save ST0.
      sacl  *+      ; Save ACCL.
      sach  *+      ; Save ACCH.
      popd  *+      ; Save top two levels of
      popd  *+      ; H/W stack only.
      spm   0
      sph   *+      ; Save PH.
      spl   *+      ; Save PL.
      mpyk  1       ; Save T.
      spl   *+
      sar   AR2,*+  ; Save aux registers that
      sar   AR3,*+  ; are not saved by C compiler.
      sar   AR4,*+
      sar   AR5,*+
      call  _Int1   ; Call C ISR.
      mar   *-      ; Decrement stack ptr.
      lar   AR5,*-  ; Restore aux registers.
      lar   AR4,*-
      lar   AR3,*-
      lar   AR2,*-
      lacl  *-      ; Temp save T in ACCL.
      lt    *-      ; Restore PL.
      mpyk  1
      lph   *       ; Restore PH.
      sacl  *       ; Restore T.
      lt    *-
      pshd  *-      ; Restore two levels of H/W
      pshd  *-      ; stack only.
      lacc  *-,16   ; Restore ACCH.
      adds  *-      ; Restore ACCL.
      lst   *-      ; Restore ST0.
      lst1  *-      ; Restore ST1.
      eint          ; Global interrupt enable.
      ret           ; Return to interrupted code.
;
; External H/W Interrupt 2, ISR context save/restore:
;
; Benchmark = 61 cycles (includes branch, call & ret,
;                            ret) ;
; Notes: Higher priority interrupts disabled for 20
;   cycles.
;
Int2CTXT: ; Assumed ARP -> SP.
      mar   *+      ; Increment stack pointer.
      sst1  *+      ; Save ST1.
      sst   *+      ; Save ST0.
      sacl  *+      ; Save ACCL.
      sach  *+      ; Save ACCH.
      ldpk  0       ; DP -> 0.
      pshd  IMR     ; Save IMR.
```

```
lack  00000001b   ; Set mask to enable INT1.
and   IMR         ; Mask with IMR.
sacl  IMR         ; Set IMR.
rptk  3           ; Save top four levels of
popd  *+          ; H/W stack only.
eint              ;Global interrupt enable.
spm   0
sph   *+          ; Save PH.
spl   *+          ; Save PL.
mpyk  1           ; Save T.
spl   *+
sar   AR2,*+      ; Save aux registers that
sar   AR3,*+      ; are not saved by C compiler.
sar   AR4,*+
sar   AR5,*+
call  _Int2       ; Call C ISR.
mar   *-          ; Decrement stack ptr.
lar   AR5,*-      ; Restore aux registers.
lar   AR4,*-
lar   AR3,*-
lar   AR2,*-
lacl  *-          ; Temp save T in ACCL.
lt    *-          ; Restore PL.
mpyk  1
lph   *           ; Restore PH.
sacl  *           ; Restore T.
lt    *-
dint              ; Global interrupt disable.
rptk  3           ; Restore four levels of H/W
pshd  *-          ; stack only.
ldpk  0           ; DP -> 0.
popd  *-          ; Restore IMR.
lacc  *-,16       ; Restore ACCH.
adds  *-          ; Restore ACCL.
lst   *-          ; Restore ST0.
lst1  *-          ; Restore ST1.
eint              ; Global interrupt enable.
ret               ; Return to interrupted code.
```

Example 3. C Code File with the Interrupt Service Routines:

```
void main( void )
{
     /* user code */
}
void Int1( void )
{
     /* INT1 Interrupt service code */
}
void Int2( void )
{
     /* INT2 Interrupt service code */
}
```