TMS320 DSP
DESIGNER'S NOTEBOOK
Number 36

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

Contributed by Alex Tessarolo

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 the same concepts may be applied to other DSPs in the TI family.

Example:
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 a minimum interrupt latency:

The following is the 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.
          .
          .
```

Figure 1.

The following is an 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.
;
```

Figure 1. (continued)

```
; 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.
```

Figure 1. (continued)

The following is an example C 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 */
}
```

Figure 1. (continued)

Similar techniques can be applied to the ’C5x, ’C3x, and ’C4x compilers to improve ISR performance in C.