32X TECHNICAL BULLETIN #27

To: **Sega and Third Party Developers**

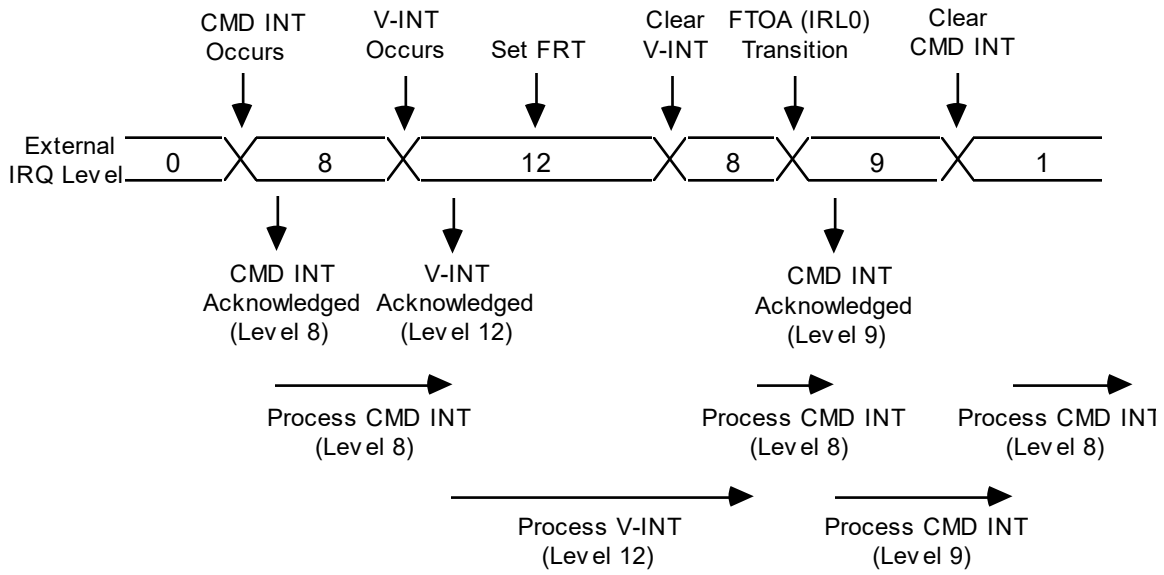
From: **Developer Technical Support**

Date: **December 8, 1994**

Re: **SH2 Interrupt Problems on the 32X**

When two or more types of interrupts are used by the SH2, an interrupt of the same level may be acknowledged twice consecutively in relation to the occurrence of a single interrupt. The following explains this phenomena:

Example



Explanation

When the CMD INT occurs, the SH2 acknowledges the interrupt (recognized as level 8) and starts to process it. However, if a V-INT occurs before the SR is masked, the SH2 processes the V-INT instead since it has higher priority than the currently masked level (in this case, 8). Moreover, the FRT is set and the V-INT is cleared during this process, changing the external interrupt level as a result.

After the V-INT is processed, processing on the previously acknowledged CMD INT begins again. However, if the FRT output value changes before the SR is masked, the

SH2 will judge that a higher priority level interrupt has occurred than the currently masked level (in this case level 8). As a result, the SH2 begins to process a CMD INT (acknowledged as level 9) instead. After this CMD INT is processed, the SH2 will process the CMD INT acknowledged as a level 8 interrupt once again. This means that processing on the CMD INT will occur twice, even though it actually occurred once. Refer to the interrupt handler code example below that corrects this problem (This code can also be found in the **32X Hardware Manual Supplement 2**).

(Note: this problem only occurs on the 32X. It does not occur on the Saturn.)

```
;---------------------------------------------------------------------
;      Interrupt Handler Example Code (1994.9.20)
;---------------------------------------------------------------------

CS0             .equ    h'00000000      ; boot ROM, register
CS1             .equ    h'02000000      ; cartridge ROM
CS2             .equ    h'04000000      ; Frame Buffer
CS3             .equ    h'06000000      ; SDRAM

TH              .equ    h'20000000      ; cache thru
CS0TH           .equ    h'20000000      ; boot ROM, register (cache thru)
CS1TH           .equ    h'22000000      ; cartridge ROM (cache thru)
CS2TH           .equ    h'24000000      ; Frame Buffer (cache thru)
CS3TH           .equ    h'26000000      ; SDRAM (cache thru)

_SERIALMODE     .equ    h'fffffe00      ; Serial Mode Register

_FRT            .equ    h'fffffe10      ; Free Run Timer
_TIER           .equ    h'00            ; Timer Interrupt Enable Register
_TCSR           .equ    h'01            ; Timer Control & Status Register
_FRC_H          .equ    h'02            ; Free Running Counter High
_FRC_L          .equ    h'03            ; Free Running Counter Low
_OCR_H          .equ    h'04            ; Output Compare Register High
_OCR_L          .equ    h'05            ; Output Compare Register Low
_TCR            .equ    h'06            ; Timer Control Register
_TOCR           .equ    h'07            ; Timer Output Compare Control Register

_CCRREG         .equ    h'fffffe92      ; cache control register

_JR             .equ    h'ffffff00      ; DIVU
_HRL32          .equ    h'ffffff04      ; DIVU
_HRH            .equ    h'ffffff10      ; DIVU
_HRL            .equ    h'ffffff14      ; DIVU

_DMASOURCE0     .equ    h'ffffff80      ; DMA Source Address 0
_DMADEST0       .equ    h'ffffff84      ; DMA Destination Address 0
_DMACOUNT0      .equ    h'ffffff88      ; DMA Transfer Count 0
_DMACHANNEL0    .equ    h'ffffff8c      ; DMA Channel Control 0
_DMASOURCE1     .equ    h'ffffff90      ; DMA Source Address 1
_DMADEST1       .equ    h'ffffff94      ; DMA Destination Address 1
_DMACOUNT1      .equ    h'ffffff98      ; DMA Transfer Count 1
_DMACHANNEL1    .equ    h'ffffff9c      ; DMA Channel Control 1
_DMAVECTORN0    .equ    h'ffffffa0      ; DMA Vector No. N0
_DMAVECTORE0    .equ    h'ffffffa4      ; DMA Vector No. E0
_DMAVECTORN1    .equ    h'ffffffa8      ; DMA Vector No. N1
_DMAVECTORE1    .equ    h'ffffffac      ; DMA Vector No. E1
_DMAOPERATION   .equ    h'ffffffb0      ; DMA Operation
_DMAREQACK0     .equ    h'ffffffb4      ; DMA Request/Ack Select Control 0
_DMAREQACK1     .equ    h'ffffffb8      ; DMA Request/Ack Select Control 1

;
; SYSREG.
;
_sysreg         .equ    h'00004000+TH   ; SYSREG.
adapter         .equ    h'00            ; Adapter Control Register
intmask         .equ    h'01            ; Interrupt Mask
standby         .equ    h'02            ; transfer stand-by monde
hcount          .equ    h'05            ; H Interrupt Counter Reister
vdpfifo         .equ    h'06            ; Frame Buffer FIFO Condition
dreqctl         .equ    h'07            ; DREQ Control
dreqsource      .equ    h'08            ; DREQ Source Address
```

```
dreqdest        .equ    h'0c            ; DREQ Destination Address
dreqlen         .equ    h'10            ; DREQ Lengtth
fifo            .equ    h'12            ; FIFO
vresintclr      .equ    h'14            ; VRES Interrupt Clear

vintclr         .equ    h'16            ; V Interrupt Clear
hintclr         .equ    h'18            ; H Interrupt Clear
cmdintclr       .equ    h'1a            ; CMD Interrupt Clear
pwmintclr       .equ    h'1c            ; PWM Interrupt Clear
comm0           .equ    h'20            ; Communication Port
comm2           .equ    h'22            ;
comm4           .equ    h'24            ;
comm6           .equ    h'26            ;
comm8           .equ    h'28            ;
comm9           .equ    h'29            ;
comm10          .equ    h'2a            ;
comm12          .equ    h'2c            ;
comm14          .equ    h'2e            ; PWM Timer Control
timerctl        .equ    h'30            ; PWM Control
pwmctl          .equ    h'31            ; PWM
cycle           .equ    h'32            ;
lchwidth        .equ    h'34            ;
rchwidth        .equ    h'36            ;
monowidth       .equ    h'38            ;

;
; VDPREG.
;
_vdpreg         .equ    h'00004100+TH   ; VDPREG.
tvmode          .equ    h'00            ; TV Mode Register
bitmapmd        .equ    h'01            ; Bitmap Mode Register
shift           .equ    h'03            ; Shift Control Register
filllength      .equ    h'05            ; Auto Fill Length Register
fillstart       .equ    h'06            ; Auto Fill Start Address Register
filldata        .equ    h'08            ; Auto Fill Data Register
vdpsts          .equ    h'0a            ; VDP Status Register
framectl        .equ    h'0b            ; Frame Buffer Control Register

_palette        .equ    h'00004200+TH   ; Palette RAM
_framebuffer    .equ    CS2TH           ; Frame Buffer
_overwrite      .equ    CS2TH+h'20000   ; Over Write Image

;
; SH2 Vector
;
vector:
                .data.l start           ; Power On Reset PC
_stack:
                .data.l CS3+h'3ff00,    ; Power On Reset SP
+                       start,          ; Manual Reset PC
+                       CS3+h'3ff00     ; Manual Reset SP
                .data.l error0,         ; incorrect general command
+                       h'00000000,     ; reserved for system
+                       eror0,          ; incorrect slot command
+                       h'20100400,     ; reserved for system (ICE Vector)
+                       h'20100420,     ; reserved for system (ICE Vector)
+                       error0,         ; CPU Address Error
+                       error0,         ; DMA Address Error
+                       error0,         ; NMI
+                       error0          ; user break
                .datab.l 19,h'00000000  ; reserved for system
```

```
                .datab.l  32,error0       ; trap command
                .data.l   m_int,          ; interrupt 1
+                         m_int,          ; interrupt 2, 3
+                         m_int,          ; interrupt 4, 5
+                         m_int,          ; interrupt 6, 7
+                         m_int,          ; interrupt 8, 9
+                         m_int,          ; interrupt 10, 11
+                         m_int,          ; interrupt 12, 13
+                         m_int,          ; interrupt 14, 15

;
; Program Start
;
Start:
                mov.l     #_sysreg,r14
                ldc       r14,gbr

                mov.l     #_FRT,r1        ; Set Free Run Timer
                mov       #h'00,r0
                mov.b     r0,@(_TIER,r1)
                mov       #h'e2,r0
                mov.b     r0,@(_TOCR,r1)
                mov       #h'00,r0
                mov.b     r0,@(_OCR_H,r1)
                mov       #h'01,r0
                mov.b     r0,@(_OCR_L,r1)
                mov       #0,r0
                mov.b     r0,@(_TCR,r1)
                mov       #1,r0
                mov.b     r0,@(_TCSR,r1)
                mov       #h'00,r0
                mov.b     r0,@(_FRC_H,r1)
                mov.b     r0,@(_FRC_L,r1)
wait_md:
                mov.l     @(comm0,gbr),r0   ; sync timing with GENESIS
                cmp/eq    #0,r0
                bf        wait_md

                mov       #h'20,r0
                ldc       r0,sr           ; SH2 Interrupt Enable

;
; Interrupt Control
;
m_int:
                mov.l     r0,@r15
                mov.l     r1,@r15
                mov.l     r2,@r15

                stc       sr,r1
                mov.w     #h'f0,r2
                ldc       r2,sr
                mov.w     #h'fe10,r2
                xor       r0,r0
                mov.b     r0,@(_TOCR,r2)
                mov.b     @(_TOCR,r2),r0
                sts.l     pr,@r15
                mov       r1,r0
                shlr2     r0
                and       #h'3c,r0
                mov.l     #inttable,r1
```

```
                mov.l   @(r0,r1),r0
                jsr     @r0
                nop

                lds.l   @r15+,pr
                mov.l   @r15+,r2
                mov.l   @r15+,r1
                mov.l   @r15+,r0
                rte
                nop

                .align  4
int table:
                .data.l noret,                  ; Illegal Interrupt
+noret,noret,noret,noret,noret,                 ; Level 1~5
+pwmint,pwmint,cmdint,cmdint                    ; Level 6~9
+hint,hint,vint,vint,vresint,vresint            ; Level 10~15

;Make the even and odd labels for the external interrupt vectors
;the same address

;
;Ignore
;
noret:
                rts
                nop

;
;VRES Interrupt
;
vresint:
                mov.l   #sysreg,r0
                ldc     r0,gbr

                mov.w   r0,@(vresintclr,gbr)    ; V Interrupt Clear

                mov.l   #_stack,r1      ; modify stack pointer
                mov.l   @1,r15

                mov.l   #_hotstart,r0
                mov     r0,@15          ; modify PC
                mov.w   #h'f0,r0
                mov     r0,@(4,r15)     ; mask SR
                rte
                nop

;
;V Interrupt
;
vint:
                stc.l   gbr,@r15
                mov.l   #_sysreg,r0
                ldc     r0,gbr

                mov.l   #_FRT,r1
                mov.w   r0,@(vintclr,gbr)
                mov     #h'02,r0
                mov.b   r0,@(_TOCR,r1)

                mov.b   @(_TOCR,r1),r0
```

```
                mov.w   @(vintclr,gbr),r0

;Other processing (more than 8 clocks necessary for rte)

                ldc.l   @15+,gbr
                rts
                nop

;Do the same as above for H,CMD, and PWM interrrupts.
```