SEGA™

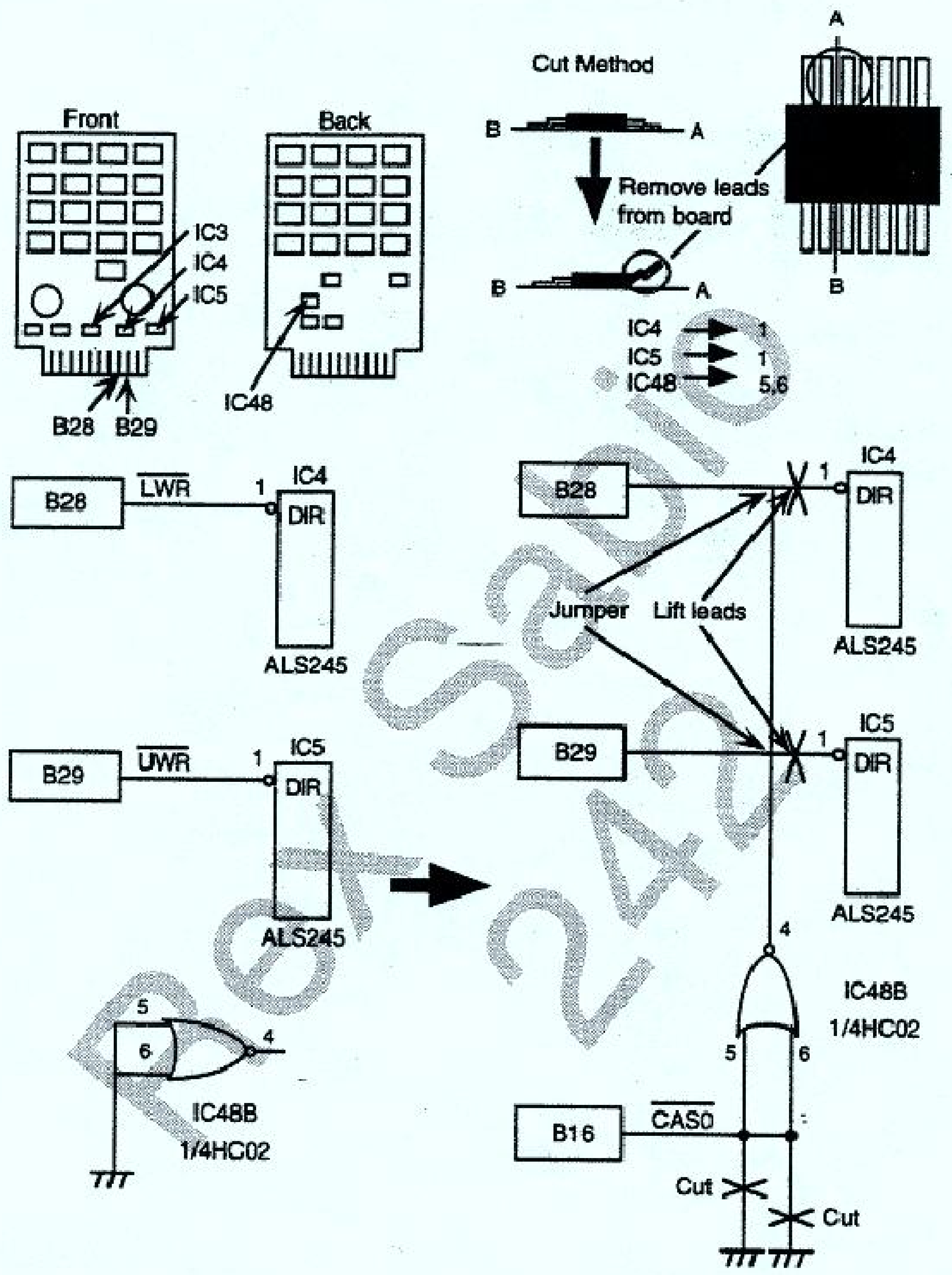
SEGA OF AMERICA, INC.
Consumer Products Division

32X
Technical Information

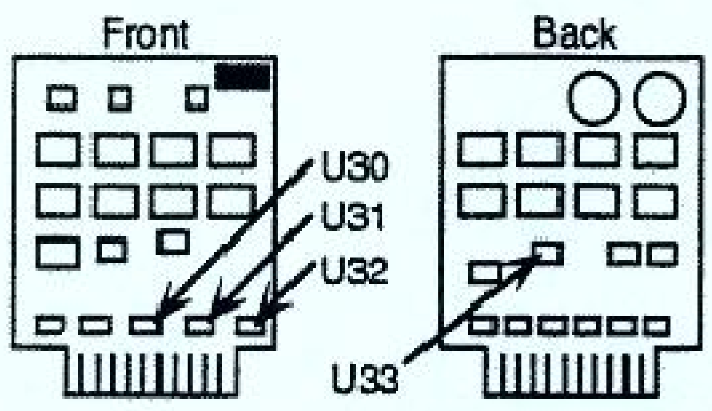
Doc. # MAR-41-R8-090694

32X Technical Information

1. The following modifications are necessary when a Mega Drive 32 Mbit SRAM board is used with the 32X.



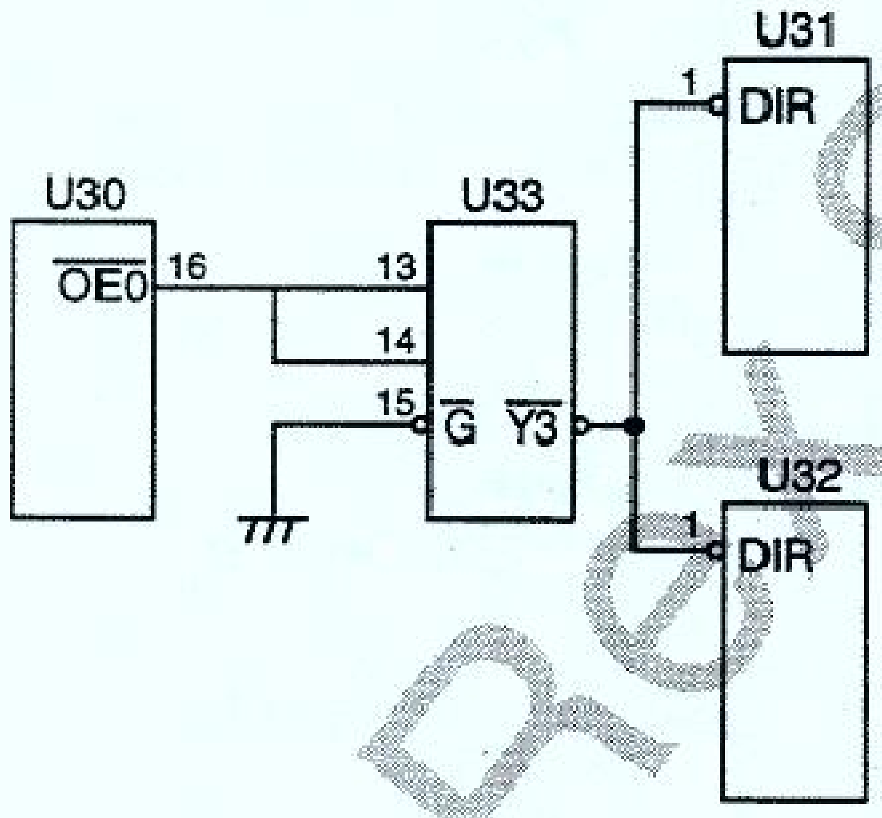
2. The following modifications are necessary when a Mega Drive 16 Mbit SRAM board is used with the 32X.



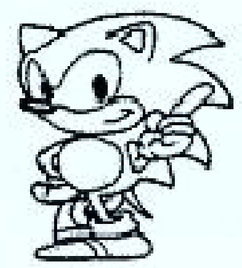
Work Procedure

1. Lift no. 1 pins of U31 and U32.
2. Similarly, lift no. 13, 14, and 15 pins of U33.
3. Jump the no. 16 pin of U30 and the no. 13 and 14 pins of U33.
4. Connect the no. 15 pin of U33 to the GND (no. 8 pin).
5. Jump the no. 9 pin of U33 to each no. 1 pin of U31 and U32.

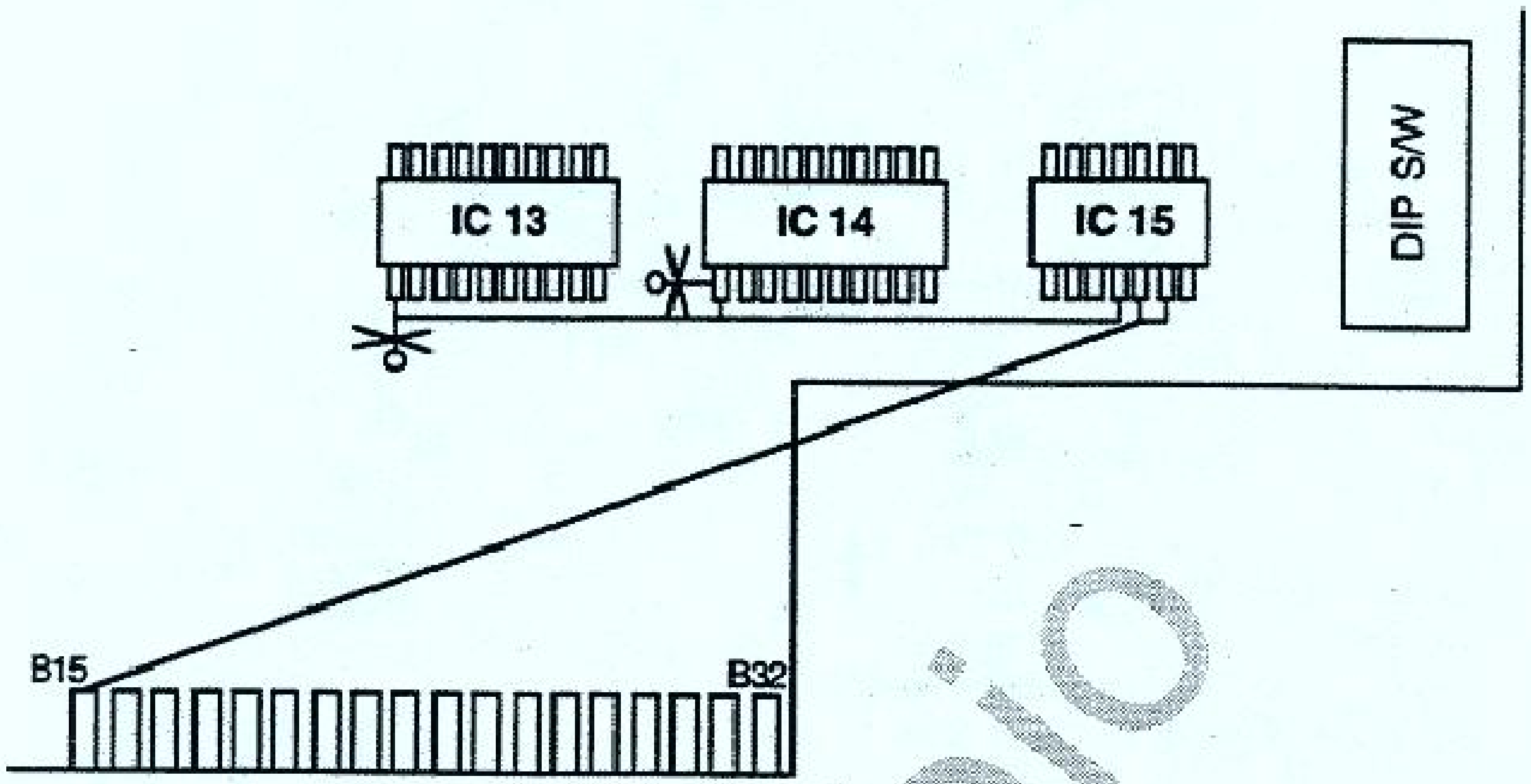
After Modification



Operation cannot be guaranteed if U31 and U32 are not AC or ALS type.
If equipped with the LS type, theAC or ALS type should be attached, replacing the LS type.



3. The following modifications are necessary when a Mega Drive 32 Mbit ROM board is used with the 32X.



Work Procedure

1. Cut between the pattern of the IC13 no.1 pin and the through-hole below the pin.
2. Cut between the pattern of the IC14 no.1 pin and the through-hole to the left of the pin.
3. Cut the trace pattern of the IC 15 no.5 and no.6 pins.
4. Wire (connect) IC13 no.1 pin, IC14 no.1 pin, and IC15 no.4 pin.
5. Wire (connect) th IC 15 no.5 and no.6 pins to B16 of the card edge.

4. ICE must not be reset when in the break condition.
It should be reset while the go command is being executed.
If ICE is reset in the break condition, it will not operate when the power is turned on.

5. The 32x target Ver. 2.0 does not operate unless SH2 is cut 2.3 or greater and the IPI version is 1.21 or greater.

6. If doing VRESET with the RV bit equal to 1, it will not restart if the power is turned of. See Technical Information No. 1 for more information.

7. The RV bit should be equal to 1 when reading and writing to the FRAM. Normallyit is not possible to read and write when the RV bit is equal to 0. The RV bit is irrelevant when reading and writing to the SRAM.

8. Development equipment is not required when using the bank IC, but initialization by software must be done due to miss operations by the sample cartridge.

   (8/1/94)
   Specifically, after the power is turned on

```
          lea        $A130F1.a1      ;
          tst.b      (a1)            ;    bank dummy read
          moveq      #0, d0          ;    do $200000 ~ $3fffff in ROM
          move.w     #$2700, sr      ;    set in stop interrupt condition
          jsr        BankSet         ;    address=$c0 (routine within Vector ROM)
          move.w     #$2000, sr      ;    Return SR to origin if required
```

   Implement corrective measures with respect to the super 16 Mbit software whether or not a bank IC is used due to the structure (design) of the commercial cartridge (not required with 16 Mbit + backup).

9. All 68000 interrupts must be prohibited during the period that the RV bit = 1.

10. Initialization of the free run timer must be done whether or not interrupt is used in coping with SH2 interrupt.

11. Concerning versions of the MD I/F chip

Chip No.	Development target	
315-5780	Ver. 2.0	Bug in PWM, chip must be exchanged
315-5818	Ver. 2.0A	Speed is slow
315-5818A	Ver. 2.0A	Speed is fast

   5818 and 5818A are mixed in the product

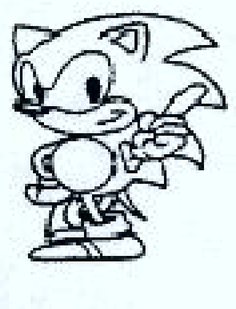
   With respect to 5818, as errors seldom occur in SH2 - 68000 communication, please use 5818A-loaded target/product in final check.

12. About ROM read when RV =1

   The following address data is not normally read when RV=1.
   $1070 ~ $1073 (4 bytes)
   $2070 ~ $2073 (4 bytes)
   $3070 ~ $3073 (4 bytes)   total of 12 bytes

13. When a CD is used, Word-RAM-to-VRAM-capture-DMA has a problem in the timing of data fetch, and since data becomes undefined, it can not be used.

14. In the 32X VDP, the shift bit becomes invalid when the lower byte of the base address set in the line table is $FF. Therefore, make sure the lower byte in the table is not $FF when using shift.



15. In the current (Aug. 15, 1994) target version 2.0A, the 68000 locks up when Z80 accesses the 68000 area. Revision is required to develop a program in which Z80 accesses the 68000 area.

    With target version 2.0B (contains corrective measures to the above problems concerning version 2.0A), version 2.1 (scheduled for Sept. 1994), and their products, the problems above are solved but the following limitation is added.

    Writing to $840000 ~ $9FFFFF as well as the $A15100 ~ $A153FF area by the Z80 causes the 68000 to locks up; therefore, in such cases do not write by the Z80 (read is okay). The MD work RAM and PSG can be written to.

16. About target Version 2.1

    Target version 2.1 (scheduled for Sept. 1994) performs corrective action no. 6 VRES in this document. Z80 corrective action of no. 15 is performed. The NMI of each CPU is collected and wired to one spot.

17. Regarding 32X development board security release:

    32X development board starts up from 68000 side boot ROM when power is switched on. As a result, when power is on, R/W against cartridge connector is not allowed. To avert this, make sure to write 1 using 1 byte to $A14100. GENESIS OS will be released, and the normal access will be allowed.

18. (8/22/94)
    For EPROM, please use those with access time 120 nsec or less. Specifically, make sure the Tacc and Tce catalog values are 120 nsec (MAX) or less (this should be a condition).

    Major Makers' model number examples:

Toshiba	16M	16-bit bus	TC5716200D-120
	4M	16-bit bus	TC574200D-120
NEC	4M	8-bit bus	µPD27C4001D-120

19. (8/25/94)
    Only word unit-based access is allowed from SH2 to PWM Lch pulse width register (20004034H), Rch pulse width register (20004036H), and MONO pulse width register (20004038H). The byte unit-based access is prohibited.

    With respect to PWM control register and period (cycle) register, the byte unit-based access is allowed.

    From the 68000, with respect to each pulse width register, the byte unit-based access is allowed by writing in order of the upper byte and the lower byte.

20. (8/29/94)
In target ver 2.0x development board, compared to FM sound source, PWM sound volume has become fairly small. Please add the following modification. The volume products have already been modified. Some working samples have not yet been modified. The target ver 3.0 (the next development target scheduled for the end of September) has already been modified.

Modification: Change R71 and R73 from 3 KΩ to 10 KΩ.

Through the above measures, the balance between FM sound source and PWM sound source is improved. However, please note that by maximizing both sound sources, the sound after mixing becomes distorted. After mixing, set the both sound source volumes so that the sound is not distorted. Also, with PWM sound source, please note that because of a wider dynamic range over the FM sound source, the sound at peak time tends to get distorted when setting through an average sound volume.

21. Following target ver 3.0, WS, the volume product uses a different encoder and filter compared with target ver 2.0x. As a result, the screen display is somewhat different. Following target ver 3.0, WS, the volume product, WS, compared with ver 2.0x, has the following features:

1. Colors are somewhat more vivid.
2. Less blot (blur) in the picture.

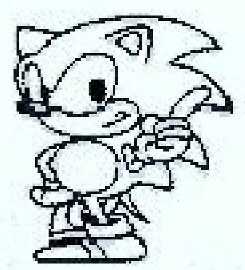
Make the final graphics check, after target ver 3.0, using WS, volume product.

22. When Z80 accesses PSG, make sure that bank settings at access time are values outside the $000000~$3FFFFF and $840000~9FFFFF ranges.

After Z80 accesses PSG, the MegaDrive internal circuitry bank automatically becomes $C000xx. However, after the Z80 access is over, the currently set bank values will be output to the 68000 side, and here, there will a period of time existing during which the signal that displays access on the 68000 side will be in an active mode.

With MegaDrive, there would be no problems; however, 32X compared to 68000 has a much faster reaction. Therefore, it would react to the above state, and if bank settings are in the $000000~$3FFFFF and $840000~9FFFFF ranges, 32X will misunderstand that access to the adapter has occurred. And soon after 68000 accesses this area, the misunderstood data could be passed to 68000. In addition, depending on the address at that time, the contents of 32X registers could be damaged.

Also, please note that because this phenomenon could occur at very different frequencies depending on the MegaDrive individual differences, the problem may not occur when the software is being checked.



SEGA™

SEGA OF AMERICA, INC.
Consumer Products Division

32X
Technical Information
Attachment 1

Doc. # MAR-42-072694

32X Technical Information Attachment 1

Details of 32X Technical Information 6

When using the 32X, and the RV bit of A15106H addess is "1", the normal operation of the Mega Drive can be affected after reset is applied. To correct this, the hardware has been changed so that the 32X system is reset by the watch-dog-timer output when VRES interrupt occurs on the SH2 (Master) side, and the RV bit is checked and is "1".

With respect to each application, the determination must be made whether or not the SH2 resets the system by checking the RV bit in the process within the VRES interrupt. On the MD side, the initial program operates if the system is reset, but because the MD side I/O isn't reset, the initial program moves onto application execution without executing the adapter usage procedure and determines whether or not the adapter usage procedure is performed; if the procedure hasn't been performed, it then must be performed .

Apart from the above procedure, it must be determined whether all processes at the start time are performed as a corrective measure when reset is applied repeatedly. With regard to applications that don't change the RV bit, the above operation is not required.

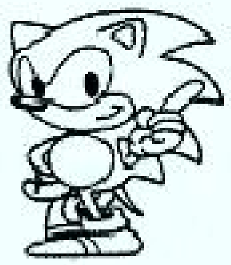
The above corrective measure will go into effect from the Ver. 2.1 (new board scheduled for release after Sept. 1994) development board. This problem cannot be avoided for development boards prior to Ver. 2.1 even if corrective action is taken by software.

The corrective measure with respect to the actual program is shown below. (From the sample program).

68000 Side Corrective Program Sample

```
*-------------------------------------------------------------------------*
:  Vector / Mega Drive ID / Mars Initial Program
          .include        source¥header.prg          ;  Mega Drive & Mars Header
          .include        source¥icd_mars.prg        ;  Sega indicated Initial Program & Security
*-------------------------------------------------------------------------*
                bcs         _error0                    ;  if cs = 1  then ID error
                                                       ;     or Self check error

                move.l      #0, initflug               ;  clear initial flag
                btst        #15, d0
                bne.b       VresStart                  ;  Reset with VRES Button?

                bra         _init
```

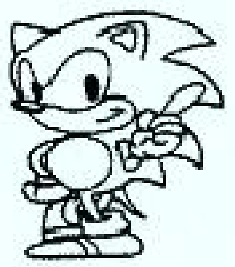
```
VresStart:
                lea         marsreg, a5
                btst.b      #ADEN, adapter (a5)
                bne         AdapterEnable           ; has 32X gone into effect?

*----           SUPER 32X Usage Procedure

                move.l      #0, comm8 (a5)

                lea         ?10, a0                 ; copy from ROM to WRAM
                lea         $ff0000, al

                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+
                move.l      (a0)+, (a1)+

                lea         $ff0000, a0
                jmp         (a0)                    ; jump workram

        ?10:
                move.b      #1, adapter (a5)        ; SUPER 32X Mode
                                                    ; SH2 reset - wait 10ms -
                lea         Restarticd, a0
                adda.l      #marsipl, a0
                jmp         (a0)                    ; jump ROM (+$880000)

Restarticd:
                lea         $a10000, a5
                move.l      #-64, a4
                move.w      #3900, d7               ; 8
                lea         marsipl+$6e4, a1
                jmp         (a1)                    ; jump icd_mars.prg ?res_wait

AdapterEnable:
                lea         marsreg, a5
                btst.b      #RES, adapter (a5)
                bne         _hotstart               ; SH2 reset canceled?
                bra.b       Restarticd              ; If not canceled reset once again
                                                    ; operate icd_mars.prg

*-----------------------------------------------------*
*       Main Program
*-----------------------------------------------------*

_init:
                lea         marsreg, a5
        ?w:
                cmp.l       #'M_OK' , comm0 (a5)    ; SH2 Master OK ?
                bne.b       ?w
        ?w1:
                cmp.l       #'S_OK' , comm4 (a5)    ; SH2 Slave OK ?
                bne.b       ?w1

                moveq       #0, d0                  ; SH2 Start
                move.l      d0, comm0 (a5)          ; Master
```



```
                move.l      d0, comm4 (a5)       ; Slave
                move.l      #"INIT", initflug
_hotstart:
                cmp.l       #"INIT", initflug    ; Has initial process ended ?
                bne.b       _init
                move.l      $880000, a7
                bsr         SysInit              ; Mega Drive Init
        ?start:
                move.w      #$2000, sr
                move.w      #$8164, reg_1 (a6)   ; Display On
                move.w      #$8164, _vdpreg      ; V Interrupt Enable
```

SH2 (Master) Side Corrective Program Sample

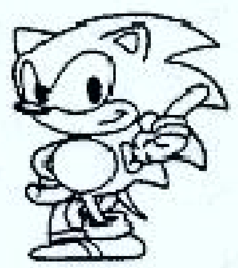
```
;-----------------------------------------------------------------*
;       SH2 (Master) Vector
;-----------------------------------------------------------------*

vector:
        .data.l         start                   ; Power On Reset PC
_stack:
        .data.l         M_STACK,                ; Power On Reset SP
+                       start,                  ; Manual Reset PC
+                       M_STACK                 ; Manual Reset SP

        .data.l         error0,                 ; General invalid command
+                       h'00000000,             ; System reserve
+                       error0,                 ; Slot invalid command
+                       h'20100400,             ; System reserve  (ICE Vector)
+                       h'20100420,             ; System reserve  (ICE Vector)
+                       error0,                 ; CPU address error
+                       error0,                 ; DMA address error
+                       error0,                 ; NMI
+                       error0                  ; User break

        .datab.l        19, h'00000000          ; System reserve
        .datab.l        32, error0              ; Trap command (User vector)

        .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)        ; for Correcting Interrupt
                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_L, r1)       ;
                mov.b       r0, @ (_FRC_H, r1)       ;

                mov         #h'_f2, r0               ; for Correcting VRES
                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         #h  e2, r0
                mov.b       r0, @ (_TOCR, r1)        ;
wait_md:
                mov.l       @ (comm0, gbr), r0       ; Combine Mega Drive
                                                     ; and timing

                cmp/eq      #0, r0
                bf          wait_md

                mov.l       #"SLAV",r1
wait_slave:
                mov.l       @ (comm8, gbr), r0       ; Combine SH2 Slave
                                                     ; and timing

                cmp/eq      r1, r0
                bf          wait_slave

                mov.l       #_SERIAL, r1
                mov         #b' 10000000, r0
                mov.b       r0, @r1                  ; Serial Mode Register
                mov         #74, r0
                mov.b       r0, @ (1, r1)            ; Bit Rate Register
                mov         #b' 00000000, r0
                mov.b       r0, @ (2, r1)            ; Serial Control Register
                mov.l       #4*74, r0
w_serial:
                nop
                dt          r0
                bf          w_serial
                mov         #b' 00100000, r0
                mov.b       r0, @ (2, r1)            ; Serial Control Register (start)
                mov         #0, r0
                mov.b       r0, @ (4, r1)
_hotstart:
                mov         #h' 20, r0
                ldc         r0, sr                   ; SH2 Interrupt Enable
                                |
                                |
                                |
                                |
```



```
;-----------------------------------------------------*
;
;        Interrupt Control
;
;-----------------------------------------------------*

m_int:
                push    0, 1
                sts.l   pr, @ - r15

                stc     sr, r0
                shlr2   r0
                and     #h' 3c, r0
                mov.l   #inttable, rl

                add     r1, r0
                mov.l   @r0, r1
                jsr     @r1
                nop

                lds.l   @r15+, pr
                pop     0, 1
                rte
                nop

                .align          4
inttable:
        .data.l             noret,                  ; Illegal Interrupt
+                           noret,                  ; Level 1
+                           noret,                  ; Level 2
+                           noret,                  ; Level 3
+                           noret,                  ; Level 4
+                           noret,                  ; Level 5
+                           pwmint,                 ; Level 6
+                           pwmint,                 ; Level 7
+                           cmdint,                 ; Level 8
+                           cmdint,                 ; Level 9
+                           hint,                   ; Level 10
+                           hint,                   ; Level 11
+                           vint,                   ; Level 12
+                           vint,                   ; Level 13
+                           vresint,                ; Level 14
+                           vresint,                ; Level 15

noret:
                rts
                nop

;-----------------------------------------------------*
;
;        VRES Interrupt
;
;-----------------------------------------------------*

vresint:
                mov.l       #_sysreg, r0
                ldc         r0, gbr

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

                mov.b       @ (dreqctl, gbr), r0        ; VRES corrective action
                tst         #RV, r0
                bf          mars_reset
                mov.l       #M_STACK - 8, r15        ; Stack change
                mov.l       #_hotstart, r0
                mov         r0, @r15                 ; PC change
                mov.w       #h f0, r0
                mov         r0, @ (4, r15)           ; SR mask

                mov.l       #_DMAOPERATION, r1
                mov         #0, r0
                mov.l       r0, @r1                  ; DMA Off

                mov.l       #_DMACHANNEL0, r1
                mov         #0, r0
                mov.l       r0, @r1

                mov.l       #b'0100010011100000, r1
                mov.l       r0, @r1                  ; Channel Control

                rte
                nop

mars_reset
                mov.l       #_FRT, r1                ; System Reset
                mov.b       @ (_TOCR, r1), r0        ;
                or          #h' 01, r0               ;
                mov.b       r0, @ (_TOCR, r1)        ;
vresloop:
                bra         vresloop

                                |
                                |
```

Corrective Program Sample of SH2 (Slave) Side

```
                                |
                                |

;--------------------------------------------------------*
;           Interrupt Control
;--------------------------------------------------------*

s_int:
                push        0, 1
                sts.l       pr, @-r15

                stc         sr, r0
                shlr2       r0
                and         #h' 3c, r0
                mov.l       #inttable, r1
                add         r1, r0
                mov.l       @r0, r1
                jsr         @r1
                nop

                lds.l       @r15+ pr
                pop         0, 1
                rte
                nop

                .align      4
```



```
inttable:
        .data.l          noret,                  ; Illegal Interrupt
+                        noret,                  ; Level 1
+                        noret,                  ; Level 2
+                        noret,                  ; Level 3
+                        noret,                  ; Level 4
+                        noret,                  ; Level 5
+                        pwmint,                 ; Level 6
+                        pwmint,                 ; Level 7
+                        cmdint,                 ; Level 8
+                        cmdint,                 ; Level 9
+                        hint,                   ; Level 10
+                        hint,                   ; Level 11
+                        vint,                   ; Level 12
+                        vint,                   ; Level 13
+                        vresint,                ; Level 14
+                        vresint                 ; Level 15

noret:
                rts
                nop

;-----------------------------------------------------*
;       VRES Interrupt
;-----------------------------------------------------*

vresint:
                mov.l     #_sysreg, r0
                ldc       r0, gbr

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

                mov.b     @ (dreqctl, gbr), r0         ; VRES corrective action
                tst       #RV, r0
                bf        vresloop

                mov.l     #S_STACK-8, r15              ; Stack change
                mov.l     #_hotstart, r0
                mov       r0, @r15                     ; PC change
                mov.w     #h f0, r0
                mov       r0, @ (4, r15)               ; SR mask

                mov.l     #_DMAOPERATION, r1
                mov       #0, r0
                mov.l     r0, @ r1                     ; DMA Off

                mov.l     #_DMACHANNEL0, r1
                mov       #0, r0
                mov.l     r0, @ r1
                mov.l     #b' 0100010011100000, r1
                mov.l     r0, @r1                      ; Channel Control

                rte
                nop
vresloop:
                bra       vresloop

                          |
                          |
```