APPENDIX D

DIRECTORY OF COLECOVISION SOFTWARE BULLETINS

BULLETIN NO. 0001
May 25, 1982

TO: DISTRIBUTION

cc: Eric Bromely
Marshall Caras
Robert Schenck

FROM: DAVID HWANG

SUBJECT: COLECOVISION SOFTWARE BULLETIN

The Colecovision Software Bulletin has been set up to assist Colecovision programming users to understand, maintain and develop the system/application software.

More specifically, its purposes are:

(1) Part of the continuing effort to document the operating system (currently OS_7:OS);

(2) Keep users updated regarding any patches and revisions of the operating system;

(3) Function as a user's library for information exchange. Any proven routines or modules which can be used as tools to facilitate software development will be properly documented here with author(s) duly credited.

BULLETIN NO. 0002

May 25, 1982

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: Z. SMITH/D. HWANG

SUBJECT: ERROR IN WRITE_VRAM ROUTINE

WRITE_VRAM has a problem:

- It works as advertised for byte counts less than 100H and for byte counts that are even multiples of 100H. For other values, it subtracts 100H from the actual byte count that is written.

- Cartridge programmers should deal with this problem (and corresponding problems it will cause in any OS routine that writes VRAM, except for WR_SPR_NM_TBL) by always sending numbers of bytes that are less than or even multiples of 100H.

- They should not deal with it by padding their byte counts as this may lead to cartridges that fail when the bug is fixed.

BULLETIN NO. 0003

June 7, 1982

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: Z. SMITH/D. HWANG

SUBJECT: ERROR IN OS SOUND PACKAGE

There is a bug in the OS sound software:

- The data structure PTR_TO_LST_OF_SND_ADDR, which takes up 11 RAM bytes, is not located in OS RAM above 73BAH as it should be, but instead has been placed in the cartridge programmer's RAM at 7020H. Cartridge programmers should avoid using RAM from 7020H thru 702AH when the sound software is in operation.

BULLETIN NO. 0004

June 7, 1982

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: Z. SMITH/D. HWANG

SUBJECT: A TECHNIQUE FOR TURNING OFF SONGS AND SOUNDS

The need has arisen for a safe way of turning off an individual "song" or sound event before its time. The most obvious method, which involves writing OFFH to the first byte of that song's sound area is not recommended, since it could lead to incompatibility later on if, for any reason, we wanted to change the length of the sound area data structure.

The safest method that I have been able to come up with is to declare a "null song" for each sound area which consists of nothing more than a non-repeating 64'th rest. If this song is allowed to be the highest priority song for a given area, playing it will have the effect of turning off whatever sound happened to be playing in that area previously.

BULLETIN NO. 0005

6/18/82

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: Z. Smith/D. HWANG

SUBJECT: BUG IN OS ACTIVATE ROUTINE

There is a bug in the OS Activate routine which surfaces when Activate is called on a Semi-Mobile object in Graphics Mode 1.

In this mode, Activate writes the pattern generators for a Semi-Mobile object to VRAM properly, but miscalculates the number and placement in VRAM of the corresponding color bytes when operating on generators in the upper half of the stable.

This leads to 2 problems:

- The upper half of the color table is not written by Activate.
- The color bytes intended for this half of the table are written elsewhere in VRAM possibly overwriting some other table.

Cartridge programmers should avoid using Activate to write pattern generators to VRAM in Graphics Mode 1 whenever possible. Or, if it is absolutely necessary to use Activate in this way they should count, first of all, on having to write the color table separately, and second, on guarding against the second problem listed above by isolating the color table.

BULLETIN NO. 0006

SEPTEMBER 17, 1982

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: K. LAGACE/D. HWANG

SUBJECT: RELEASE OF ADDITIONAL OS ENTRY POINTS
OS_SYMBOLS:OS REV. 1

Module Name	Address	Description	Inputs	Outputs	Regs. Destroyed
ADD816	001B1H	Adds 8 bit signed number in "A" to 16 bit number pointed to by "HL".	- 8 bit # in A - 16 bit # addr in HL	Altered 16 bit # at HL addr.	A,F,B
DECLSN	00190H	Decrements LSN of byte pointed to by "HL" without affecting MSN or "HL".	- 8 bit # addr. in HL	Altered 8 bit # at HL addr. Z flag if 0 C flag if -1 .	A,F
DECMSN	00198H	Decrements MSN of byte pointed to by "HL" without affecting LSN or "HL".	- 8 bit # addr. in HL	Altered 8 bit # at HL addr. Z flag if 0 C flag if -1	A,F
[illegible]OLSN	001A6H	Copies MSN of byte pointed to by "HL" to LSN of that byte.	- 8 bit # addr. in HL	MSN|LSN of # at HL addr. becomes MSN|MSN	A,F,B

FOR SOUND USE ONLY

ATN_SWEEP	0012FH	Creates attenuation sweeps by altering attenuation data stored in song data area.	See Sound Users Manual	See Sound Users Manual	All
FREQ_SWEEP	000FCH	Creates frequency sweeps by altering frequency data stored in song data area.	See Sound Users Manual	See Sound Users Manual	All
EFXOVER	002EEH	See Sound Users	See Sound Users	See Sound Users	All
LEAVE_EFFECT	001D5H	See Sound Users	See Sound Users	See Sound Users	All

BULLETIN NO. 0007

OCTOBER 21, 1982

TO: DISTRIBUTION

cc: Robert Schenck

FROM: ARD SOFTWARE ENGINEERING

SUBJECT: HEADER:UTL REV. 3

Released and supported on HP64000 systems under USERID:UTL is the new revision of HEADER which has been adopted by Software Engineering as part of standard documentation for any module and program developed in house (Ref. R. Jepson's memo July 27, 1982).

The SETHEAD:UTL has also been updated to support HEADER Rev. 3

M E M O R A N D U M

NO. 0008
OCTOBER 27, 1982

TO: DISTRIBUTION cc: Robert Schenck

FROM: MUSIC AND SOUND DEPT./D. HWANG

SUBJECT: MUSIC TABLES

LST_OF_SND_ADDRS has formerly been used in all games to denote the starting address of a list of pointers to song tables and work areas. This label will not be used in the future games. Instead a label with postfix "NOTES" will be used.

For example, in the upcoming games:

DONKEY KONG JR	will use	KONGJRNOTES
OMEGA RACE	will use	OMEGANOTES
GORF	will use	GORFNOTES

M E M O R A N D U M

NO. 0009
OCTOBER 27, 1982

TO: DISTRIBUTION

cc: Robert Schenck

FROM: MUSIC AND SOUND DEPT./D. HWANG

SUBJECT: SONGBIRD FILE

Effective immediately all work pertaining to music and sounds will be done in the SONGBIRD file. To play a song, a call to a descriptive label, which is supplied by the music group, will be used. For example:

```
CALL   BELL_SOUND
```

Where BELL_SOUND is a global label in the SONGBIRD file which will contain all that is necesary to play that sound or song. This one call approach is replacing the former procedure such as:

```
LD     B,3      ; THE SONG NUMBER
CALL   PLAY_IT
LD     B,4
CALL   PLAY_IT
```

Within the SONG_BIRD file, song numbers will be EQUAT to descriptive labels instead of using absolute numbers.

BULLETIN NO. 0010

OCTOBER 27, 1982

TO: DISTRIBUTION

cc: Eric Bromley
Marshall Caras
Robert Schenck

FROM: R. JEPSON/D. HWANG

SUBJECT: INTERRUPT HANDLING ROUTINES

Due to the hardware configuration of ColecoVision, certain routines which write to VRAM, such as WRITE-VRAM, could have undetermined results if they are interrupted during execution.

One possible solution to this problem is a defer interrupt routine. The function of this routine is to hold off (defer) the servicing of the interrupt until the cartridge program believes it is safe to service it.

An example of a defer interrupt service is attached using a binary semifore type construct. Note that the routines do not stop the actual interrupt (they can't) but just hold off processing the interrupt until later.

The key point to the routine is that another interrupt cannot occur until the VDP status register is read.

An example of the use of DEF_INT:

```
        LD      IX,WRITE_VRAM       ;IX gets routine that the
                                    ;cartridge program does
                                    ;not want interrupted
        CALL    DEF_INT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE
                       1 "Z80"
                       3  NAME ^Rev 0 Def_int, Nmi_int^
                       4
                       5 DESCRIPTION MACRO
                       6         .GOTO ENDESCRIPTION
                       7
                       8  Auther:          Ineventions
                       9  Userid:          C16
                      10  Starting date:   May 1982
                      11
                      12  From release Date:
                      13  From release Rev:
                      14
                      15  Header Rev: 1
                      16
                      17 ***********************************************************************
                      18 *
                      19 *     DEF_INT          NMI_INTERRUPT                                  *
                      20 *
                      21 *                  Coleco Vision Software Engineering                 *
                      22 *
                      23 ***********************************************************************
                      24
                      25         This file contains two modules, DEF_INT and NMI_INTERRUPT.
                      26     DEF_INT is a utility routine which is called with a
                      27     routine as an input parameter and will defer the interrupt while
                      28     the given routine is executing.  NMI_INTERRUPT is the routine vectored
                      29     to when a NMI interrupt is generated by the VDP hardware.
                      30
                      31        Rev History (one line note indicating the change)
                      32
                      33        Rev.    Date            Name            Change
                      34         1
                      35         0      12sept1207a
                      36
                      37
                      38
                      39 ENDESCRIPTION:
                      40         MEND
                      41
                      42 ;Inputs:
                      43
                      44 ;       IX = Address of routine to be called
                      45
                      46         GLB     INT_DEF_BYTE
                      47
                      48         DATA
0000                  49 INT_DEF_BYTE:
0000                  50         DEFS    1
                      51
                      52    *-------------------------------------------------------------------
                      53    * INT     |         |         |         |         |         |  INTER  |
                      54    * OCCURED |         |         |         |         |         |  DEFER  |
                      55    *---------|---------|---------|---------|---------|---------|---BIT---|
                      56    *    7         6         5         4         3         2         1         0
                      57
                      58         PROG
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
 59
 60 ;Outputs:
 61         GLB     NMI_INTERRUPT
 62         GLB     DEF_INT
 63
 64 ;Calls:
 65 ;       routine passed in IX in the case of DEF_INT
 66 ;OS Calls
 67         EXT     VDP_STATUS_BYTE
 68         EXT     READ_REGISTER
 69
 70 COMMENT MACRO
 71         .GOTO ENDCOMMENT
 72
 73 ****************************************************************************
 74 *
 75 *     DEF_INT                                                    Rob
 76 *
 77 ****************************************************************************
 78
 79 ;Psuedo code
 80
 81  Procedure DEF_INT (^OS_routine);
 82     Push AF                 (In case it has an input parameter)
 83     INT_DEF_BYTE := DEFER; (set bit 0)
 84     Pop  AF                 (With input parameter to OS routine)
 85     Save HL                 (in case of input parameter)
 86     HL := Return address
 87     Exchange Return address with saved HL value
 88     Goto OS routine passed as input (in IX)
 89        .       .
 90        .(do os routine processing) (and possibly an interrupt will occur
 91        .         but not be processed.)
 92     (return from OS routine)
 93     INT_DEF_BYTE := NOT (DEFER); (reset bit 0)
 94     Check for INTERRUPT_OCCURRED in INT_DEF_BYTE (bit 7)
 95     INT_DEF_BYTE := NOT (DEFER,INTERRUPT_OCCURRED);  (0)
 96     IF (INTERRUPT_OCCURRED in INT_DEF_BYTE) (From previous test)  (bit 7)
 97         then
 98             NMI_INTERRUPT;
 99
100  END; (DEF_INT)
101
102 ENDCOMMENT;
103         MEND
104
105 ****************************************************************************
106
107         PROC
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    130 COMMENT2 MACRO
                    131         .GOTO ENDCOMMENT
                    132
                    133
                    134 ************************************************************************
                    135 *
                    136 *                    NMI_INTERRUPT              Rob                    *
                    137 *                                                                      *
                    138 ************************************************************************
                    139
                    140     We get here whenever VDP interrupts (every 1/60 sec.).  The deffered VRAM writes
                    141     take place here and the interrupt driven sound, keyboard scan and timer
                    142     routines are also exercised.  At the end of the routine, the VDP status
                    143     register is read and its contents are saved in STAT_SAVE.
                    144
                    145     Before the routine runs, it checks interrupt defer bit of INT_DEF_BYTE (bit 0).  If this
                    146     bit is set, it is an indication that a routine is in progress which accesses the VDP and
                    147     therefore should not be interrupted by another routine which also accesses the
                    148     VDP.  Interrupt occured bit of INT_DEF_BYTE (bit 7) is then set and the routine returns
                    149     without performing any other functions.
                    150
                    151  Psuedo code
                    152
                    153     Procedure NMI_INTERRUPT
                    154         Save Accumulator
                    155         Save all registers except Accumulator
                    156         Save alternate registers
                    157         If DEFER in INT_DEF_BYTE (bit 0)
                    158             then INT_DEF_BYTE := INT_DEF_BYTE + [INTERRUPT_OCCURED]; Set bit 7
                    159
                    160             else
                    161                 Begin
                    162                     Process interrupt routines
                    163                 End;
                    164         Restore alternate registers
                    165         Restore all registers except Acc
                    166         Read VDP status register
                    167         Restore Acc
                    168
                    169     End;    (NMI_INTERRUPT)
                    170
                    171 END_COMMENT;
                    172             MEND
                    173
                    174             PROG
```

BULLETIN NO. 11
DECEMBER 22, 1982

TO: DISTRIBUTION

FROM: ARD SOFTWARE ENGINEERING Daidyga Hong

cc: Eric Bromley
Robert Schenck
Marshall Caras
Tom Helmer

SUBJECT: RELEASE OF COLECOVISION PROGRAMMERS MANUAL REV. 5

The ColecoVision Programmer's Manual Rev. 5 has been released. This manual is written for the applications programmer and is intended as both a day-to-day reference source as well as a training document for programmers new to ColecoVision.

This new edition contains the overview for both hardware and software. Subsequently, detail descriptions are given in the areas of:

Graphics Generation Software
Interrupt Handling
Timing
Controller Software
Sound Generation Software
Boot up Software and Utilities
Defined Reference Locations

The Rev. 5 manual pertains to the current production OS_7. Fundamental knowledge of the OS is presented in the manual without elaborating on application examples and design approaches. These materials will be documented in the proposed ColecoVision Applications Manual, scheduled to be released in second quarter 1983.

In the Appendix B you will find the graphics documentation (Rev. 1.0) has been updated with addition of materials describing PUT_SPRIT and PUT_COMPLEX.

The Sound documentation also received updates in the form of Notes and Errata attached at the end of Appendix C.

User feedback should be addressed to the Manager of Software Engineering of Coleco ARD. All adopted changes will be brought to your attention via ColecoVision Bulletin announcements.

This manual is confidential and should not be copied. All releases have to be signed out through the ARD Engineering secretary S. Rakowski.

DISTRIBUTION:

C. Baldyga
R. Dionne
A. Godfrey
L. Gray
C. Hager
R. Harris
R. Jepson
D. Jonker
K. Lagace
J. Michaels
M. Minto
A. Nguyen
L. Olbrych
D. Schulze
D. Stern
D. Thompson

CONFIDENTIAL

COLECO INDUSTRIES, INC. Executive Offices: 945 ASYLUM AVENUE HARTFORD, CT 06105 (203) 278-0280

ColecoVision Software Bulletin

BULLETIN NO. 0012
March 17, 1983

TO: DISTRIBUTION

FROM: ARD SOFTWARE ENGINEERING DKH KAL

RE: CORRECTIONS IN REGARD TO BULLETIN NO. 0004

---

(1) The statement that "Sound Data Areas are off limits to programmers" is not true.

(2) The "Null Song" method wastes CROM space. Writing OFFH to the first byte of the song's sound area IS recommended.

Since the ColecoVision Operating System turns off sounds by placing OFFH into the first byte of the Sound Data Areas anyway and changing the data structures of the Sound Data Areas would entail changing the operating system. It has been proven that the above method is the fastest and most direct way to abort sounds.

The "null song" method may still be used, but each additional song uses at least five bytes of CROM; four for the LST_OF_SND_ADDRS and one for the END code.

DISTRIBUTION:

C. Baldyga
R. Dionne
A. Godfrey
L. Gray
C. Hager
R. Harris
L. Henderson
R. Jepson
D. Jonker
K. Lagace
J. Michaels
M. Minto
A. Nguyen
L. Olbrych
R. Rizzo
D. Schulze
D. Stern
D. Thompson
B. Zawislak

CC: E. Bromley
R. Schenck
C. M. Caras
T. Helmer

CONFIDENTIAL

COLECO INDUSTRIES, INC. Executive Offices: 945 ASYLUM AVENUE HARTFORD, CT 06105 (203) 278-0280

ColecoVision Software Bulletin

Bulletin No. 0013
April 4, 1983

TO: Distribution

FROM: ARD Software Engineering DKH RFS

RE: Release of Additional ColecoVision OS Entry Points

The following is a list of additional entry points to the ColecoVision OS ROM.

PX_TO_PTRN_POS	EQU	07E8H
PUT_FRAME	EQU	080BH
GET_BKGRND	EQU	0898H
CALC_OFFSET	EQU	08C0H

Attached is a brief description of the routines which correspond to the entry points released.

DISTRIBUTION

C. Baldyga
R. Dionne
L. Gray
C. Hager
R. Harris
L. Henderson
R. Jepson
D. Jonker
K. Lagace
J. Michaels
M. Minto
A. Nguyen
L. Olbrych
R. Rizzo
B. Rochette
D. Schulze
D. Stern
D. Thompson
B. Zawislak

CC: E. Bromley
R. Schenck
D. Hwang
C. M. Caras
T. Helmer
File

CONFIDENTIAL

Here are the graphic subroutines which would be useful to have access to, along with a brief description of what each one does.

****

PX_TO_PTRN_POS (Pixel to pattern plane position)
(entry point xxxxH)

This routine divides the 16 bit signed value in the DE register pair by 8. An 8 bit signed result is returned in register E. Results of less than -127 are returned as -128, results of greater than +126 are returned as +127.

If this routine is passed the X(or Y) pixel coordinate position of a point on the pattern plane, the X(or Y) coordinate in pattern positions will be returned.

INPUT: DE = N (16 bit signed number)

OUTPUT:		
	N/8 < -128	E = -128
	-128 <= N/8 <= 127	E = N/8
	N/8 > +126	E = +127

REGISTERS AFFECTED:

FLAGS
DE

****

PUTFRAME
(entry point xxxxH)

PUTFRAME moves data from cpu RAM to the Pattern Name Table in VRAM. The data is assumed to be an array of Pattern Generator Names which when moved to the Pattern Name Table, will produce a rectangular graphic, or frame, composed of the patterns specified by these Pattern Generator Names. The array must be arranged in row major order.

The dimensions of array are passed to the routine in the BC register pair. These dimensions also define the height and width (in pattern plane positions) of the frame when displayed.

The upper left corner of the frame will appear on the pattern plane at a position determined by Y_PAT_POS and X_PAT_POS which are passed in the DE register pair. Y and X_PAT_POS are row and column coordinates in pattern plane positions as measured from the upper left corner of the pattern plane. Y and X_PAT_POS are interpreted as 8 bit signed values and, therefore, the corner of the frame may placed anywhere within or outside the boundaries

of the pattern plane. Therefore, the frame itself may be placed partially off screen in any direction.

The HL register pair must contain the address of the start of the array of pattern names.

INPUT: HL = Address of array in CPU RAM
B = Y dimension of array and Y_EXTENT of frame
C = X dimension of array and X_EXTENT of frame
D = Y_PAT_POS of upper left corner of frame
E = X_PAT_POS of upper left corner of frame

OUTPUT: Modifies VRAM name table

REGESTERS AFFECTED:

All registers used

As an example, if an array exists in CPU memory space which looks like...

```
ARRAY:  DB      0,1,2,3,4,5
```

and the first six pattern generators in VRAM have been initialized with the following patterns...

Pattern Generator #	Graphic
0	A
1	B
2	C
3	D
4	E
5	F

Then the following code sequence...

```
        LD HL,ARRAY
        LD B,2          ;B := Y_EXTENT
        LD C,3          ;C := X_EXTENT
        LD D,2          ;D := Y_PAT_POS
        LD E,-1         ;E := X_PAT_POS
        CALL PUT_FRAME
```

will produce this display...

```
            0 X_PAT_POS ->
Y_PAT_POS   . . . . . .
        V  0. . . . . .
           . . . . . .
           .B.C. . . .
           .E.F. . . .
           . . . . . .
```

(diagram of upper left corner of pattern plane)

Note: Patterns A and D are not seen, since they would be to the left of the left-hand edge of the pattern plane.

****

GET_BKGRND
(entry point xxxxH)

This routine is the inverse of the PUT_FRAME routine described above. GET_BKGRND moves an array of names from the pattern name table in VRAM into CPU RAM. The dimensions of the array and the position of the upper left corner of the frame it defines, are passed to the routine in same manner as in PUT_FRAME. The names are moved to the location in CPU RAM specified by the contents of the HL register pair.

If part of the frame extends beyond the pattern plane, the names that correspond to positions which are not on the pattern plane will not be defined.

INPUTS: HL = Destination address in CPU RAM to which names will be moved
B = Y_EXTENT of frame
C = X_EXTENT of frame
D = Y_PAT_POS of upper left corner of frame
E = X_PAT_POS of upper left corner of frame

OUTPUTS: CPU RAM from HL to HL+(B*C)-1 filled with names from pattern name table

REGISTERS AFFECTED:

All registers used

****

CALC_OFFSET
(entry point xxxxH)

This routine calculates the offset from the start of the pattern name table corresponding to a pattern plane position specified by the coordinates Y_PAT_POS and X_PAT_POS.

The coordinates are passed to, and the result is passed back in the DE register pair.

INPUTS: D = Y_PAT_POS
E = X_PAT_POS

OUTPUTS: DE = Offset from start of pattern name table

REGISTERS AFFECTED:
FLAGS
DE

COLECO
INDUSTRIES, INC.

Executive Offices:
945 ASYLUM AVENUE HARTFORD, CT 06105 (203) 278-0280

ColecoVison Software Bulletin

Bulletin No. 0014
April 12, 1983

TO: Distribution

FROM: ARD Software Engineering DKH RFJ

RE: OS_SYMBOLS Rev.4

Attached please find a listing of OS_SYMBOLS Rev. 4. This listing holds al ColecoVision OS reserved data entry points released to date.

Attachment

Distribution:

C. Baldyga
R. Dionne
L. Gray
C. Hager
R. Harris
L. Henderson
R. Jepson
D. Jonker
K. Lagace
J. Michaels
J. Milano
M. Minto
A. Nguyen
L. Olbrych
R. Rizzo
B. Rochette
D. Schulze
D. Stern
D. Taylor
D. Thompson
B. Zawislak

CC:

E. Bromley
C. M. Caras
T. A. Helmer
D. Hwang
R. Schenck
File

CONFIDENTI

LOCATION OBJECT CODE LINE SOURCE LINE

```
1  "Z80"
3  NAME "Rev 4 - RFJ"
4
5  DESCRIPTION MACRO
6          .GOTO ENDESCRIPTION
7
8  Author:          Zac Smith
9  Userid:          OS
10 Starting date:   13may1982
11 Header Rev: 1
12 ********************************************************************
13 *                                                                  *
14 *                                OS_SYMBOLS                        *
15 *                ColecoVision Operating System                     *
16 *                     Software Engineering                         *
17 *            Advanced Research and Development                     *
18 *                       Coleco Industries                          *
19 *                                                                  *
20 *                      *** Confidential ***                        *
21 *                                                                  *
22 ********************************************************************
23
24      List of access points to the Colecovision Operating system ROM.
25  Only those points listed in this file have been approved as absolute
26  locations of which the cartridge developer can access the OS rom.
27      Additionally, access to any memory locations indirectly, or by
28  offset to locations defined herein is denied accept where defined by
29  the ColecoVision Programmer's Manual ( current rev 85 ).
30
31  List of OS symbols in alphabetical order with defining and referencing
32      modules (if any).
33
34       Rev History (one line note indicating the change)
35
36       Rev.    Date          Name       Change
37        4      13apr1359     Rob        Remove Zaxxon related documentation
30                                         in preparation for re-release of
39                                         this file for general distribution
40               11apr1626     Rob        Added PUTFRAME (no underline) to
41                                         match label in OS listing.  Kept
42                                         PUT_FRAME due to Software Bulletin
43                                         released.
44               11apr 900     Rob        Updated Header to expand the
45                                         description of this file.
46                                         GLBed locations added in rev 3
47        3      05apr1444     Rob        Added locations:
48                                         PX_TO_PTRN_POS  PUT_FRAME
49                                         GET_BKGRND      CALC OFFSET
50        2      13sept1114p Rob          Added documentation specific to
51                                         Zaxxon Development.
52        1      2sept1153p  Ken Lagace Added 9 SOUND OS equates
53        0      13may       Zac Smith  Initial Jump table equates
54                                         DATE     : 5/13/82
55                                         FOR REV : 5 (OS 5:OS)
56 ENDESCRIPTION:
57         MEND
```

CONFIDENTIAL

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      59
                      60 ;Symbol              Absolute  Partial Xref of routines used
                      61 ;Name                Address   by other OS routines
                      62 ;-----------------------------------------------------------
                      63                                ;Start of defined reference points
(1FF7)                64 ACTIVATE          EQU 01FF7H
(1F64)                65 ACTIVATEP         EQU 01F64H
(01B1)                66 ADD816            EQU 001B1H
(0069)                67 AMERICA           EQU 00069H
(006A)                68 ASCII_TABLE       EQU 0006AH
(012F)                69 ATN_SWEEP         EQU 0012FH
(08C0)                70 CALC_OFFSET       EQU 008C0H
(8000)                71 CARTRIDGE         EQU 08000H
(8008)                72 CONTROLLER_MAP    EQU 08008H    ;CONTROLLE:OS
(0190)                73 DECLSN            EQU 00190H
(019B)                74 DECMSN            EQU 0019BH
(1F79)                75 DECODER           EQU 01F79H
(73C6)                76 DEFER_WRITES      EQU 073C6H    ;PUT_OBJEC:OS
(02EE)                77 EFXOVER           EQU 002EEH
(1F73)                78 ENLARGE           EQU 01F73H
(1D6C)                79 ENLRG             EQU 01D6CH
(1F82)                80 FILL_VRAM         EQU 01F82H    ;GAME_OPT:OS
(1FCA)                81 FREE_SIGNAL       EQU 01FCAH
(1F9D)                82 FREE_SIGNALP      EQU 01F9DH
(00FC)                83 FREQ_SWEEP        EQU 000FCH
(8024)                84 GAME_NAME         EQU 08024H    ;LOGO:OS
(1F7C)                85 GAME_OPT          EQU 01F7CH
(0898)                86 GET_BKGRND        EQU 00898H
(1FBB)                87 GET_VRAM          EQU 01FBBH    ;PUT_MOBIL:OS  PUT_SPR:OS
(1F8E)                88 GET_VRAMP         EQU 01F8EH
(1FC1)                89 INIT_SPR_ORDER    EQU 01FC1H
(1F94)                90 INIT_SPR_ORDERP   EQU 01F94H
(1FB8)                91 INIT_TABLE        EQU 01FB8H    ;GAME_OPT:OS  LOGO:OS`
(1F8B)                92 INIT_TABLEP       EQU 01F8BH
(1FC7)                93 INIT_TIMER        EQU 01FC7H
(1F9A)                94 INIT_TIMERP       EQU 01F9AH
(1FE5)                95 INIT_WRITER       EQU 01FE5H
(1FAF)                96 INIT_WRITERP      EQU 01FAFH
(801E)                97 IRQ_INT_VECT      EQU 0801EH
(01D5)                98 LEAVE_EFFECT      EQU 001D5H
(1F7F)                99 LOAD_ASCII        EQU 01F7FH    ;GAME_OPT:OS
(8002)               100 LOCAL_SPR_TBL     EQU 08002H    ;TABLE_MA:OS
(1F85)               101 MODE_1            EQU 01F85H    ;GAME_OPT:OS
(01A6)               102 MSNTOLSN          EQU 001A6H
(73C7)               103 MUX_SPRITES       EQU 073C7H    ;TABLE_MA:OS
(8021)               104 NMI_INT_VECT      EQU 08021H
(006C)               105 NUMBER_TABLE      EQU 0006CH
(1FF1)               106 PLAY_IT           EQU 01FF1H
(1FB5)               107 PLAY_ITP          EQU 01FB5H
(1F61)               108 PLAY_SONGS        EQU 01F61H
(1FEB)               109 POLLER            EQU 01FEBH
(080B)               110 PUTFRAME          EQU 0080BH
(1FFA)               111 PUTOBJ            EQU 01FFAH    ;PUT_CMPLX:OS
(1F67)               112 PUTOBJP           EQU 01F67H
(080B)               113 PUT_FRAME         EQU 0080BH
(1FBE)               114 PUT_VRAM          EQU 01FBEH    ;GAME_OPT:OS  LOGO:OS  PUT_MOBIL:OS  PUT_SPR:OS
(1F91)               115 PUT_VRAMP         EQU 01F91H
```

CONFIDENTIAL

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    116 PX_TO_PTRN_POS  EQU 07E8H
                    117 RAND_GEN        EQU 01FFDH
                    118 RAND_NUM        EQU 073C8H
                    119 READ_REGISTER   EQU 01FDCH    ;LOGO:OS
                    120 READ_VRAM       EQU 01FE2H    ;PUT_MOBIL:OS
                    121 READ_VRAMP      EQU 01FACH
                    122 REFLECT_HORIZON EQU 01F6DH
                    123 REFLECT_VERTICA EQU 01F6AH
                    124 REQUEST_SIGNAL  EQU 01FCDH
                    125 REQUEST_SIGNALP EQU 01FA0H
                    126 ROTATE_90       EQU 01F70H
                    127 RST_10H_RAM     EQU 0800FH
                    128 RST_18H_RAM     EQU 08012H
                    129 RST_20H_RAM     EQU 08015H
                    130 RST_28H_RAM     EQU 08018H
                    131 RST_30H_RAM     EQU 0801BH
                    132 RST_8H_RAM      EQU 0800CH
                    133 SOUND_INIT      EQU 01FEEH
                    134 SOUND_INITP     EQU 01FB2H
                    135 SOUND_MAN       EQU 01FF4H
                    136 SPRITE_ORDER    EQU 08004H    ;TABLE_MA:OS
                    137 STACK           EQU 073B9H
                    138 START_GAME      EQU 0800AH    ;LOGO:OS
                    139 TEST_SIGNAL     EQU 01FD0H
                    140 TEST_SIGNALP    EQU 01FA3H
                    141 TIME_MGR        EQU 01FD3H
                    142 TURN_OFF_SOUND  EQU 01FD6H
                    143 UPDATE_SPINNER  EQU 01F88H
                    144 VDP_MODE_WORD   EQU 073C3H    ;GRAPHICS:OS  VD_DRIVER:OS  TABLE_MA:OS  PUT_MOBIL:OS  ACT2:OS
                    145 VDP_STATUS_BYTE EQU 073C5H
                    146 WORK_BUFFER     EQU 08006H    ;GRAPHICS:OS  PUT_MOBIL:OS  PUT_SPR:OS  PUTSEMI2:OS  ACT2:OS
                    147 WRITER          EQU 01FE8H
                    148 WRITE_REGISTER  EQU 01FD9H    ;GAME_OPT:OS  LOGO:OS
                    149 WRITE_REGISTERP EQU 01FA6H
                    150 WRITE_VRAM      EQU 01FDFH    ;GAME_OPT:OS  LOGO:OS  PUT_MOBIL:OS
                    151 WRITE_VRAMP     EQU 01FA9H
                    152 WR_SPR_NM_TBL   EQU 01FC4H
                    153 WR_SPR_NM_TBLP  EQU 01F97H
                    154                           ;End of defined reference points
```

LOCATION OBJECT CODE LINE SOURCE LINE

```
                    156
                    157                 |The following defines each access point
                    158  GLB ACTIVATE   | as Global.
                    159  GLB ACTIVATEP
                    160  GLB ADD816
                    161  GLB AMERICA
                    162  GLB ASCII_TABLE
                    163  GLB ATN_SWEEP
                    164  GLB CALC_OFFSET
                    165  GLB CARTRIDGE
                    166  GLB CONTROLLER_MAP
                    167  GLB DECLSN
                    168  GLB DECMSN
                    169  GLB DECODER
                    170  GLB DEFER_WRITES
                    171  GLB EFXOVER
                    172  GLB ENLARGE
                    173  GLB ENLRG
                    174  GLB FILL_VRAM
                    175  GLB FREE_SIGNAL
                    176  GLB FREE_SIGNALP
                    177  GLB FREQ_SWEEP
                    178  GLB GAME_NAME
                    179  GLB GAME_OPT
                    180  GLB GET_BKGRND
                    181  GLB GET_VRAM
                    182  GLB GET_VRAMP
                    183  GLB INIT_SPR_ORDER
                    184  GLB INIT_SPR_ORDERP
                    185  GLB INIT_TABLE
                    186  GLB INIT_TABLEP
                    187  GLB INIT_TIMER
                    188  GLB INIT_TIMERP
                    189  GLB INIT_WRITER
                    190  GLB INIT_WRITERP
                    191  GLB IRQ_INT_VECT
                    192  GLB LEAVE_EFFECT
                    193  GLB LOAD_ASCII
                    194  GLB LOCAL_SPR_TBL
                    195  GLB MSNTOLSN
                    196  GLB MODE_1
                    197  GLB MUX_SPRITES
                    198  GLB NMI_INT_VECT
                    199  GLB NUMBER_TABLE
                    200  GLB PLAY_IT
                    201  GLB PLAY_ITP
                    202  GLB PLAY_SONGS
                    203  GLB POLLER
                    204  GLB PUTFRAME
                    205  GLB PUTOBJ
                    206  GLB PUTOBJP
                    207  GLB PUT_FRAME
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     213  GLB READ_REGISTER
                     214  GLB READ_VRAM
                     215  GLB READ_VRAMP
                     216  GLB REFLECT_HORIZON
                     217  GLB REFLECT_VERTICA
                     218  GLB REQUEST_SIGNAL
                     219  GLB REQUEST_SIGNALP
                     220  GLB ROTATE_90
                     221  GLB RST_10H_RAM
                     222  GLB RST_18H_RAM
                     223  GLB RST_20H_RAM
                     224  GLB RST_28H_RAM
                     225  GLB RST_30H_RAM
                     226  GLB RST_8H_RAM
                     227  GLB SOUND_INIT
                     228  GLB SOUND_INITP
                     229  GLB SOUND_MAN
                     230  GLB SPRITE_ORDER
                     231  GLB STACK
                     232  GLB START_GAME
                     233  GLB TEST_SIGNAL
                     234  GLB TEST_SIGNALP
                     235  GLB TIME_MGR
                     236  GLB TURN_OFF_SOUND
                     237  GLB UPDATE_SPINNER
                     238  GLB VDP_MODE_WORD
                     239  GLB VDP_STATUS_BYTE
                     240  GLB WORK_BUFFER
                     241  GLB WRITER
                     242  GLB WRITE_REGISTER
                     243  GLB WRITE_REGISTERP
                     244  GLB WRITE_VRAM
                     245  GLB WRITE_VRAMP
                     246  GLB WR_SPR_NM_TBL
                     247  GLB WR_SPR_NM_TBLP

Errors=    0
```