VECTREX

PROGRAMMER'S

MANUAL

VOLUME II

DETAILED DESCRIPTION OF
EXECUTIVE SUBROUTINES

REV	DATE	PROG	COMMENT(S)
-	MM/DD/YY	JJH	Initial Release

[FDT: Revision 1-1 , 12/06/99]

ABSAB (ABSVAL)

Description:

Form the absolute value of registers ‘A’ & ‘B’.

Entry Address = $F584

Maximum Stack Requirements = 2 bytes

Entry Values

A = Value to be made positive
B = Value to be made positive

Return Values

A = Absolute value of entry ‘A’
B = Absolute value of entry ‘B’

Executive subroutines utilized

ABSB

Comments

An entry value of $80 will not evaluate properly

ABSB (AOK)

Description:
Form the absolute value of register ‘B’

Entry Address = $F58B

Maximum Stack Requirements = 2 bytes

Entry Values
B = Value to be made positive

Return Values
B = Absolute value of entry ‘B’

Comments
An entry value of $80 will not evaluate properly

ACTGND (ZEREF)

Description:

Set active ground sample / hold to zero volts.

Entry Address = $F35B

Maximum Stack Requirements = 2 bytes

Entry Values

DP = $D0

Return Values

A = $03
B = $01

Control register modifications

CNTRL, DAC

Comments

The active ground sample / hold should be set approximately every 16 vectors (this really needs to be determined by trial and error).

ADOT

Description:
Position and then draw dot

Entry Address = $EA5D

Maximum Stack Requirements = 10 bytes

Entry Values
Y = Absolute ‘Y:X’ position
DP = $D0

Return Values
Same as entry values

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized
ABSAB, ABSB, ACTGND
DOT, DOTAB
POSITN
ZERGND

ADROT (DIFROT)

Description:

Rotate ‘DIFFY’ style list

List Description:

Byte 0 / 1 = Vector #1 (Y:X)

- .

Byte n / n+ 1 = Vector #n (Y:X)

Entry Address = $F616

Maximum Stack Requirements = 9 bytes

Entry Values

X = ‘DIFFY’ list pointer
U = Destination buffer pointer

ANGLE = Angle of rotation ($00 - $3F)
LIST = Number of vectors - 1

Return Values

A = $00
B = Destroyed
X = Entry value + 1
U = Entry value + 1

LIST = $00

Executive storage modifications

LAG, LEG
WCSINE, WSINE

Executive subroutines utilized

APROT
COSINE
DPRAM
LCSINE, LSINE
MCSINE, MSINE
SINCOS, SINE

ALNROT (ROTAR)

Description:
Rotate a single line

Entry Address = $F603

Maximum Stack Requirements = 8 bytes

Entry Values
A = Initial ‘Y’ value
DP = $C8

ANGLE = Angle of rotation ($00 - $3F)

Return Values
A = Rotated ‘Y’ vector value
B = Rotated ‘X’ vector value

Executive storage modifications
LAG, LEG
WCSINE, WSINE

Executive subroutines utilized
COSINE
LSINE, LCSINE
SINCOS, SINE

APACK

Description:

Position and draw packet

List Description:

```
Byte 0  /   1   /  2    = Vector #1 (C:Y:X)
     -    -   -             -
     n / n+1 / n+2 = Vector #n (C:Y:X)
     n+3             = $01 (packet terminator)
```

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $EA7F

Maximum Stack Requirements = 10 bytes

Entry Values

B = Zoom value (scale factor)
X = ‘Packet’ list pointer
Y = Absolute ‘Y:X’ position
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

Same as entry values

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
CZERO
PACKET, POSITD, POSITN
TPACK
ZERGND

APROT (POTATE)

Description:

Rotate ‘Packet’ style list

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

Entry Address = $F622

Maximum Stack Requirements = 9 bytes

Entry Values

X = ‘Packet’ list pointer
U = Destination buffer pointer

ANGLE = Angle of rotation ($00 - $3F)

Return Values

A = ‘Packet’ terminator value
B = Destroyed
X = End of ‘Packet’ list + 1
U = End of destination buffer + 1

LIST = $00

Executive storage modifications

LAG, LET
WCSINE, WSINE

Executive subroutines utilized

DPRAM
COSINE
LCSINE, LSINE
MCSINE, MSINE
SINCOS, SINE

ASMESS

Description:

Position and draw raster message

Message List Description:

Byte 0 – n = Raster message string ($20 - $6F)

n + 1 = Raster terminator ($80)

Entry Address = $EAA8

Maximum Stack Requirements = 12 bytes

Entry Values

Y = Absolute ‘Y:X’ position

U = Message list pointer

DP = $D0

SIZRAS = ‘YX’ size of raster message

Return Values

Same as entry values

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND

DEL13

POSITD, POSITN

MRASTR

RASTER

ZERGND

ASPLAY (SOPLAY)

Description:

Set tune sequence with alternate note set

Tune List Description:

Byte 0 / 1 = Fade list pointer
2 / 3 = Vibrato list pointer
n = Note
$40 = Noise enable
$80 = Next channel enable
n+1 = Tone period ($80 = tune list terminator)

Fade List Description

<no description provided>

Vibrato List Description

<no description provided>

Entry Address = $F690

Maximum Stack Requirements = 4 bytes

Entry Values

X = User note table pointer
U = Tune list pointer
DP = $C8

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed
U = Destroyed

Executive storage modifications

DOREMI, FADE, FADEA, FADEB, FADEC, NEWGEN, REQ0 – REQD, RESTC, TONEB, TONEC, TSTAT, TUNE, VIBE

Executive subroutines utilized

BCLR
CLRBLK
INTREQ
TPLAY
XPLAY

BCLR (CLRSOM)

Description:
Clear ‘B’ bytes starting at value in ‘X’

Entry Address = $F53F

Maximum Stack Requirements = 2 bytes

Entry Values
B = Number of bytes to be cleared
X = buffer pointer

Return Values
A = $FF
B = $FF

Executive subroutines utilized
CLRBLK

BDROT (DISROT)

Description:

Rotate ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F613

Maximum Stack Requirements = 9 bytes

Entry Values

B = Number of vectors – 1
X = ‘Diffy’ list pointer
U = Destination buffer pointer

Return Values

A = $00
B = Destroyed
X = Entry value + 1
U = Entry value + 1

LIST = $00

Executive storage modifications

LAG, LEG
WCSINE, WSINE

Executive subroutines utilized

ADROT, APROT
COSINE
DPRAM
LCSINE, LSINE
MCSINE, MSINE
SINCOS, SINE

BLKFIL (FILL)

Description:

Set a block of memory starting at ‘X’

Entry Address = $F552

Maximum Stack Requirements = 2 bytes

Entry Values

A = Data to be written
B = number of bytes to be written
X = Buffer pointer

Return Values

B = $00

BLKMOV (STFAUX)

Description:

Transfer ‘A’ bytes from source to destination buffer

Entry Address = $F683

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of bytes to be transfered ($00 - $7F)
X = Destination buffer pointer
U = Source buffer pointer

Return Values

A = $FF
B = Contents of last byte transferred

Executive subroutines utilized

BLKMV1

BLKMV1 (BAGAUX)

Description:
Transfer ‘A’ + 1 bytes from source to destination buffer

Entry Address = $F67F

Maximum Stack Requirements = 2 bytes

Entry Values
A = Number of (bytes – 1) to be transferred ($00 - $7F)
X = Destination buffer pointer
U = Source buffer pointer

Return Values
A = $FF
B = Contents of last byte transferred

Executive subroutines utilized
BLKMOV

BXTEST (FINEBOX)

Description:
Symmetric collision test

Entry Address = $F8FF

Maximum Stack Requirements = 10 bytes

Entry Values
A = Box ‘Y’ dimension (delta ‘Y’)
B = Box ‘X’ dimension (delta ‘X’)
X = Y:X coordinates of point to be tested
Y = Y:X coordinates of box center

Return Values
C = 1 – collision detected

BYTADD (SHADD)

Description:

Add contents of ‘A’ to indicated score

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F85E

Maximum Stack Requirements = 4 bytes

Entry Values

A = 2-digit BCD number
X = Score field pointer

LIST = $00

Return Values

A = Destroyed
B = Destroyed
U = 4-digit BCD extension of entry ‘A’

Executive subroutines utilized

SCRADD, STKADD

CLRBLK (GILL)

Description:
Clear a block of memory

Entry Address = $F548

Maximum Stack Requirements = 2 bytes

Entry Values
D = Number of bytes to be cleared
X = Buffer pointer

Return Values
D = $FFFF

CLREX (CLRMEM)

Description:
Clear executive area of memory ($C800 - $C8FF)

Entry Address = $F542

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
D = $FFFF
X = $C800

Executive subroutines utilized
CLRBLK, CLR256

CLR80 (NEGSOM)

Description:

Set a block of memory starting at ‘X’ to value $80

Entry Address = $F550

Maximum Stack Requirements = 2 bytes

Entry Values

B = Number of bytes to be set ($01 - $7F)
X = buffer pointer

Return Values

A = $80
B = $00

Executive subroutines utilized

BLKFIL

CLR256

Description:
Clear 256 bytes starting at ‘X’

Entry Address = $F545

Maximum Stack Requirements = 2 bytes

Entry Values
X = Buffer pointer

Return Values
D = $FFFF

Executive subroutines utilized
CLRBLK

CMPASS (COMPAS)

Description:
Return angle for given delta ‘Y:X’

Entry Address = $F593

Maximum Stack Requirements = 6 bytes

Entry Values
A = Delta ‘Y’
B = Delta ‘X’
DP = $C8

Return Values
A = Angle for given delta ‘Y:X’
B = Angle for given delta ‘Y:X’ (same as exit ‘A’)

ANGLE = Angle for given delta ‘Y:X’ (same as exit ‘A’)

Executive storage modifications
ABSX, ABSY

Executive subroutines utilized
ABSAB, ABSB

CONE

Description:
Select direction within limit cones

Entry Address = $EA3E

Maximum Stack Requirements = 9 bytes

Entry Values
SEED = Random number pointer (Normally ‘RANCID’)

Return Values
B = Random number within limit cones

Executive storage modifications
RANCID

Executive subroutines utilized
RANDOM

COSINE (COSGET)

Description:
Calculate the cosine of ‘A’

Entry Address = $F5D9

Maximum Stack Requirements = 2 bytes

Entry Values
A = Angle to be evaluated

Return Values
A = Cosine of given angle
B = Sign / overflow for resulting cosine
X = $FC6D (#RTRIGS)

Executive subroutines utilized
SINE

CZERO (ZEGO)

Description:

Depending on the setting of ‘ZSKIP’, zero integrators and set the sample / hold for active ground.

Entry Address = $F34F

Maximum Stack Requirements = 2 bytes

Entry Values

DP = $D0

ZSKIP = $00 - Skip integrator zeroing
!= 0 - Zero integrators

Return Values

A = $03 ($00 if ZSKIP = $00)
B = $01 (Entry value if ZSKIP = $00)

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT

Executive subroutines utilized

ACTGND
ZERGND

D2TMR (DEKR3)

Description:
Decrement 3 interval timers (XTMR0 – XTMR2)

Entry Address = $F55A

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF
X = $C82E (#XTMR0)

Executive storage modifications
XTMR0 – XTMR2

Executive subroutines utilized
DECTMR

Comments
If used, generally called once per frame

DASHDF (DASHY)

Description:

Draw a dashed version of the given ‘DIFFY’ list

List Description:

byte 0 / 1 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F437

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DIFFY’ list pointer
DP = $D0

DASH = Dash pattern
LIST = Number of vectors – 1
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
ZERGND

Comments

Execution of ‘CZERO’ is inconsistent!!!

DASHPK (DASHY3)

Description:

Draw dashed lines according to ‘Packet’ format

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $00 – draw blank line
$01 – packet terminator
$02 – draw solid line
$FF – draw dashed line (uses ‘DASH’)

Entry Address = $F46E

Maximum Stack Requirements = 5 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

DASH = Dash pattern
LIST = $00
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 1

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DASHDF, DIFFAB, DIFFY, DUFFAB, DUFFY
ZERGND

DBNCE (ENPUT)

Description:
Read controller switches and debounce switch status.

Entry Address = $F1B4

Maximum Stack Requirements = 3 bytes

Entry Values
A = Direct response switch mask
DP = $D0

Return Values
A = Contents of ‘EDGE’
B = $00
X = $C81A (#KEY7 + 1)

Executive storage modifications
EDGE, KEY0 – KEY7, TRIGGR, TRIGGR + 1

Control register modifications
CNTRL, DAC, DDAC

Executive subroutines utilized
INPUT

DDOT

Description:
Position with 16-bit ‘Y:X’ values and draw dot

Entry Address = $EA6D

Maximum Stack Requirements = 10 bytes

Entry Values
Y = Pointer to 32-bit absolute ‘Y:X’ position
DP = $D0

DWELL = Dot ‘ON’ time

Return Values
Same as entry values

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized
ABSAB, ABSB, ACTGNC
DOT, DOTAB
POSITN
ZERGND

DECBIT (BITE)

Description:
Decode bit position

Entry Address = $F57E

Maximum Stack Requirements = 2 bytes

Entry Values
A = Bit number ($00 - $07)

Return Values
A = Result as below

'A'	Value Returned
$00	$01
$01	$02
$02	$04
$03	$08
$04	$10
$05	$20
$06	$40
$07	$80

X = $F9DC (#DECTBL)

DECTMR (DEKR)

Description:
Decrement interval timers (XTMR0 – XTMR5)

Entry Address = $F55E

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF
X = $C82E (#XTMR0)

Executive storage modifications
XTMR0 – XTMR5

Comments
If used, generally called once per frame

DEFLOK

Description:

Over-come screen collapse circuitry

Entry Address = $F2E6

Maximum Stack Requirements = 6 bytes

Entry Values

DP = $D0

Return Values

A = $03
B = $01
X = $F9F4 (#KEPALV + 4)

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
POSIT2, POSITB, POSITN, POSITX
ZERGND, ZERO

Comments

'DEFLOK' is performed by calling 'FRWAIT'. However, it has been necessary with some games to add additional 'DEFLOK's to prevent long-term screen collapse.

DEL

Description:
Delay execution for a minimum of 20 cycles (x.xxx us)

Entry Address = $F57A

Maximum Stack Requirements = 2 bytes

Entry Values
B = Delay period

Return Values
B = $FF

Executive subroutines utilized
DEL13

DEL13

Description:
Delay execution for 13 cycles (x.xxx us)

Entry Address = $F57D

Maximum Stack Requirements = 2 bytes

Entry Values
None required

DEL20

Description:
Delay execution for 20 cycles (x.xxx us)

Entry Address = $F579

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF

Executive subroutines utilized
DEL, DEL13

DEL28

Description:
Delay execution for 28 cycles (x.xxx us)

Entry Address = $F575

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF

Executive subroutines utilized
DEL, DEL13

DEL33

Description:
Delay execution for 33 cycles (x.xxx us)

Entry Address = $F571

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF

Executive subroutines utilized
DEL, DEL13

DEL38

Description:
Delay execution for 38 cycles (x.xxx us)

Entry Address = $F56D

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
B = $FF

Executive subroutines utilized
DEL, DEL13

DIFDOT

Description:

Draw dots according to ‘Diffy’ format

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F2D5

Maximum Stack Requirements = 8 bytes

Entry Values

X = ‘DIFFY’ list pointer
DP = $D0

DWELL = Dot ‘ON’ time
LIST = Number of vectors – 1
T1LOLC = Vector length (scale factor)

Return Values

A = $03
B = $01
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DOT, DOTAB, DOTX
POSITN
ZERGND

DIFFAB

Description:

Draw a single vector from the current beam position using the relative vector values given in ‘D’.

Entry Address = $F3DF

Maximum Stack Requirements = 2 bytes

Entry Values

A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
DP = $D0

LIST = $00
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
ZERGND

DIFFAX

Description:

Draw from ‘Diffy’ style list

List Description:

byte 0 = Number of vectors - 1
1 / 2 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F3CE

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DIFFY’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY
LDIFFY
ZERGND

DIFFY

Description:

Draw from ‘Diffy’ style list

List Description:

byte 1 / 2 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F3DD

Maximum Stack Requirements = 2 bytes

Entry Values

X = “DIFFY’ list pointer

DP = $D0

LIST = Number of vectors – 1

T1LOLC = Vector length (scale factor)

ZSKIP = $00 – Skip integrator zeroing

!= $00 – Zero integrators

Return Values

A = Destroyed

B = Destroyed

X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND

CZERO

DIFFAB, LDIFFY

ZERGND

DIFLST (DIFFX)

Description:

Draw from ‘Diffy’ style list

List Description:

byte 0 = Number of vectors – 1
1 = Vector length (scale factor)
2 / 3 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F3D6

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DIFFY’ list pointer
DP = $DO

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A – Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY
LDIFFY
TDIFFY
ZERGND

DIFTIM

Description:

Draw from ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F3D2

Maximum Stack Requirements = 2 bytes

Entry Values

B = Vector length (scale factor)
X = ‘DIFFY’ list pointer
DP = $D0

LIST = Number of vectors – 1
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY
LDIFFY
ZERGND

DOT

Description:
Draw dot

Entry Address = $F2C5

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

DWELL = Dot ‘ON’ time

Return Values
A = $FF
B = $00

Control register modifications
SHIFT

DOTAB

Description:
Position relative and draw dot

Entry Address = $F2C3

Maximum Stack Requirements = 6 bytes

Entry Values
A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
DP = $D0

DWELL = Dot ‘ON’ time
T1LOLC = Vector length (scale factor)

Return Values
A = $FF
B = $00

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized
ABSAB, ABSB
DOT
POSITN

DOTPCK (DOTPAK)

Description:

Draw dots according to ‘Packet’ format

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$80 - $FF – position for dot

Entry Address = $F2DE

Maximum Stack Requirements = 8 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

DWELL = Dot ‘ON’ time
T1LOLC = Vector length (scale factor)

Return Values

A = $03
B = $01
X = End of list + 1

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DOT, DOTAB, DOTX
POSITN
ZERGND

DOTTIM

Description:

Draw one dot from ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F2BE

Maximum Stack Requirements = 6 bytes

Entry Values

B = Dot ‘ON’ time
X = ‘DIFFY’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)

Return Values

A = $FF
B = $00
X = Entry value + 2

Executive storage modifications

DWELL

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ABSAB, ABSB
DOT, DOTAB, DOTX
POSITN

DOTX

Description:

Draw one dot from ‘Diffy’ style list

List Description:

byte 0 / 1 = Positioning vector (Y:X)

Entry Address = $F2C1

Maximum Stack Requirements = 6 bytes

Entry Values

X = “DIFFY’ list pointer
DP = $D0

DWELL = Dot ‘ON’ time
T1LOLC = Vector length (scale factor)

Return Values

A = $FF
B = $00
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ABSAB, ABSB
DOTAB, DOT
POSITN

DPACK

Description:

Position with 16-bit ‘X:Y’ values and draw packet

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $EA8D

Maximum Stack Requirements = 10 bytes

Entry Values

B = Zoom value (scale factor)
X = ‘Packet’ list pointer
Y = Pointer to 32-bit absolute ‘Y:X’ position
DP = $D0

Return Values

Same as entry values

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
CZERO
PACKET, POSITN, POSWID
TPACK
ZERGND

DPIO

Description:
Set 6809 ‘DP’ register for I/O accesses ($D0)

Entry Address = $F1AA

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
A = $D0
DP = $D0

DPRAM

Description:
Set 6809 ‘DP’ register for RAM accesses ($C8)

Entry Address = $F1AF

Maximum Stack Requirements = 2 bytes

Entry Values
None required

Return Values
A = $C8
DP = $C8

DROT (DANROT)

Description:

Rotate ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F610

Maximum Stack Requirements = 9 bytes

Entry Values

A = Rotation angle
B = Number of vectors – 1
X = ‘DIFFY’ list pointer
U = Destination buffer pointer

Return Values

A = $00
B = Destroyed
X = Entry value + 1
U = Entry value + 1

LIST = $00

Executive storage modifications

ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized

ADROT, APROT
BDROT
COSINE
DPRAM
LCSINE, LSINE
MCSINE, MSINE
SINCOS, SINE

DSHDF (DASHEL)

Description:

Draw dashed lines according to ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F434

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors – 1
X = “DIFFY’ list pointer
DP = $D0

DASH = Dash pattern
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DASHDF, DSHDF1
ZERGND

Comments

Execution of ‘CZERO’ is inconsistent !!!

DSHDF1 (DASHE)

Description:

Draw dashed lines according to ‘Diffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F433

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors
X = ‘DIFFY’ list pointer
DP = $D0

DASH = Dash pattern
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DASHDF, DSHDF
ZERGND

Comments

Execution of ‘CZERO’ is inconsistent !!!

DSHIP (SHIPSHO)

Description:
Display markers (counters remaining)

Entry Address = $F393

Maximum Stack Requirements = 17 bytes

Entry Values
A = ASCII code of symbol
B = Number of markers remaining
X = Position of marker on screen
DP = $D0

SIZRAS = 'YX' size of raster message

Return Values
A = $03
B = $01
X = $FBB4
U = Destroyed

Executive storage modifications
MESAGE

Control register modifications
ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized
ABSAB, ABSB, ACTGND
DEL13, DEL28
POSITD, POSITN
MRASTR, MSSPOS
RASTER, RSTPOS
ZERGND

DUFFAB

Description:

Move a single vector from the current beam position using the relative vector values given in ‘D’

Entry Address = $F3BE

Maximum Stack Requirements = 2 bytes

Entry Values

A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
DP = $D0

LIST = $00
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB
ZERGND

DUFFAX

Description:

Draw from ‘Duffy’ style list

List Description:

byte 0 = Number of vectors - 1
1 / 2 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F3AD

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DUFFY’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB, DUFFY
LDIFFY, LDUFFY
ZERGND

DUFFY

Description:

Draw from ‘Duffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F3BC

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DUFFY’ list pointer
DP = $D0

LIST = Number of vectors – 1
T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB
LDIFFY
ZERGND

DUFLST (DUFFX)

Description:

Draw from ‘Duffy’ style list

List Description:

byte 0 = Number of vectors – 1
1 = Vector length (scale factor)
2 / 3 = Vector #1 (Y:X)
- -
n / n+1 = Vector #n (Y:X)

Entry Address = $F3B5

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘DUFFY’ list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB, DUFFY
LDIFFY, LDUFFY
TDUFFY
ZERGND

DUFTIM

Description:

Draw from ‘Duffy’ style list

List Description:

byte 0 / 1 = Vector #1 (Y:X)

\- -

n / n+1 = Vector #n (Y:X)

Entry Address = $F3B1

Maximum Stack Requirements = 2 bytes

Entry Values

B = Vector length (scale factor)
X = ‘DUFFY’ list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB, DUFFY
LDIFFY
ZERGND

DZERO (ZERO.DP)

Description:

Set the 6809 ‘DP’ register to the I/O page ($D0), zero the integrators and set active ground

Entry Address = $F34A

Maximum Stack Requirements = 4 bytes

Entry Values

None required

Return Values

A = $03
B = $01
DP = $D0

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT

Executive subroutines utilized

ACTGND
DPIO
ZERGND

EXPLOD (AXE)

Description:

Complex explosion sound-effect handler

Explosion Parameter Table Description

Byte 0 = Tone and noise channel enables
Bit 0 = Tone channel #
1 = Tone channel #
2 = Tone channel #
3 = Noise source #
4 = Noise source #
5 = Noise source #

Byte 1 = Noise source sweep
= 0 – Sweep frequency UP
> 0 – Sweep frequencey DOWN
< 0 – Inhibit frequency sweep

Byte 2 = Volume sweep
= 0 – Sweep volume UP
> 0 – Sweep volume DOWN
< 0 – Inhibit volume sweep

Byte 3 = Explosion duration
$01 – Longest explosion duration
$80 – Shortest explosion duration

Entry Address = $F92E

Maximum Stack Requirements = 4 bytes

Entry Values
U = Explosion parameter table pointer
DP = $C8

Return Values
A = Destroyed
B = Destroyed
X = Destroyed

XACON = $00 (when explosion is completed)

Executive storage modifications
RATEA, RATEB, RATEC, REQ4 – REQ7, SATUS, TUNE

Executive subroutines utilized
BLKMOV
DECBIT
RANDOM
SETAMP

FRWAIT (FRAM20)

Description:

Wait for beginning of frame boundary (Timer #2 = $0000). Since the program may exceed frame time, this routine will assure a given maximum frame rate.

Entry Address = $F192

Maximum Stack Requirements = 8 bytes

Entry Values

No register parameters required

FRMTIM = Frame to frame interval

Return Values

A = $03
B = $01
X = $F9F4 (#KEPALV + 4)
DP = $D0

Executive storage modifications

FRAME

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC, T2LOLC, T2HOC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEFLOK, DPIO
POSIT2, POSITB, POSITN, POSITX
ZERGND, ZERO

Comments

Performs one ‘DEFLOK’

HISCR (HIGHSCR)

Description:

Calculate high score and save for opening logo

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F8D8

Maximum Stack Requirements = 8 bytes

Entry Values

X = Score field pointer
U = High score field pointer

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
U = Destroyed

Executive subroutines utilized

WINNER

INPUT

Description:
Read the status of controll buttons.

Entry Address = $F1BA

Maximum Stack Requirements = 3 bytes

Entry Values
DP = $D0

Return Values
A = Contents of ‘EDGE’
B = $00
X = $C81A (#KEY7 + 1)

Executive storage modifications
EDGE, KEY0 – KEY7, TRIGGR, TRIGGR + 1

Control register modifications
CNTRL, DAC, DDAC

INT1Q

Description:
Set intensity at _ level

Entry Address = $F29D

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $05
B = $01

Executive storage modifications
TENSITY

Control register modifications
CNTRL, DAC

Executive subroutines utilized
INTENS

Comments
Sets intensity to $1F

INT2Q (INTMID)

Description:
Set intensity at 1/2 level

Entry Address = $F2A1

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $05
B = $01

Executive storage modifications
TENSITY

Control register modifications
CNTRL, DAC

Executive subroutines utilized
INTENS

Comments
Sets intensity to $3F

INT3Q

Description:
Set intensity at 3/4 level

Entry Address = $F2A5

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $05
B = $01

Executive storage modifications
TENSITY

Control register modifications
CNTRL, DAC

Executive subroutines utilized
INTENS

Comments
Sets intensity to $5F

INTALL (INITALL)

Description:
Initialize the Vectrex hardware and executive parameters.

Entry Address = $F18B

Maximum Stack Requirements = 8 bytes

Entry Values
None required

Return Values
A = $3F
B = $FF
X = $C83F (#REQ0)
DP = $D0

Executive storage modifications
RAM between REG0 [$C800] and OPTION [$C87A] (inclusive) are cleared ($00)

DWELL = $05 (Dot 'ON' time)
EPOT0 = $01 (Enable – Controller #1: Right / Left)
EPOT1 = $03 (Enable – Controller #1: Up / Down)
EPOT2 = $05 (Enable – Controller #2: Right / Left)
EPOT3 = $07 (Enable – Controller #2: Up / Down)
FRMTIM = $3075 (#MSEC20 – 50 Hertz frame rate)
RANCID = Non zero
SEED = $C87D (#RANCID)

Control register modifications
ACNTRL, CNTRL, DAC, DCNTRL, PCNTRL, SHIFT, T1HOC, T1LOLC, T2LOLC, PSG0 - PSGE

Executive subroutines utilized
ABSAB, ABSB, ACTGND
BCLR
CLRBLK
DEFLOK, DPIO, DPRAM
FRWAIT
INTMSC, INTPIA, INTPSG, INTREQ
POSIT2, POSITB, POSITN, POSITX
WRPSG, WRREG
ZERGND, ZERO

Comments
Zeroes the integrators and sets active ground on return to user.

INTENS

Description:
Set intensity at user value

Entry Address = $F2AB

Maximum Stack Requirements = 2 bytes

Entry Values
A = Intensity level ($00 - $7F)
DP = $D0

Return Values
A = $05
B = $01

Executive storage modifications
TENSITY

Control register modifications
CNTRL, DAC

Comments
The value given for the intensity setting must not be negative ($80 - $FF). Setting the intensity to a negative value may result in damage to the Vectrex.

INTMAX

Description:
Set intensity at maximum level

Entry Address = $F2A9

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $05
B = $01

Executive storage modifications
TENSITY

Control register modifications
CNTRL, DAC

Executive subroutines utilized
INTENS

Comments
Sets intensity to $7F

INTMSC (INITMSC)

Description:
Initialize misc. executive parameters

Entry Address = $F164

Maximum Stack Requirements = 4 bytes

Entry Values
None required

Return Values
A = $05
B = $07
X = $C800 (#REG0)
DP = $C8

Executive storage modifications
RAM between REG0 [$C800] and OPTION [$C87A] (inclusive) are cleared ($00)

DWELL = $05 (Dot 'ON' time)
EPOT0 = $01 (Enable – Controller #1: Right / Left)
EPOT1 = $03 (Enable – Controller #1: Up / Down)
EPOT2 = $05 (Enable – Controller #2: Right / Left)
EPOT3 = $07 (Enable – Controller #2: Up / Down)
FRMTIM = $3075 (#MSEC20 – 50 Hertz frame rate)
RANCID = Non zero
SEED = $C87D (#RANCID)

Executive subroutines utilized
BCLR
CLRBLK
DPRAM

INTPIA (INITPIA)

Description:

Initialize the programmable interface adapter (PIA).

Entry Address = $F14C

Maximum Stack Requirements = 6 bytes

Entry Values

No register parameters required

FRMTIM = Frame to frame interval

Return Values

A = $03
B = $01
X = $F9F4 (#KEPALV + 4)
DP = $D0

Control register modifications

ACNTRL, CNTRL, DAC, DCNTRL, PCNTRL, SHIFT, T1HOC, T1LOLC, T2LOLC, T2HOC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEFLOK, DPIO
FRWAIT
POSIT2, POSITB, POSITN, POSITX
ZERGND, ZERO

Comments

Zeroes the integrators and sets active ground on return to user.

INTPSG (INITPSG)

Description:
Initialize programmable sound generator (PSG).

Entry Address = $F272

Maximum Stack Requirements = 4 bytes

Entry Values
DP = $D0

Return Values
A = $3F
B = $FF
X = $C83F (#REQ0)

Executive storage modifications
REG0 – REGE, REQ0 - REQD

Control register modifications
CNTRL, DAC, PSG0 - PSGE

Executive subroutines utilized
BCLR
CLRBLK
INTREQ
WRPSG, WRREG

INTREQ (IREQ)

Description:
Initialize the ‘REQx’ area (sound mirror).

Entry Address = $F533

Maximum Stack Requirements = 4 bytes

Entry Values
None required

Return Values
A = $3F
B = $FF
X = $C83F (#REQ0)

Executive storage modifications
REQ0 – REQ5, REQ7 – REQD = $00
REQ6 = $3F

Executive subroutines utilized
BCLR
CLRBLK

JOYBIT (PBANG4)

Description:

Read the UP / DOWN, RIGHT / LEFT status of the controller joysticks

Entry Address = $F1F8

Maximum Stack Requirements = 2 bytes

Entry Values

DP = $D0

EPOT0 = $01 ($00 to disable) – Controller #1: Right / Left
EPOT1 = $03 ($00 to disable) – Controller #1: Up / Down
EPOT2 = $05 ($00 to disable) – Controller #2: Right / Left
EPOT3 = $07 ($00 to disable) – Controller #2: Up / Down
LIST = $00 - $7F
POTRES = Joystick resolution limit

Return Values

A = $01
B = Contents of 'POT3'
X = $C823 (#LIST)

LIST = $00

POT0 = Controller #1: Right / Left
POT1 = Controller #1: Up / Down
POT2 = Controller #2: Right / Left
POT3 = Controller #2: Up / Down

where:

< 0 – joystick is left or down
= 0 – joystick is centered
> 0 – joystick is right or up

Control register modifications

CNTRL, DAC

JOYSTK (POTS4)

Description:

Read the absolute position of the controller joysticks.

Entry Address = $F1F5

Maximum Stack Requirements = 2 bytes

Entry Values

DP = $D0

EPOT0 = $01 ($00 to disable) – Controller #1: Right / Left
EPOT1 = $03 ($00 to disable) – Controller #1: Up / Down
EPOT2 = $05 ($00 to disable) – Controller #2: Right / Left
EPOT3 = $07 ($00 to disable) – Controller #2: Up / Down
LIST = $00
POTRES = Joystick resolution limit, where
$00 – 8 bits (default)
$01 – 7 bits
$02 – 6 bits
$04 – 5 bits
$08 – 4 bits
$10 – 3 bits
$20 – 2 bits
$40 – 1 bit

Return Values

A = $01
B = Contents of ‘POT3’
X = $C823 (#LIST)

LIST = $00
POT0 = Controller #1: Right / Left
POT1 = Controller #1: Up / Down
POT2 = Controller #2: Right / Left
POT3 = Controller #2: Up / Down

Control register modifications

CNTRL, DAC

Executive subroutines utilized

JOYBIT

LCSINE (RCOS)

Description:
Multiply ‘LEG’ by previous cosine value

Entry Address = $F663

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $C8

LEG = Multiplier
WCSINE = Previous cosine result

Return Values
A = Product of LEG * WCSINE
B = Contents of WCSINE + 1

Executive storage modifications
LAG

LDIFFY (DIFLST)

Description:

Draw according ‘Diffy’ style list

List Description:

Byte 0 / 1 = Vector #1 (Y:X)

- .

Byte n / n+ 1 = Vector #n (Y:X)

Entry Address = $F3DA

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors – 1
X = ‘DIFFY’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY
ZERGND

LDUFFY (DUFLST)

Description:

Draw according to ‘Duffy’ style list

List Description:

Byte 0 / 1 = Vector #1 (Y:X)

- .

Byte n / n+ 1 = Vector #n (Y:X)

Entry Address = $F3B9

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors – 1
X = ‘DUFFY’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB, DUFFY
LDIFFY
ZERGND

LNROT (ROTOR)

Description:
Rotate a single line

Entry Address = $F601

Maximum Stack Requirements = 8 bytes

Entry Values
A = Initial ‘Y’ value
B = Angle of rotation
DP = $C8

Return Values
A = Rotated ‘Y’ vector value
B = Rotated ‘X’ vector value

Executive storage modifications
ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized
ALNROT
COSINE
LCSINE, LSINE
SINCOS, SINE

LPACK (PACXX)

Description:

Draw according to ‘Packet’ format

List Description:

Byte 0 = Vector length (scale factor)
1 / 2 / 3 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $F40C

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOD, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
PACKET
TPACK
ZERGND

LROT90 (RATOR)

Description:
Rotate a single line

Entry Address = $F5FF

Maximum Stack Requirements = 8 bytes

Entry Values
A = Initial ‘Y’ value
B = Angle of rotation
DP = $C8

Return Values
A = Rotated ‘Y’ vector value
B = Rotated ‘X’ vector value

Executive storage modifications
ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized
ALNROT
COSINE
LCSINE, LSINE, LNROT
SINCOS, SINE

LSINE (RSIN)

Description:
Multiply ‘LEG’ by previous sine value

Entry Address = $F65D

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $C8

LEG = Multiplier
WSINE = Previous Sine result

Return Values
A = Product of LEG * WSINE
B = Contents of WSINE + 1

Executive storage modifications
LAG

Executive subroutines utilized
LCSINE

MCSINE (RCOSA)

Description:
Multiply ‘A’ by previous cosine value

Entry Address = $F661

Maximum Stack Requirements = 2 bytes

Entry Values
A = Multiplier
DP = $C8

WCSINE = Previous cosine result

Return Values
A = Product of ‘A’ * WCSINE
B = Contents of WCSINE + 1

LEG = Entry ‘A’ value

Executive storage modifications
LAG

Executive subroutines utilized
LCSINE

MLTY8

Description:
Form ‘Y:X’ displacements (x8)

Entry Address = $E7B5

Maximum Stack Requirements = 16 bytes

Entry Values
A = speed vector
B = Direction (Angle of rotation)
DP = $C8

Return Values
X = ‘X’ Displacement value (x8)
Y = ‘Y’ Displacement value (x8)

Executive storage modifications
ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized
ALNROT
COSINE
LCSINE, LSINE, LNROT
SINCOS, SINE

MLTY16

Description:
Form ‘Y:X’ displacements (x16)

Entry Address = $E7D2

Maximum Stack Requirements = 18 bytes

Entry Values
A = Speed vector
B = Direction (Angle of rotation)
DP = $C8

Return Values
X = ‘X’ Displacement value (x16)
Y = ‘Y’ Displacement value (x16)

Executive storage modifications
ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized
ALNROT
COSINE
LCSINE, LSINE, LNROT
MLTY8
SINCOS, SINE

MRASTR (RASTER)

Description:

Display raster string indicated by ‘MESAGE’

Message List Description:

byte 0 – n = Raster message string ($20 - $6F)
n+1 = Raster terminator ($80)

Entry Address = $F498

Maximum Stack Requirements = 2 bytes

Entry Values

DP = $D0

MESAGE = Raster message string pointer
SIZRAS = ‘YX’ size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = End of message string + 1

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT

Executive subroutines utilized

ACTGND
DEL13
ZERGND

MSINE (RSINA)

Description:
Multiply ‘A’ by previous sine value

Entry Address = $F65B

Maximum Stack Requirements = 2 bytes

Entry Values
A = Multiplier
DP = $C8

WSINE = Previous sine result

Return Values
A = Product of ‘A’ * WSINE
B = Contents of WSINE + 1

LEG = Entry ‘A’ value

Executive storage modifications
LAG

Executive subroutines utilized
LCSINE, LSINE

MSSPOS (POSDRAS)

Description:

Position and display raster message

Message List Description:

byte 0 – n = Raster message string ($20 - $6F)
n+1 = Raster terminator ($80)

Entry Address = $F37A

Maximum Stack Requirements = 6 bytes

Entry Values

A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
U = Message string pointer
DP = $D0

SIZRAS = ‘YX’ size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEL13, DEL28
POSITD, POSITN
MRASTR
RASTER
ZERGND

OFF1BX (OFF1BOX)

Description:
Off-center symmetric collision test

Entry Address = $F8E5

Maximum Stack Requirements = 10 bytes

Entry Values
A = Box ‘Y’ dimension (Delta ‘Y’)
B = Box ‘X’ dimension (Delta ‘X’)
X = ‘Y:X’ coordinates of point to be tested
Y = ‘Y:X’ coordinates of center of box
U = Off-set value pointer

Return Values
C = 1 – Collision detected

Executive subroutines utilized
BXTEST

OFF2BX (OFF2BOX)

Description:
Off-center symmetric collision text

Entry Address = $F8F3

Maximum Stack Requirements = 10 bytes

Entry Values
A = Box ‘Y’ dimension (Delta ‘Y’)
B = Box ‘X’ dimension (Delta ‘X’)
X = ‘Y:X’ coordinates of point to be tested
Y = ‘Y:X’ coordinates of center of box
U = Off-set value (‘Y:X’)

Return Values
C = 1 – Collision detected

Executive subroutines utilized
BXTEST
OFF1BX

PACK1X (PAC1X)

Description:

Draw according to ‘Packet’ style list

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $F408

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND, CZERO, PACKET, TPACK, ZERGND

Comments

Uses 1x scale factor ($7F)

PACK2X (PAC2X)

Description:

Draw according to ‘Packet’ style list

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $F404

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
PACKET
TPACK
ZERGND

Comments

Uses 2x scale factor ($FF)

PACKET

Description:

Draw according to ‘Packet’ style list

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $F410

Maximum Stack Requirements = 2 bytes

Entry Values

X = ‘PACKET’ list pointer
DP = $D0

T1LOLC = Vector length (scale factor)
ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ACTGND
CZERO
ZERGND

POSIT1

Description:

Release integrators and position beam

List Description:

byte 0 / 1 = Positioning vector (Y:X)

Entry Address = $F30C

Maximum Stack Requirements = 4 bytes

Entry Values

X = List pointer
DP = $D0

Return Values

A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ASAB, ABSB
POSITB, POSITN, POSITX

Comments

Uses 1x scale factor ($7F)

POSIT2

Description:
Release integrators and position beam

List Description:
byte 0 / 1 = Positioning vector (Y:X)

Entry Address = $F308

Maximum Stack Requirements = 4 bytes

Entry Values
X = List pointer
DP = $D0

Return Values
A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized
ASAB, ABSB
POSITB, POSITN, POSITX

Comments
Uses 2x scale factor ($FF)

POSITB

Description:

Release integrators and position beam

List Description:

byte 0 / 1 = Positioning vector (Y:X)

Entry Address = $F30E

Maximum Stack Requirements = 4 bytes

Entry Values

B = Vector length (scale factor)
X = List pointer
DP = $D0

Return Values

A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ASAB, ABSB
POSITB, POSITN, POSITX

Comments

Uses scale factor specified in ‘B’ register

POSITD

Description:
Release integrators and position beam

Entry Address = $F2FC

Maximum Stack Requirements = 4 bytes

Entry Values
A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
DP = $D0

Return Values
A = Destroyed
B = Destroyed

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized
ASAB, ABSB
POSITN

POSITN

Description:
Release integrators and position beam

Entry Address = $F312

Maximum Stack Requirements = 4 bytes

Entry Values
A = Relative ‘Y’ vector value
B = Relative ‘X’ vector value
DP = $D0

T1LOLC = Vector length (scale factor)

Return Values
A = Destroyed
B = Destroyed

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized
ASAB, ABSB

Comments
Uses 1x scale factor ($7F)

POSITX

Description:

Release integrators and position beam

List Description:

byte 0 / 1 = Positioning vector (Y:X)

Entry Address = $F310

Maximum Stack Requirements = 4 bytes

Entry Values

X = List pointer
DP = $D0

T1LOLC = Vector length (scale factor)

Return Values

A = Destroyed
B = Destroyed
X = Entry value + 2

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC

Executive subroutines utilized

ASAB, ABSB
POSITN

POSWID

Description:

Release integrators and position beam using 16-bit ‘Y:X’ values

List Description:

byte 0 / 1 = ‘Y’ Position vector (16-bits)
byte 2 / 3 = ‘X’ Positioning vector (16-bits)

Entry Address = $F2F2

Maximum Stack Requirements = 4 bytes

Entry Values

X = List pointer
DP = $D0

Return Values

A = Destroyed
B = Destroyed

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB
POSITN

Comments

Uses 1x scale factor ($7F)

PROT (POTATA)

Description:

Rotate ‘Packet’ style list

List Description:

```
Byte 0  /   1   /  2    = Vector #1 (C:Y:X)
     -    -   -             -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3           = $01 (packet terminator)
```

Entry Address = $F61F

Maximum Stack Requirements = 9 bytes

Entry Values

A = Rotation angle
X = ‘Packet’ list pointer
U = Destination buffer pointer

Return Values

A = ‘Packet’ terminator value
B = Destroyed
X = End of ‘Packet’ list + 1
U = End of destination buffer + 1

LIST = $00

Executive storage modifications

ANGLE, LAG, LEG, WCSINE, WSINE

Executive subroutines utilized

APROT
COSINE
DPRAM
LCSINE, LSINE
MCSINE, MSINE
SINCOS, SINE

PSGLST (PSGLUP)

Description:
Send sound string to PSG and mirror

Entry Address = $F27D

Maximum Stack Requirements = 4 bytes

Entry Values
U = Pointer to sound string
DP = $D0

Return Values
D = Sound string terminator
X = $C800 (#REG0)
U = Points to end of sound string

Executive storage modifications
REGx

Control register modifications
CNTRL, DAC, PSGx

Executive subroutines utilized
PSGMIR
WRPSG

PSGMIR (PSGULP)

Description:
Send sound string to PSG and indicated mirror

Entry Address = $F284

Maximum Stack Requirements = 4 bytes

Entry Values
X = Pointer to PSG mirror
U = Pointer to sound string
DP = $D0

Return Values
D = Sound string terminator
U = Points to end of sound string

Control register modifications
CNTRL, DAC, PSGx

Executive subroutines utilized
WRPSG

RAND3

Description:
Generate random number

Entry Address = $F511

Maximum Stack Requirements = 5 bytes

Entry Values
No register parameters required

SEED = Random number pointer (Normally ‘RANCID’)

Return Values
A = Random number

Executive storage modifications
RANCID

Executive subroutines utilized
RANDOM

RANDOM

Description:
Generate random number

Entry Address = $F517

Maximum Stack Requirements = 5 bytes

Entry Values
No register parameters required

SEED = Random number pointer (Normally ‘RANCID’)

Return Values
A = Random number

Executive storage modifications
RANCID

RANPOS

Description:
Determine random ‘Y:X’ position

Entry Address = $E98A

Maximum Stack Requirements = 9 bytes

Entry Values
No register parameters required

SEED = Random number pointer (Normally ‘RANCID’)

Return Values
A = ‘Y’ axis value ($00 - $FF)
B = ‘X’ axis value ($60 - $7F, $A0 - $FF)

Executive storage modifications
RANCID

Executive subroutines utilized
RANDOM

RASTER (RASTUR)

Description:

Display raster string as indicated by ‘U’

Message List Description:

byte 0 – n = Raster message string ($20 - $6F)
n+1 = Raster terminator ($80)

Entry Address = $F495

Maximum Stack Requirements = 2 bytes

Entry Values

U = Message string pointer
DP = $DP

SIZRAS = ‘YX’ size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT

Executive subroutines utilized

ACTGND
DEL13
MRASTR
ZERGND

REPLAY

Description:

Set tune sequence

Tune List Description:

byte 0 / 1 = Fade list pointer
2 / 3 = Vibrato list pointer
n = Note
$40 = Noise enable
$80 = Next channel enable
n+1 = Tone period ($80 = tune list terminator)

Fade List Description

<no description provided>

Vibrato List Description

<no description provided>

Entry Address = $F687

Maximum Stack Requirements = 4 bytes

Entry Values

U = Tune list pointer
DP = $C8

TSTAT = .

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed

Executive storage modifications

DOREMI, FADE, FADEA, FADEB, FADEC, NEWGET, REQ0 – REQD, RESTC, TONEA, TONEB, TONEC, TSTAT, TUNE, VIBE

Executive subroutines utilized

ASPLAY
BCLR
CLRBLK
INTREQ
SPLAY
TPLAY
XPLAY

REQOUT

Description:
Send ‘REQx’ to PSG and mirror

Entry Address = $F289

Maximum Stack Requirements = 4 bytes

Entry Values
DP = $D0

REQ0 – REQD - .

Return Values
A = $FF
B = Contents of ‘REQD’
X = $C80D (#REGD)
U = $C84C (#REQD + 1)

Executive storage modifications
REG0 – REGD, REQ0 - REQD

Control register modifications
CNTRL, DAC, PSGx

Executive subroutines utilized
WRPSG

RSTPOS (POSNRAS)

Description:

Fetch position and display raster message

Message List Description:

byte 0 / 1 = Absolute screen position (Y:X)
2 – n = Raster message string ($20 - $6F)
n+1 = Raster terminator ($80)

Entry Address = $F378

Maximum Stack Requirements = 6 bytes

Entry Values

U = Message string pointer
DP = $D0

SIZRAS = 'YZ' size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEL13, DEL28
MRASTR, MSSPOS
POSITD, POSITN
RASTER
ZERGND

RSTSIZ (SIZPRAS)

Description:

Fetch size, position and display raster message

Message List Description:

byte 0 / 1 = Raster message size (SIZRAS)
2 / 3 = Absolute screen position (Y:X)
4 – n = Raster message string ($20 - $6F)
n+1 = Raster terminator ($80)

Entry Address = $F373

Maximum Stack Requirements = 6 bytes

Entry Values

U = Message string pointer
DP = $D0

SIZRAS = 'YZ' size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE, SIZRAS

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEL13, DEL28
MRASTR, MSSPOS
POSITD, POSITN
RASTER, RSTPOS
ZERGND

SCLR

Description:

Clear indicated score field

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F84F

Maximum Stack Requirements = 2 bytes

Entry Values

X = Score field pointer

Return Values

A = $30
B = $80

SCRADD (SADD)

Description:

Add indicated BCD value to score field

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F87C

Maximum Stack Requirements = 4 bytes

Entry Values

D = 4-digit BCD number
X = Score field pointer

LIST = $00

Return Values

A = Destroyed
B = Destroyed

Executive subroutines utilized

STKADD

SCRBTH

Description:

Draw scores for both players

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $EACF

Maximum Stack Requirements = 14 bytes

Entry Values

DP = $D0

PLAYRS = Number of players selected – 1
SCOR1 = Raster score for player #1
SCOR2 = Raster score for player #2

Return Values

A = Destroyed
B = Destroyed
Y = Destroyed
U = Destroyed

Executive storage modifications

MESAGE, SIZRAS, TENSTY

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND, ASMESS
DEL13
INTENS, INTMAX
MRASTR
POSITD, POSITN
RASTER
ZERGND

SCRMES

Description:

Draw score for currently active player

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $EAB4

Maximum Stack Requirements = 14 bytes

Entry Values

DP = $D0

ACTPLY = Currently active player ($00 or $02)
SCOR1 = Raster score for player #1
SCOR2 = Raster score for player #2

Return Values

A = Destroyed
B = Destroyed
Y = Destroyed
U = Destroyed

Executive storage modifications

MESAGE, SIZRAS, TENSTY

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND, ASMESS
DEL13
INTENS, INTMAX
MRASTR
POSITD, POSITN
RASTER
ZERGND

SELOPT (OPTION)

Description:

Fetch number of players and options from player

Entry Address = $F7A9

Maximum Stack Requirements = 10 bytes

Entry Values

A = Number of possible players (0 – 9)
B = Number of possible options (0 – 9)

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed
U = Destroyed

LIST = $00
PLAYRS = Number of players selected
OPTION = Number of options selected

Executive storage modifications

EDGE, FADE, FADEA, FADEB, FADEC, FRAME, KEY0 – KEY7, LAG, MESAGE, SIZRAS, TENSITY, TONEA, TONEB, TRIGGR, TRIGGR + 1, XTMR0 – XTMR2

Control register modifications

ACNTRL, CNTRL, DAC, DDAC< PCNTRL, SHIFT, T1HOC, T1LOLC, T2LOLC, T2HOC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
BYTADD
D3TMR, DBNCE, DEFLOK, DEL13, DEL28, DPIO, DPRAM
FRWAIT
INPUT, INTENS, INTMAX
MRASTR, MSSPOS
POSIT2, POSITB, POSITD, POSITN, POSITX
RASTER, RSTPOS
SCLR, SCRADD, STKADD
XPLAY
ZERGND, ZERO

SETAMP (LOUDIN)

Description:
Set amplitude in ‘REQx’

Entry Address = $F9CA

Maximum Stack Requirements = 2 bytes

Entry Values
B = Volume setting
DP = $C8

TUNE = .

Return Values
A = Destroyed
X = Destroyed

Executive storage modifications
REQ3 – REQ5

SHIPX (SHIPSAT)

Description:

Display markers (counters remaining)

List Description:

byte 0 / 1 = positioning vector (Y:X)

Entry Address = $F391

Maximum Stack Requirements = 17 bytes

Entry Values

A = ASCII code of symbol
B = Number of markers remaining
X = Pointer to screen position list
DP = $D0

SIZRAS = ‘YX’ size of raster message

Return Values

A = $03
B = $01
X = $FBB4
U = Destroyed

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEL13, DEL28, DSHIP
MRASTR, MSSPOS
POSITD, POSITN
RASTER, RSTPOS
ZERGND

SINCOS

Description:
Calculate COSINE and SINE of ‘ANGLE’

Entry Address = $F5EF

Maximum Stack Requirements = 6 bytes

Entry Values
DP = $C8

ANGLE = Angle of rotation ($00 - $3F)

Return Values
D = Cosine of given angle

WCSINE = .
WSINE = .

Executive subroutines utilized
COSINE
SINE

SINE (SINGET)

Description:
Calculate SINE of value given in ‘A’

Entry Address = $F5D8

Maximum Stack Requirements = 2 bytes

Entry Values
A = Angle to be evaluated

Return Values
A = Sine of given angle
B = Sign / overflow for resulting sine
X = $FC6D (#RTRIGS)

SPLAY

Description:

Set tune sequence

Tune List Description:

byte 0 / 1 = Fade list pointer
2 / 3 = Vibrato list pointer
n = Note
$40 = Noise enable
$80 = Next channel enable
n+1 = Tone period ($80 = tune list terminator)

Fade List Description

<no description provided>

Vibrato List Description

<no description provided>

Entry Address = $F68D

Maximum Stack Requirements = 4 bytes

Entry Values

U = Tune list pointer
DP = $C8

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed
U = Destroyed

Executive storage modifications

DOREMI, FADE, FADEB, FADEC, NEWGET, REQ0 – REQD, RESTC, TONEB, TONEC, TSTAT, TUNE, VIBE

Executive subroutines utilized

ASPLAY
BCLR
CLRBLK
INTREQ
TPLAY
XPLAY

STKADD (SADD2)

Description:

Add value on stack to indicated score field

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F880

Maximum Stack Requirements = 4 bytes

Entry Values

S = .

LIST = $00

Return Values

A = Destroyed
B = Destroyed
S = Entry value + 2

TDIFFY (DIFTLS)

Description:

Draw according to ‘Diffy’ style list

List Description:

Byte 0 / 1 = Vector #1 (Y:X)

- .

Byte n / n+ 1 = Vector #n (Y:X)

Entry Address = $F3D8

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors – 1
B = Vector length (scale factor)
X = “DIFFY” list pointer
DP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY
LDIFFY
ZERGND

TDUFFY (DUFTLS)

Description:

Draw according to ‘Duffy’ style list

List Description:

Byte 0 / 1 = Vector #1 (Y:X)

- .

Byte n / n+ 1 = Vector #n (Y:X)

Entry Address = $F3B7

Maximum Stack Requirements = 2 bytes

Entry Values

A = Number of vectors – 1
B = Vector length (scale factor)
X = ‘DUFFY’ list pointer
FP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list + 2

LIST = $00

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
DIFFAB, DIFFY, DUFFAB, DUFFY
LDIFFY, LDUFFY
ZERGND

TPACK (PACB)

Description:

Draw according to ‘Packet’ style list

List Description:

Byte 0 / 1 / 2 = Vector #1 (C:Y:X)
- - - -
n / n+1 / n+2 = Vector #n (C:Y:X)
n+3 = $01 (packet terminator)

where C = $01 – packet terminator
$00 – draw blank line
$FF – draw solid line

Entry Address = $F40E

Maximum Stack Requirements = 2 bytes

Entry Values

B = Vector length (scale factor)
X = ‘PACKET’ list pointer
FP = $D0

ZSKIP = $00 – Skip integrator zeroing
!= $00 – Zero integrators

Return Values

A = Destroyed
B = Destroyed
X = End of list

Control register modifications

CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ACTGND
CZERO
PACKET
ZERGND

TPLAY (YOPLAY)

Description:

Set tune sequence

Tune List Description:

byte 0 / 1 = Fade list pointer
2 / 3 = Vibrato list pointer
n = Note
$40 = Noise enable
$80 = Next channel enable
n+1 = Tone period ($80 = tune list terminator)

Fade List Description

<no description provided>

Vibrato List Description

<no description provided>

Entry Address = $F692

Maximum Stack Requirements = 4 bytes

Entry Values

U = Tune list pointer
DP = $C8

DOREMI = .

Return Values

A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed
U = Destroyed

Executive storage modifications

FADE, FADEA, FADEB, FADEC, NEWGEN, REQ0 – REQD, RESTC, TONEB, TONEC, TSTAT, TUNE, VIBE

Executive subroutines utilized

BCLR
CLRBLK
INTREQ
XPLAY

TXTPOS (TEXTPOS)

Description:

Fetch position and display multiple text strings

Message List Description:

byte 0 / 1 = Absolute screen position (Y:X)
2 – n = Raster message string ($20 - $6F)
n +1 = Raster terminator ($80)

The rater message string (as above, bytes 0 thru n+1) can be repeated as necessary. The terminator for multiple message strings is a $00.

Entry Address = $F38C

Maximum Stack Requirements = 8 bytes

Entry Values

U = Message string pointer
DP = $D0

SIZRAS = 'YX' size of raster message

Return Values

A = $00
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGNC
DEL13, DEL28
MRASTR, MSSPOS
POSITD, POSITN
RASTER, RSTPOS
ZERGND

TXTSIZ (TEXSIZ)

Description:

Fetch size, position and display multiple text strings

Message List Description:

byte 0 / 1 = Raster message size (SIZRAS)
2 / 3 = Absolute screen position (Y:X)
4 – n = Raster message string ($20 - $6F)
n +1 = Raster terminator ($80)

The rater message string (as above, bytes 0 thru n+1) can be repeated as necessary. The terminator for multiple message strings is a $00.

Entry Address = $F385

Maximum Stack Requirements = 8 bytes

Entry Values

U = Message string pointer
DP = $D0

Return Values

A = $00
B = $01
X = $FBB4
U = End of message string + 1

Executive storage modifications

MESAGE
SIZRAS

Control register modifications

ACNTRL, CNTRL, DAC, PCNTRL, SHIFT, T1HOC, T1LOLC

Executive subroutines utilized

ABSAB, ABSB, ACTGND
DEL13, DEL28
MRASTR, MSSPOS
POSITD, POSITN
RASTER, RSTPOS, RSTSIZ
ZERGND

WAIT

Description:
Wait for frame boundary and input from controller

Entry Address = $EAF0

Maximum Stack Requirements = 17 bytes

Entry Values
No register parameters required

ACTPLY = Currently active player ($00 or $02)
FRMTIM = Frame to frame interval
SBTN = Button de-edge mask
SCOR1 = Raster score for player #1
SCOR2 = Rater score for player #2
SJOY = Joystick multiplexer enable (controller #1)

Return Values
A = Destroyed
B = Destroyed
X = Destroyed
Y = Destroyed
U = Destroyed
DP = $D0

LIST = $00

Executive storage modifications
EDGE, EPOT0, EPOT1, EPOT2, EPOT3, FRAME, KEY0 – KEY7, MESAGE, POT0 – POT3, SIZRAS, TENSITY, TMR1 – TMR4, TRIGGR

Control register modifications
ACNTRL, CNTRL, DAC, DDAC, PCNTRL, SHIFT, T1HOC, T1LOLC, T2LOLC

Executive subroutines utilized
ABSAB, ABSB, ACTGND, ASMESS
DBNCE, DEFLOK, DEL13, DPIO
FRWAIT
INPUT, INTENS, INTMAX
JOYBIT
MRASTR
POSIT2, POSITB, POSITD, POSITN, POSITX
RASTER
SCRMES
ZERGND, ZERO

WINNER

Description:

Compare two score fields

ASCII Score Field Description

byte 0 = Hundred thousand digit ($20, $30 - $39)
1 = Ten thousand digit ($20, $30 - $39)
2 = One thousand digit ($20, $30 - $39)
3 = Hundreds digit ($20, $30 - $39)
4 = Tens digit ($20, $30 - $39)
5 = Ones digit ($30 - $39)
6 = Score field terminator ($80)

Entry Address = $F8C7

Maximum Stack Requirements = 6 bytes

Entry Values

X = Score field #1
U = Score field #2

Return Values

A = $00 – Score #1 = Score #2
$01 – Score #1 > Score #2
$02 – Score #1 < Score #2

B = Destroyed
X = Destroyed
U = Destroyed

WRPSC (PSG)

Description:
Write to PSG and indicated mirror

Entry Address = $F259

Maximum Stack Requirements = 2 bytes

Entry Values
A = PSG address ($00 - $0D)
B = PSG data
X = Pointer to user mirror area
DP = $D0

Return Values
B = $01

Control register modifications
CNTRL, DAC

WRREG (PSGX)

Description:
Write to PSG and mirror

Entry Address = $F256

Maximum Stack Requirements = 2 bytes

Entry Values
A = PSG address ($00 - $0D)
B = PSG data
DP = $D0

Return Values
B = $01
X = $C800 (#REG0)

Executive storage modifications
REG0 - REGE

Control register modifications
CNTRL, DAC

Executive subroutines utilized
WRPSG

XPLAY

Description:
Terminate current tune

Entry Address = $F742

Maximum Stack Requirements = 6 bytes

Entry Values
DP = $C8

Return Values
A = $3F
B = $FF
X = $C83F (#REQ0)

TSTAT = .

Executive storage modifications
RESTC, REQ0 – REQD, TUNE

Executive subroutines utilized
BCLR
CLRBLK
INTREQ

ZERGND (ZEROIT)

Description:
Zero the integrators and set active ground.

Entry Address = $F354

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $03
B = $01

Control register modifications
CNTRL, DAC, PCNTRL, SHIFT

Executive subroutines utilized
ACTGND

ZERO (ZERO.)

Description:
Zero the integrators only

Entry Address = $F36B

Maximum Stack Requirements = 2 bytes

Entry Values
DP = $D0

Return Values
A = $00
B = $CC

Control register modifications
PCNTRL, SHIFT