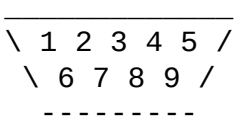
Atari Trackball#

Taming your machine with a mouse has long been a dream of the ATARI 8-bitters. The mouse had its origins around the 1960's, but did see much use until the 1980's when the Mac,ST and Amiga were unveiled. The mouse was also used by the engineering world as an input device for graphics workstations in the late 70's. The Track-ball is a close relative to the Mouse. A track-ball is basically the same device flipped over allowing the hand position the ball directly.

The Old ATARI did have some foresight in developing input devices for the 8-bit machines. The TrackBall is one of them. The Track-ball allows smooth tracking of 2 dimensional motion and its associated velocity. Atari Basic is too slow so it can not be read with atari basic. Faster languages such ACTION and Assembly can read the velocity vectors directly from the joystick registers. (mention something about which bits represent the direction and speed.) One thing the Atari Trak Ball lacks is a separate button that can function as the left mouse button. Since the mouse and the Track-ball are virtually the same device it should be possible to read and ST mouse using the Trak-ball read code. The 8-bits can read the ST left mouse if a pull-up resistor is added to pin 6.

Here's the pinout on the ST mouse.



1- up/xb
2- down/xa
3- left/ya
4- right/yb
5- not connected
6- Fire/left mouse button
7- +5vdc
8- ground
9- Joystick 1 Fire/Right Mouse button.

I have included 3 programs that demostrate the trak-balls ability to read direction and velocity. One program is written in basic with a short assemble used to read the T-ball input vector. The other 2 programs are written in ACTION!

```
10 REM :TBALL2.BAS
100 REM *** POKE MACHINE CODE ***
110 REM 1536-1619
111 DATA 104,169,0,133,212,133,213,173,0,211,41,2,133,205,160,255,173,0,211
112 DATA 41,2,197,205,240,2,230,212,133,205,136,208,240,173,0,211,41,1,208
113 DATA 6,165,212,9,128,133,212,173,0,211,41,8,133,205,160,255,173,0,211
114 DATA 41,8,197,205,240,2,230,213,133,205,136,208,240,173,0,211,41,4,208
115 DATA 6,165,213,9,128,133,213,96,-1
116 FOR I=1536 TO 1619:READ J:K=K+J:POKE I,J:NEXT I
117 IF K-11306 THEN ? "BAD DATA!":END
120 REM
130 GRAPHICS 0:POKE 710,0:POKE 752,1:REM BLACK BACKGROUND ,NO CUSOR
131 POSITION COL,ROW:? " ";:REM ERASE OLD OBJECT
```

```
200 REM READ TBALL
210 U=USR(1536):Y=INT(U/256):X=U-Y*256
220 IF X>127 THEN X=X-128:IF X THEN X=-X
221 IF Y>127 THEN Y=Y-128:IF Y THEN Y=-Y
310 POSITION COL,ROW:? " ";
320 COL=COL+X:REM CALCULATE NEW COLUNM
321 IF COL>39 THEN COL=39
322 IF COL<0 THEN COL=0
330 ROW=ROW+Y
331 IF ROW<0 THEN ROW=0
332 IF ROW>22 THEN ROW=22
340 POSITION COL,ROW:? "+";
350 GOTO 200
```

```
;TRACK1.ACT
;Display the value read from
;port 1 as track-ball values.
;9/87 Written bye Joe McFarland

PROC PrintT(BYTE val)
;Binary number print:
;Print byte in base Two.
;Modified to only print 4 LSbits.
BYTE mask,n
mask=$08
FOR n=0 TO 3 DO
  IF val&mask THEN
    Put('1)
  ELSE
    Put('0)
  FI
  mask==RSH 1
OD
RETURN

PROC Main()
BYTE b,cursor=752,consol=53279
cursor=1
Position(2,3)
PrintE("|||LHorizontal Dir 0=left, 1=right")
PrintE("||LHorizontal Rate")
PrintE("|Vertical Dir 0=up, 1=down")
PrintE("Vertical Rate")
DO
 b=Stick(0)
 Position(2,2)
 PrintT(b)
Until consol<>7
OD
cursor=1
RETURN
```

```
;TRACK3.ACT
;Rudimentary PM cursor positioning
;using Track-Ball peripheral
;PM routines added
;Single line rez.
;9/87
```

```
BYTE ARRAY player_base
BYTE ARRAY shposp(4) ;pm horiz shadow array.

CHAR ARRAY imagep=[
                  $F0 $90 $90 $90
                  $90 $90 $90 $F0

                  $00 $18 $18 $7E
                  $7E $18 $18 $00
                 ]
;*********************************

;Move specified player to the
;ABSOLUTE x location (0 to ?).

PROC MovePlayerHor(BYTE pl_num
                   BYTE pl_x)
BYTE ARRAY hposp=53248
 shposp(pl_num)=48+pl_x
 hposp(pl_num)=shposp(pl_num)
RETURN

;*********************************
MODULE
BYTE ARRAY old_pl_y(4)=~[0 0 0 0]
;Move specified player to the
;ABSOLUTE y location. (from 0 to ?.)

PROC MovePlayerVer(CARD pl_num
                   BYTE pl_y)
 BYTE playery
 CARD pl_offset

 pl_offset=player_base+$400+pl_num LSH 7
 Zero(pl_offset+old_pl_y(pl_num),8)
 playery=15+pl_y
 MoveBlock(pl_offset+playery,
          imagep+pl_num LSH 3,8)
 old_pl_y(pl_num)=playery
RETURN

;*********************************
;Move player to absolute x,y
;x=0 to ?, y=0 to ?

PROC MovePlayer(BYTE pl_num,pl_x,pl_y)
 MovePlayerHor(pl_num,pl_x)
 MovePlayerVer(pl_num,pl_y)
RETURN

;*********************************

PROC PlayerCursor()
BYTE pmbase=54279,gractl=53277,
     gprior=623
BYTE ARRAY pl_color=704,
     PMWidth(5)=$D008
BYTE ramtop=106,sdmactl=$22F
```

```
ramtop=$A0-8 ;presumes 40K of memory

Graphics(0)
player_base=(ramtop)*256

pmbase=player_base/256
sdmactl=32+8+2+16  ;no missles...
gractl=2        ;again no missles.
Zero(player_base,$800)
pl_color(0)=110
;pl_color(1)=70
gprior=1
MovePlayer(0,0,0)
;MovePlayer(1,4,10)
RETURN

;***********************************

PROC ClearPM()
BYTE ramtop=106,sdmactl=$22F
BYTE cursor=752
BYTE pmbase=54279,gractl=53277,
     gprior=623
cursor=0
gractl=0
sdmactl=32+2
;Zero(hposp,4)
ramtop=$A0
Graphics(0)
RETURN

;***********************************

PROC Main()
 BYTE cursor=752,consol=53279,
      left_margin=82
 BYTE lastx,lasty,vx,vy,st
 INT x,y,oldx=~[0],oldy=~[0]

PlayerCursor()
cursor=1
left_margin=0
SetColor(2,9,0)
PutE()
PrintE("This is a line of normal text.")
PutE()
PrintE("This is a line of inverse text.")
X=0 Y=0
lastx=0 lasty=0

WHILE consol&$01
DO
 st=stick(0)
 vx=st&$02
 vy=st&$08
 IF lastx<>vx
   THEN IF st&$01
           THEN x==+1
           ELSE x==-1
```

```
       FI
 FI
 IF lasty<>vy
   THEN IF st&$04
           THEN y==+1
           ELSE y==-1
        FI
 FI

 lastx=vx
 lasty=vy
 IF x>157 THEN x=157 FI
 IF y>201 THEN y=201 FI
 IF x<0  THEN x=0 FI
 IF y<17 THEN y=17 FI
IF oldx<>x OR oldy<>y THEN
 MovePlayer(0,X,Y)
FI
IF STRIG(0)
 THEN
 ELSE
 Position(0,10)
 PrintF("X=%I  %EY=%I  %E",X,Y)
FI
oldx=x
oldy=y
OD
ClearPM()
Graphics(0)
left_margin=2
cursor=0
RETURN
```

The next program demostrates ACTION!'s Ability to run 2 or more procedures at the same time. The Move_cursor routine runs independent of Main Proc. This Program is extra for Action Programers.

```
;TRACK4.ACT
;Rudimentary PM cursor positioning
;using Track-Ball peripheral
;PM routines added
;Single line rez.
;Vertical blank
;9/87

DEFINE SPEED="2"
DEFINE JMP="$4C",
       XITVBV="$E462",
SAVETEMPS=
"[$A2 $07 $B5 $C0 $48 $B5 $A0 $48 $B5 $80
10 dF1 $A5 $D3 $48]",
GETTEMPS=
"[$68 $85 $D3 $A2 $00 $68 $95 $A8 $68 $95 $80
  $68 $95 $A0 $68 $95 $C0 $E8 $E0 $08 $D0 $EF]"
CARD OldVbi,VBIvec=$224
BYTE critic=$42

BYTE ARRAY player_base
```

```
BYTE ARRAY shposp(4) ;pm horiz shadow array.

CHAR ARRAY imagep=[
                   $0 $90 $90 $90
                   $90 $90 $90 $F0

                   $00 $18 $18 $7E
                   $7E $18 $18 $00
                  ]
  INT x=~[0],y=~[0]
;***********************************

;Move specified player to the
;ABSOLUTE x location (0 to ?).

PROC MovePlayerHor(BYTE pl_num
                   BYTE pl_x)
BYTE ARRAY hposp=53248
 shposp(pl_num)=48+pl_x
 hposp(pl_num)=shposp(pl_num)
RETURN

;***********************************
MODULE
BYTE ARRAY old_pl_y(4)=~[0 0 0 0]
;Move specified player to the
;ABSOLUTE y location. (from 0 to ?.)

PROC MovePlayerVer(CARD pl_num
                   BYTE pl_y)
 BYTE playery
 CARD pl_offset

 pl_offset=player_base+$400+pl_num LSH 7
 Zero(pl_offset+old_pl_y(pl_num),8)
 playery=15+pl_y
 MoveBlock(pl_offset+playery,
          imagep+pl_num LSH 3,8)
 old_pl_y(pl_num)=playery
RETURN

;***********************************
;Move player to absolute x,y
;x=0 to ?, y=0 to ?

PROC MovePlayer(BYTE pl_num,pl_x,pl_y)
 MovePlayerHor(pl_num,pl_x)
 MovePlayerVer(pl_num,pl_y)
RETURN

;***********************************

PROC PlayerCursor()
BYTE pmbase=54279,gractl=53277,
     gprior=623
BYTE ARRAY pl_color=704,
     PMWidth(5)=$D008
BYTE ramtop=106,sdmactl=$22F
ramtop=$A0-8 ;presumes 40K of memory
```

```
Graphics(0)
player_base=(ramtop)*256

pmbase=player_base/256
sdmactl=32+8+2+16  ;no missles...
gractl=2        ;again no missles.
Zero(player_base,$800)
pl_color(0)=110
;pl_color(1)=70
gprior=1
MovePlayer(0,0,0)
;MovePlayer(1,4,10)
RETURN

;**********************************

PROC ClearPM()
BYTE ramtop=106,sdmactl=$22F
BYTE cursor=752
BYTE pmbase=54279,gractl=53277,
     gprior=623
cursor=0
gractl=0
sdmactl=32+2
;Zero(hposp,4)
ramtop=$A0
Graphics(0)
RETURN

;**********************************

PROC Move_Cursor()
 BYTE lastx=~[0],lasty=~[0],
      vx=~[0],vy=~[0],st
 INT oldx=~[0],oldy=~[0]
  SAVETEMPS

 st=stick(0)
 vx=st&$02
 vy=st&$08
 IF lastx<>vx
   THEN IF st&$01
           THEN x==+SPEED
           ELSE x==-SPEED
        FI
 FI
 IF lasty<>vy
   THEN IF st&$04
           THEN y==+SPEED
           ELSE y==-SPEED
        FI
 FI

 lastx=vx
 lasty=vy
 IF x>157 THEN x=157 FI
 IF y>201 THEN y=201 FI
 IF x<0  THEN x=0 FI
```

```
 IF y<17 THEN y=17 FI
IF oldx<>x OR oldy<>y THEN
 MovePlayer(0,X,Y)
FI
oldx=x
oldy=y
GETTEMPS     ; get temp registers
~[JMP XITVBV] ; exit the VBI

;************************************

PROC ClearVB()
critic=1
VBIvec=OldVBI
critic=0
RETURN

;************************************

PROC VBinst(); install the VBI
critic=1  ; turn off the interrupts
OldVBI=VBIvec
VBIvec=Move_Cursor ; VBI routine.
critic=0 ; turn the interrupts back on
RETURN

;**********************************

PROC Main()
 BYTE cursor=752,consol=53279,
      left_margin=82

PlayerCursor()
cursor=1
left_margin=0
SetColor(2,9,0)
PutE()
PrintE("This is a line of normal text.")
PutE()
PrintE("This is a line of inverse text.")
VBinst();This is where we start the Move_Cursor proc
WHILE consol&$01
DO
 Position(0,10)
 PrintF("X=%I  %EY=%I  %E",X,Y)
OD
ClearVB();This is where the Move_Cursor is terminated
ClearPM()
Graphics(0)
left_margin=2
cursor=0
RETURN
```