How to query the MultiJoy Interface#

Table of Contents

General Information#

The MultiJoy is an adapter to connect 8 or 16 Joysticks to a ATARI 8Bit (600XL, 800XL, 130XE, ...). The MultiJoy was designed by Radek Sterba (RASTER). You can find detail Information on this adapter on Radeks website

Mathy van Nisselroy has information about MultiJoy Games on his site

Assembler / Machine Language#

First you must initialize the communication direction of the ports. The pins of Joystick 1 work as Input and the pins of Joystick 0 work as output.

This has to be done once at the start of the program.

```
  LDA #$30   ; clear BIT 2 of PACTL (direction control register)
  STA $D302      ;PACTL, control read/write direction with PORTA
  LDA #$F0   ;4 upper bits=OUT (Joystick 1),4 lower bits=IN (Joystick 0)
  STA $D300      ;PORTA, set directions
  LDA #$34   ; restore OS default value for PACTL
  STA $D302      ;PACTL
```

Now we can query the joysticks: (proceed a delay 30 cycles at least between write to PORTA register and following reading of PORTA or TRIG0.)

```
  LDA #0      ;Number of the Joystick to query (0-7 for MultiJoy, 0-15 for MultiJoy16)
  ASL A       ; multiply by 16
  ASL A
  ASL A
  ASL A
  STA $D300       ;PORTA, select Joystick to read
  LDX #$06        ;Here is a delay 30 cycles before reading of PORTA
WAI DEX
  BNE WAI
  LDA $D300       ;PORTA, read value
  AND #$0F   ; mask out upper 4 bits
```

The joystick button can be queried after selection of the joystick with register TRIG0:

```
  LDA $D010  ;TRIG0
```

You must take care to synchronize the queries. Especially take care that not players on the first Joysticks have any unfair opportunities.

Basic / Turbo Basic#

BASIC#

Initializing:

```
POKE 54018,48 : REM control read/write direction with PORTA
POKE 54016,240 : REM 4 upper bits=OUT (Joystick 1),4 lower bits=IN (Joystick 0)
POKE 54018,52 : REM restore OS default value for PACTL
```

Query the Joystick:

```
POKE 54016,NUM*16 : REM NUM = Number of Joystick (0-7/0-15)
ST=PEEK(54016):ST=ST-INT(ST/16)*16 : REM read Joystick value
TR=PEEK(53264 ) : REM read Trigger value directly from GTIA
```

TURBO BASIC#

Initializing:

```
POKE $D302,$30
POKE $D300,$F0
POKE $D302,$34
```

Query the Joystick:

```
POKE $D300,NUM*$10 : REM NUM = Number of Joystick (0-7/0-15)
PAUSE %10 : REM wait for value
ST=PEEK($D300)&$0F : REM read Joystick value
REM ST=STICK(%0) - this does not work properly, because the Joystick value
REM which is read from register 632 is only updated every 1/50th second
REM TR=STRIG(%0) : REM this does not work either
TR=PEEK($D010): REM read it directly from the GTIA register
```

ACTION!#

Initializing:

```
PROC INITMULJOY()
 BYTE PACTL=$D302,PORTA=$D300
 ;
 PACTL==&$FB ; set bit 3 of pactl to 0. 1=use Port A for data input/output, 0=define d
 PORTA=$F0   ; set upper nibble of porta to write (1111) and lower nibble to read (000
 PACTL==%$04 ; set bit 3 of pactl to 1 again. 1=use Port A for data input/output, 0=de
RETURN
```

Query the Joystick:

```
PROC QUERYMULJOY(BYTE PL, BYTE POINTER STI,TRI)
 ; call procedure stating PL (range 0...8) and 2 pointers to variables to which the re
 ; e.g. "QUERYMULJOY(0,ST,TR)" check for Joystick 0, return values in the variable ST

 BYTE PORTA=$D300, TRIG0=$D010, WSYNC=$D40A
 ;
```

```
 PORTA=(PL LSH 4) ; LSH 4 is the same as multiply by 4
 PL=6
 DO PL==-1 UNTIL PL=0 OD         ; wait a wee bit so the PIA can adjust to the new val
 TRI^=TRIG0       ; now read the trigger
 STI^=(PORTA&$0F) ; and read the joystick, and blank out the high nibble by ANDing %00
RETURN

call the procedure with:

PROC MAIN()
 BYTE TRI,STI,PL
 BYTE POINTER S,T

 S=@STI ; pointer S now points to STI
 T=@TRI ; and T to TRI

 DO ; main loop
  FOR PL=0 to 8 DO
   QUERYMULJOY(PL,S,T) ; hand over the joystick number you want to query and two POINT
                       ; variables where you want the results.
   ; STI now holds the value for the stick
   ; TRI now holds the value for the trigger
   ; go ahead and do something with it...
   OD
 OD ; end of main loop
RETURN
```

Quick#

to be added

FORTH#

to be added