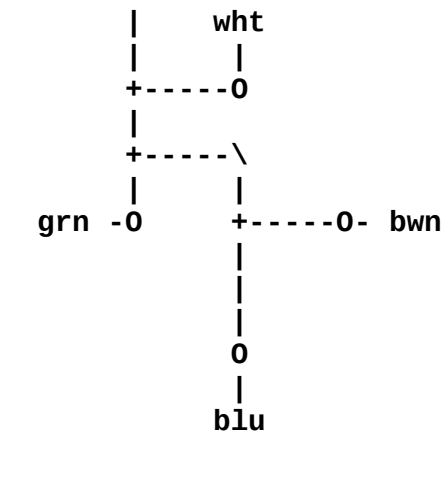
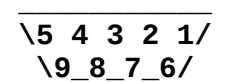
These are my notes on different types of ATARI controllers.
I decided to put them together after I read a posting by Mark Howard about the steering controllers and trackball on some ATARI 8-bit page.

Joysticks

circuital diagram



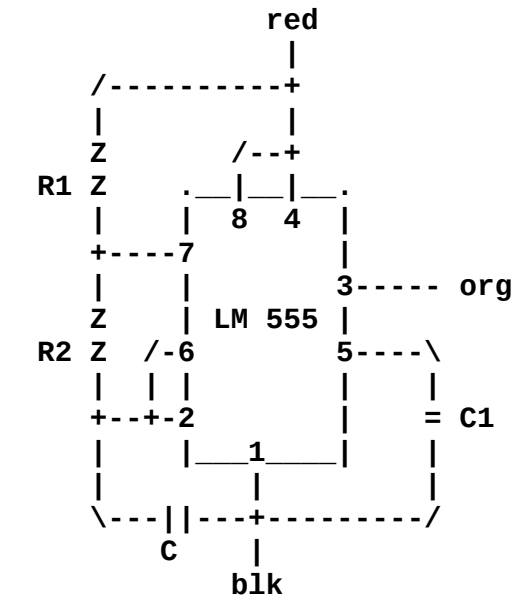
pin assignments



pin #	wire col	function
1	wht	^ (up)
2	blu	v (down)
3	grn	< (left)
4	bwn	> (right)
6	org	fire
(7	red	+5V) (*)
8	blk	GND

(*) only used in joysticks with autofire switch

autofire circuit



firing frequency:

f=1,44/[(R1+2*R2)*C]

(T=0,694*(R1+2*R2)*C)

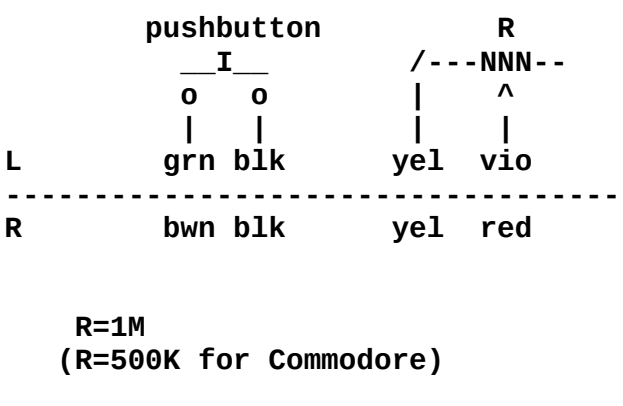
typical values:

C1=0,42 nF
R1=10 K
R2=10 K
C=100 nF

(These are the values I selected.)

Paddles

circuital diagram

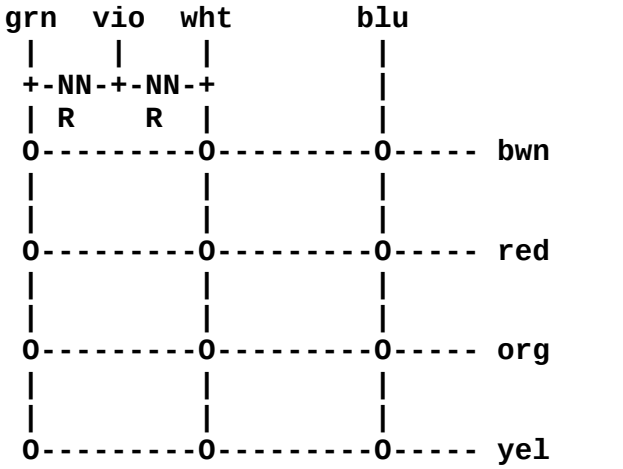


pin assignments

pin	wire	function
3	grn	L button
4	bwn	R button
5	red	R paddle
7	yel	+5V
8	blk	GND
9	vio	L paddle

Keyboard controllers/Video touch pad

circuital diagram



pin assignments

pin	wire	function
1	bwn	row 1
2	red	row 2
3	org	row 3
4	yel	row 4
5	grn	col 1
6	blu	col 3
7	vio	+5V
9	wht	col 2

Readings

The following program appeared in COMPUTE!, 2/87, "Readers Feedback" section, page 30. It allows to read a keyboard controller (or video touch pad, since they are internally identical) connected to port 1:

```
1 GRAPHICS 0
10 DIM ROW(3),I$(13),BUTTON$(1)
30 GOSUB 6000
60 POSITION 2,7:PRINT "CONTROLLER # 1:";
80 GOSUB 7000:POSITION 19,7:PRINT BUTTON$;
120 GOTO 80
6000 REM
```

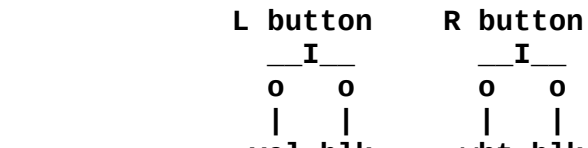
```
6010 POKE 54018,48:POKE 54016,255:POKE 54018,52:POKE 54016,221
6030 ROW(0)=238:ROW(1)=221:ROW(2)=187:ROW(3)=119
6040 I$=" 123456789*0#"
6050 RETURN
7000 PORT=54016
7010 P=1:PAD=0
7020 FOR J=0 TO 3
7030 POKE PORT, ROW(J)
7040 FOR I=1 TO 10:NEXT I
7050 IF PADDLE(PAD+1)>10 THEN P=J+J+J+2:GOTO 7090
7060 IF PADDLE(PAD)>10 THEN P=J+J+J+3:GOTO 7090
7070 IF STRIG(0)=0 THEN P=J+J+J+4:GOTO 7090
7080 NEXT J
7090 BUTTON$=I$(P,P)
7100 RETURN
```

STM1 Mouse

circuital diagram

(Are you nuts? It consists of 23 resistors, 2 capacitors, 2 movement sensors and an LM 339N quad-comparator IC!)

Well, at least the buttons:



pin assignments

pin #	wire col	function
1	bwn	L/R sense
2	org	L/R reference
3	grn	U/D sense
4	blu	U/D reference
6	yel	L button
7	red	+5V
8	blk	GND
9	wht	R button

readings

The STM1 Mouse works as an inverted trackball (which is, in fact, what it is). The left button corresponds to the joystick trigger; the right button must be read in more or less the same way as the keyboard controllers, I guess (I had to return the mouse I borrowed before finishing my analysis).

Mouse rolling forwards (up)

U/D REF	U/D SNS	
0	1	
1	1	<-\

Mouse rolling backwards (down)

U/D REF	U/D SNS	
0	0	
1	0	<-\

```
  0 1 --/                       0 0 --/
  ...                           ...
```

Mouse rolling left

```
L/R L/R
REF SNS
-------
  0 1
  1 1 <-\
  0 1 --/
  ...
```

Mouse rolling right

```
L/R L/R
REF SNS
-------
  0 0
  1 0 <-\
  0 0 --/
  ...
```

I tried to write a resident handler which should convert the mouse readings into joystick values and put them into the shadow register for joystick 2 (remember that on the ATARI XL/XE only joysticks 0 and 1 are in effect, and 2 & 3 just contain repeats of 0 and 1). Then, if you read STICK(2), you would have the translated values; STRIG(2) would give you the status of the right mouse button; and for the left button, you would still use STRIG(0). Unfortunately, the routine got hanged sooner or later, and I had to return the borrowed mouse before I could finish debugging it. Maybe I can borrow another and resume the task...

Marco Antonio Checa Funcke
Botoneros 270
Lima 33
Peru

(formerly reachable at jtkirk@urp.edu.pe
 but currently without an own account;
 however, postmast@urp.edu.pe is a close
 friend...)