skip to content

Controller port

6 pin mini DIN (like PS/2 keyboards and mouse). It is a serial port at 4800 baud, 8N1.

Pinout (using the standard numbering for mini-DIN connectors) :

1. VCC
2. CTS (from V.Smile)
3. Tx (from V.Smile)
4. GND
5. Rx (from controller)
6. RTS (from controller)

PulseView captures for some controllers

PulseView captures of booting with no controller, with a joystick (both with "Le Roi Lion") and with a keyboard (with "Clavier Tip Tap" cartridge)

Flow control

The CPU has a single UART that is used to communicate with both controllers. This requires flow control to make sure the two controllers don't try to communicate at the same time.

A controller is activated (both for transmission and reception) by setting its CTS pin high. Controllers can request attention from the console when they have data to send using their RTS line.

The RTS line changes can be detected using IRQ5. Events on the UART (transmit or receive complete) can be detected using IRQ3.

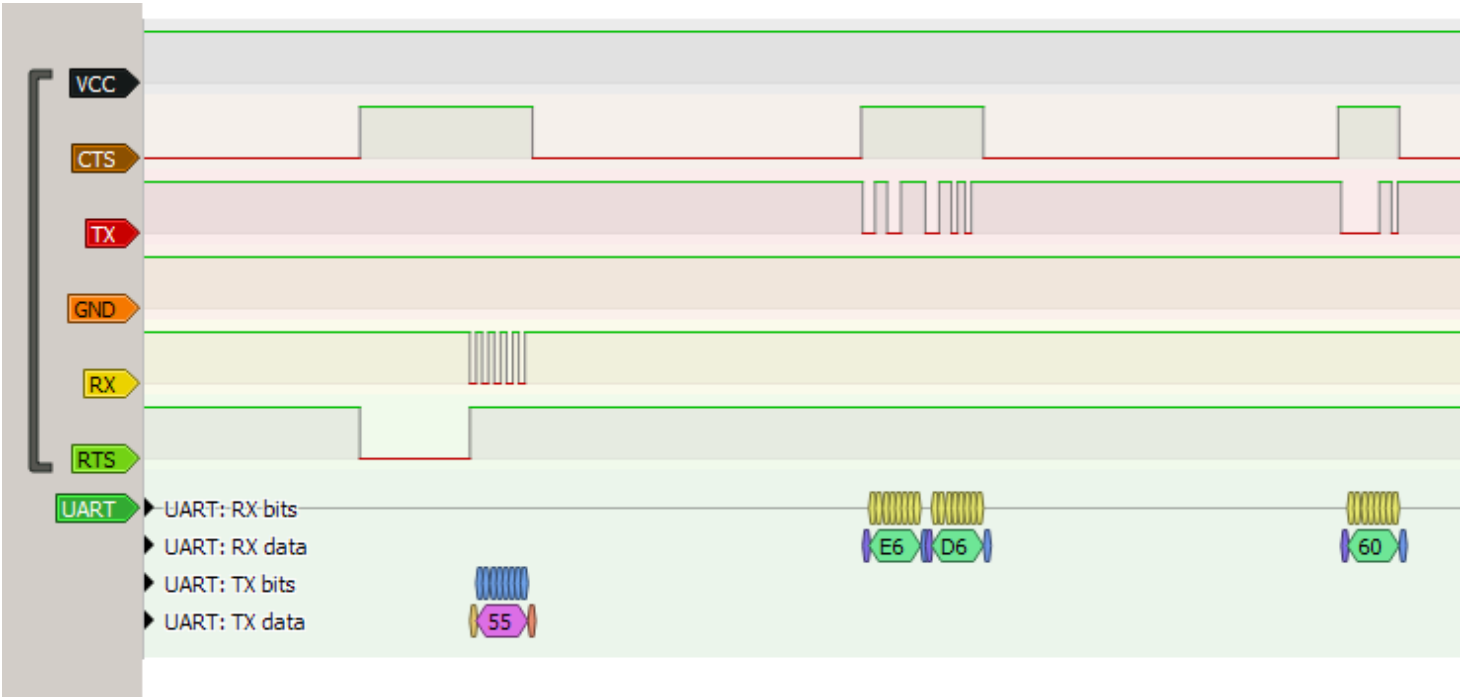
The general way to handle this is as follows, starting from an idle state with all CTS low and no pending data transfers

- If a controller has its RTS pin low, select it by setting the corresponding CTS high
- The controller will start transmitting data, receive that from the Rx register until “Rx ready” is cleared
- If needed, send a reply to the controller
- Make sure the reply is completely sent (“Tx buffer empty” in status register)
- Wait until RTS goes high (the controller has nothing to send anymore)
- Put CTS low again
- Wait a little (in case the controller was sending something just as you set CTS low) and read a possible last byte from the UART
- You are back to idle state

It is possible to send something to a controller even if it was not requesting RTS:

- Select the controller by setting the corresponding CTS high
- Send data to the controller
- Make sure the reply is completely sent (“Tx buffer empty” in status register)
- Check that RTS did not become low
- Put CTS low again
- Back to idle state

The controller keeps RTS down as long as it has more bytes to send.



Messages from the controller

When idle (no buttons touched), the console sends a byte every 20ms, it seems to be partially random. I've seen E6, D6, or 96.

Every second the controller sends 55 if nothing else is happening.

Common to joystick, dance mat and keyboard		
Button	**Press**	**Release**
OK	A1	A0
Quit	A2	A0
Help	A3	A0
ABC	A4	A0
Idle (nothing)	55	

Joystick

The joystick has 5 levels of precision in each direction. For example, C3 is “slightly up”, C7 is “all the way up”.

The 4 color buttons are allocated one bit each in the 9x range so it's possible to manage multiple of them being pressed at once.

The other buttons are Ax with x just being the button number, so it's not possible to handle multiple of them being pressed at the same time.

Joystick		
Button	**Press**	**Release**
Green	91	90
Blue	92	90
Yellow	94	90
Red	98	90
Up	C0 83 to C0 87	C0 80
Down	C0 8B to C0 8F	C0 80
Left	CB 80 to CF 80	C0 80
Right	C3 80 to C7 80	C0 80

The controller also sends Bx values for keepalive (see below).

Dance mat

Every press sends at least a “joystick position” 2-byte pair, and possibly an extra byte for the button itself (some buttons report as joystick moves, other as separate buttons). The mapping is not at all compatible with the joystick and seems a bit random. Note that for example 8B and 8D are different buttons, where on the joystick it would be different positions in the same direction.

Dance mat		
Button	**Press**	**Release**
1 / Red	C0 8B	C0 80
2 / Up	92 C0 80	90 C0 80
3 / Yellow	CB 80	C0 80
4 / Left	C0 8D	C0 80
5 / Middle	91 C0 80	90 C0 80
6 / Right	CD 80	C0 80
7 / Blue	A4 C0 80	A0 C0 80
8 / Down	94 C0 80	90 C0 80
9 / Green	98 C0 80	90 C0 80

Smart Keyboard (Clavier Tip Tap)

(sorry, I have the French/azerty version so key labels may not match up. The table is in row/column order)

Row 1 (top)		Row 2		Row 3		Row 4		Row 5	
Key	**Code**	**Key**	**Code**	**Key**	**Code**	**Key**	**Code**	**Key**	**Code**
Esc	A2	Dactylo	22	Caps	1A	Shift	A9/AA	Player 1	04
1	33	A	23			W	13	Help	A3
2	34	Z	24	Q	1B	X	14	Symbol	2C
3	35	E	25	S	1C	C	15	Space	05
4	37	R	27	D	1D	V	17	Player 2	0E
5	36	T	26	F	1F	B	16	Left	06
6	30	Y	20	G	1E	N	08	Down	0F
7	31	U	21	H	18	,	11	Right	0D
8	3E	I	3A	J	19	;	0C		
9	3F	O	3B	K	0A	:	2F		
0	38	P	3C	L	0B	Up	12		
º	29	¨	2A	M	01				
Backspace	39	Erase	3D	Enter	A1				

- Escape is mapped to Quit and works the same
- Help is mapped to Help and works the same
- Enter is mapped to OK and works the same
- Shift sends A9 on press and AA on release
- Other keys send their code on press, and code | C0 on release (so no code will be in the 90-AF range for either press or release to not conflict with the special buttons)

The joystick at the bottom of the keyboard is similar to the normal Joystick but uses different values (to avoid clashing with the keyboard release keycode range). There does not seem to be different values possible, it's all on or all off.

- Left: 7F 80
- Right: 77 80
- Down: 70 8F

- Up: 70 87

Boot sequence:

- Keyboard sends 52 52 52
- Console sends 0x02 0x02 0xE6 0xD6 0x60
- Keyboard sends language code
- Console: 0x70
- Keyboard: 0xBA

The language codes:

- 0x40: US
- 0x41: UK
- 0x42: French
- 0x44: German

Commands from the console

61, 62, 64 and 68 are sent in reply to color buttons presses. I suspect this controls the lights in the buttons. 60 is sent to turn the light off.

These are repeated every 20ms. After the controller sends 55 (idle), the game also returns to its idle reply (E6 followed by D6 for example) every 20ms.

Checksum/keepalive

The console sends 7x commands (where x is a “random” number) periodically and expects Bx replies from the controller.

The controller response algorithm to compute the correct reply is:

```
0xb0 | (((A + B + 0x0f) & 0x0f) ^ 0x05)
```

where A and B are the two most recent lower nibbles sent by the console in 7x commands. If only one nibble has been sent, the other nibble in the algorithm will have 0 as its value. The console can also reset the previous state by sending a Bx command instead which otherwise works like a 7x command.

The console must send the 7x commands, even if it ignores all the Bx replies from the controller. Without these commands, the controller eventually shuts down.