EclaireXL - Feature #11

Non-uk keyboard support (custom keyboard mapping)

04/05/2017 06:59 AM - foft

Status:	New	**Start date:**	04/05/2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:			

Description

Only a uk keyboard with stickers is supported right now.
Make this remappable via a file.

Related issues:

Related to Bug #36: Joystick is always emulated through the USB keyboard arro...	**Rejected**	**06/03/2017**

History

#1 - 04/05/2017 07:04 AM - admin

- Tracker changed from Bug to Feature

#2 - 06/02/2017 10:31 PM - admin

The mapping from ps2 to atari key code is here:
http://www.64kib.com/atarixlfpga_svn/trunk/atari_800xl/common/a8core/ps2_to_atari800.vhdl
(see http://www.computer-engineering.org/ps2keyboard/scancodes2.html)

I started with ps2 you see then added usb support on top. Since the usb layer is software anyway (on the zpu) I added a mapping from usb to ps2 here:
http://www.64kib.com/atarixlfpga_svn/trunk/atari_800xl/firmware/usb/keycodes.h
(See http://www.freebsddiary.org/APC/usb_hid_usages.php)

The atari keycodes are here:
http://atariage.com/forums/topic/188172-pokey-keyboard-codes/#entry2377650

I guess it would make sense to have a direct usb->atari one. If you can build that mapping in a simple machine readable format for several main regional keyboards then I make a utility to build a rom in a suitable format from it. Or if we have space in the firmware we can directly read it. Obviously run the format by me before making it all, so I can check its suitable.

#3 - 06/03/2017 12:24 PM - foft

- Related to Bug #36: Joystick is always emulated through the USB keyboard arrow keys added

#4 - 06/03/2017 12:24 PM - foft

- Subject changed from Non-uk keyboard support to Non-uk keyboard support (custom keyboard mapping)

#5 - 06/13/2017 10:20 PM - foft

Computer generated c code for TK2 is here:
http://atariage.com/forums/topic/263044-developerstesting-required-for-mini-itx-clone-system-eclairexl/page-6#entry3783384

Might be nice to integrate this, archive looks fairly large though in code size...

#6 - 06/17/2017 05:08 AM - 917k

foft wrote:

> Computer generated c code for TK2 is here:
> http://atariage.com/forums/topic/263044-developerstesting-required-for-mini-itx-clone-system-eclairexl/page-6#entry3783384
>
> Might be nice to integrate this, archive looks fairly large though in code size...

Is it worth the time,effort, and fpga space? Could a keymap just be read from a text file on the SD card?

#7 - 06/17/2017 08:17 AM - ndary

i think reading a key-map file from the SD CART is a good idea, an a8k file? if the key-map file is corrupt or missing a default map will be loaded from the fpga

#8 - 06/20/2017 07:41 PM - foft

What is an a8k file? Is there already a keyboard mapping file format we can use?

#9 - 06/20/2017 07:51 PM - ndary

- File Keyboard.zip added

its keyboard mapping from ATARI800WIN Emulator, see attached file

#10 - 06/21/2017 12:06 AM - 917k

I would say just use a text file, it's easier to create/edit. Also, The EclaireXL could have keys that have no equivalent in an emulator such as a keyboard shortcuts to map each drive, a key to increase the acceleration on the fly, freezer, etc.

#11 - 06/21/2017 02:37 AM - 917k

TEST

#12 - 06/21/2017 02:55 AM - 917k

Rethinking this, I'm not sure if we need to have a user-mappable keyboard at all as long as we have support for both ANSI and ISO keyboards. The only major change from the default ISO layout (besides moving a few keys to account for differences between the layouts) is that I moved the break key from way over in Siberia to the keyboard proper (which could also be done for the ISO layout). Here is my proposed mapping solution for ANSI..

USD MAPPING.png
and the modified code (with comments):

```
atari_keyboard(63)<=ps2_keys_reg(16#1C#); -- A
atari_keyboard(21)<=ps2_keys_reg(16#32#); -- B
atari_keyboard(18)<=ps2_keys_reg(16#21#); -- C
atari_keyboard(58)<=ps2_keys_reg(16#23#); -- D
atari_keyboard(42)<=ps2_keys_reg(16#24#); -- E
atari_keyboard(56)<=ps2_keys_reg(16#2B#); -- F
atari_keyboard(61)<=ps2_keys_reg(16#34#); -- G
atari_keyboard(57)<=ps2_keys_reg(16#33#); -- H
atari_keyboard(13)<=ps2_keys_reg(16#43#); --I
atari_keyboard(1)<=ps2_keys_reg(16#3B#); -- J
atari_keyboard(5)<=ps2_keys_reg(16#42#); -- K
atari_keyboard(0)<=ps2_keys_reg(16#4B#); -- L
atari_keyboard(37)<=ps2_keys_reg(16#3A#); -- M
atari_keyboard(35)<=ps2_keys_reg(16#31#); -- N
atari_keyboard(8)<=ps2_keys_reg(16#44#); -- O
atari_keyboard(10)<=ps2_keys_reg(16#4D#); -- P
atari_keyboard(47)<=ps2_keys_reg(16#15#); -- Q
atari_keyboard(40)<=ps2_keys_reg(16#2D#); -- R
atari_keyboard(62)<=ps2_keys_reg(16#1B#); -- S
atari_keyboard(45)<=ps2_keys_reg(16#2C#); -- T
atari_keyboard(11)<=ps2_keys_reg(16#3C#); -- U
atari_keyboard(16)<=ps2_keys_reg(16#2A#); -- V
atari_keyboard(46)<=ps2_keys_reg(16#1D#); -- W
atari_keyboard(22)<=ps2_keys_reg(16#22#); -- X
atari_keyboard(43)<=ps2_keys_reg(16#35#); -- Y
atari_keyboard(23)<=ps2_keys_reg(16#1A#); -- Z
atari_keyboard(50)<=ps2_keys_reg(16#45#); -- 0
atari_keyboard(31)<=ps2_keys_reg(16#16#); -- 1
atari_keyboard(30)<=ps2_keys_reg(16#1E#); -- 2
atari_keyboard(26)<=ps2_keys_reg(16#26#); -- 3
atari_keyboard(24)<=ps2_keys_reg(16#25#); -- 4
atari_keyboard(29)<=ps2_keys_reg(16#2E#); -- 5
atari_keyboard(27)<=ps2_keys_reg(16#36#); -- 6
atari_keyboard(51)<=ps2_keys_reg(16#3D#); -- 7
atari_keyboard(53)<=ps2_keys_reg(16#3E#); -- 8
atari_keyboard(48)<=ps2_keys_reg(16#46#); -- 9
atari_keyboard(17)<=ps2_keys_reg(16#16c#) or ps2_keys_reg(16#03#); -- HELP
atari_keyboard(52)<=ps2_keys_reg(16#66#); -- BACKSPACE
atari_keyboard(28)<=ps2_keys_reg(16#76#); -- ESCAPE
atari_keyboard(39)<=ps2_keys_reg(16#111#); -- INVERSE (NO IDEA WHERE THIS CODE 111 COMES FROM)
atari_keyboard(60)<=ps2_keys_reg(16#58#); -- CAPS
atari_keyboard(44)<=ps2_keys_reg(16#0D#); -- TAB
atari_keyboard(12)<=ps2_keys_reg(16#5A#); -- RETURN
atari_keyboard(33)<=ps2_keys_reg(16#29#); -- SPACE
```

```
atari_keyboard(54)<=ps2_keys_reg(16#4E#); -- LESS THAN
atari_keyboard(55)<=ps2_keys_reg(16#55#); -- GREATER THAN
atari_keyboard(15)<=ps2_keys_reg(16#5B#); -- EQUAL
atari_keyboard(14)<=ps2_keys_reg(16#54#); -- MINUS
atari_keyboard(6)<=ps2_keys_reg(16#4C#); -- PLUS (CHANGED TO 4C FROM 52 TO BTTER MATCH US KEYBOARD LAYOUT)
atari_keyboard(7)<=ps2_keys_reg(16#52#); -- ASTERIX (CHANGED TO 52 FROM 5D TO BETTER MATCH US KEYBOARD)
atari_keyboard(38)<=ps2_keys_reg(16#4A#); -- FORWARD SLASH
atari_keyboard(2)<=ps2_keys_reg(16#5D#); -- SEMI-COLON (CHANGED TO 5D FROM 4C TO BETTER MATCH US KEYBOARD)
atari_keyboard(32)<=ps2_keys_reg(16#41#); -- COMMA
atari_keyboard(34)<=ps2_keys_reg(16#49#); -- PERIOD

atari_keyboard(3)<=ps2_keys_reg(16#05#); -- 1200XL F1
atari_keyboard(4)<=ps2_keys_reg(16#06#); -- 1200XL F2
atari_keyboard(19)<=ps2_keys_reg(16#04#); -- 1200XL F3
atari_keyboard(20)<=ps2_keys_reg(16#0c#); -- 1200XL F4

consol_start_int<=ps2_keys_reg(16#0B#);
consol_select_int<=ps2_keys_reg(16#83#);
consol_option_int<=ps2_keys_reg(16#0a#);
shift_pressed<=ps2_keys_reg(16#12#) or ps2_keys_reg(16#59#);
control_pressed<=ps2_keys_reg(16#14#) or ps2_keys_reg(16#114#);
break_pressed<=ps2_keys_reg(16#0E#); -- BREAK (CHANGED FROM 77 TO 0E)

fkeys_int(0)<=ps2_keys_reg(16#05#);
fkeys_int(1)<=ps2_keys_reg(16#06#);
fkeys_int(2)<=ps2_keys_reg(16#04#);
fkeys_int(3)<=ps2_keys_reg(16#0C#);
fkeys_int(4)<=ps2_keys_reg(16#03#);
fkeys_int(5)<=ps2_keys_reg(16#0B#);
fkeys_int(6)<=ps2_keys_reg(16#83#);
fkeys_int(7)<=ps2_keys_reg(16#0a#);
fkeys_int(8)<=ps2_keys_reg(16#01#);
fkeys_int(9)<=ps2_keys_reg(16#09#);
fkeys_int(10)<=ps2_keys_reg(16#78#);
fkeys_int(11)<=ps2_keys_reg(16#07#);
```

#13 - 06/21/2017 09:44 AM - foft

Thanks

11 is left alt
extended 11 is right alt - I use 1xx for extended xx.

I'll build a test core with your changes, then we can add a switch later. I wonder if I can identify ansi/iso automatically.

#14 - 06/21/2017 08:56 PM - foft

I put up a build with your mapping as ansikbd.sof in the usual place. Let me know if it works as expected please.

#15 - 06/21/2017 11:30 PM - 917k

Mark,

It works great! Very Cool! Thank you for the test build. If we can add a toggle to the system menu and assign a variable such as **kb_type** to store the keyboard type, the code change would be pretty simple. I am not versed in VHDL at all but I think about all that would need to be done is something like:

```
---------------------------------------------------------------------------
-- (c) 2013 mark watson
-- I am happy for anyone to use this for non-commercial use.
-- If my vhdl files are used commercially or otherwise sold,
-- please contact me for explicit permission at scrameta (gmail).
-- This applies for source and binary form and derived works.
---------------------------------------------------------------------------

---------------------------------------------------------------------------
-- (ILoveSpeccy) Added PS2_KEYS Output
---------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

ENTITY ps2_to_atari800 IS
GENERIC
```

```
(
    ps2_enable : integer := 1;
    direct_enable : integer := 0
);
PORT
(
    CLK : IN STD_LOGIC;
    RESET_N : IN STD_LOGIC;
    PS2_CLK : IN STD_LOGIC := '1';
    PS2_DAT : IN STD_LOGIC := '1';
    INPUT : IN STD_LOGIC_VECTOR(31 downto 0) := (others=>'0');

    KEYBOARD_SCAN : IN STD_LOGIC_VECTOR(5 downto 0);
    KEYBOARD_RESPONSE : OUT STD_LOGIC_VECTOR(1 downto 0);

    CONSOL_START : OUT STD_LOGIC;
    CONSOL_SELECT : OUT STD_LOGIC;
    CONSOL_OPTION : OUT STD_LOGIC;

    FKEYS : OUT STD_LOGIC_VECTOR(11 downto 0);

    FREEZER_ACTIVATE : OUT STD_LOGIC;

    PS2_KEYS : OUT STD_LOGIC_VECTOR(511 downto 0);
    PS2_KEYS_NEXT_OUT : OUT STD_LOGIC_VECTOR(511 downto 0)
);
END ps2_to_atari800;

ARCHITECTURE vhdl OF ps2_to_atari800 IS
    signal ps2_keys_next : std_logic_vector(511 downto 0);
    signal ps2_keys_reg : std_logic_vector(511 downto 0);

    signal ps2_key_event : std_logic;
    signal ps2_key_value : std_logic_vector(7 downto 0);
    signal ps2_key_extended : std_logic;
    signal ps2_key_up : std_logic;

    signal direct_key_event : std_logic;
    signal direct_key_value : std_logic_vector(7 downto 0);
    signal direct_key_extended : std_logic;
    signal direct_key_up : std_logic;

    signal key_event : std_logic;
    signal key_value : std_logic_vector(7 downto 0);
    signal key_extended : std_logic;
    signal key_up : std_logic;

    signal CONSOL_START_INT : std_logic;
    signal CONSOL_SELECT_INT : std_logic;
    signal CONSOL_OPTION_INT : std_logic;

    signal FKEYS_INT : std_logic_vector(11 downto 0);

    signal FREEZER_ACTIVATE_INT : std_logic;

    signal atari_keyboard : std_logic_vector(63 downto 0);
    SIGNAL    SHIFT_PRESSED :  STD_LOGIC;
    SIGNAL    BREAK_PRESSED :  STD_LOGIC;
    SIGNAL    CONTROL_PRESSED :  STD_LOGIC;
BEGIN
    process(clk,reset_n)
    begin
        if (reset_n='0') then
            ps2_keys_reg <= (others=>'0');
        elsif (clk'event and clk='1') then
            ps2_keys_reg <= ps2_keys_next;
        end if;
    end process;

gen_ps2_on : if ps2_enable=1 generate
    keyboard1: entity work.ps2_keyboard
    PORT MAP
    (
        CLK => CLK,
        RESET_N => RESET_N,
```

```
        PS2_CLK => PS2_CLK,
        PS2_DAT => PS2_DAT,

        KEY_EVENT => PS2_KEY_EVENT,
        KEY_VALUE => PS2_KEY_VALUE,
        KEY_EXTENDED => PS2_KEY_EXTENDED,
        KEY_UP => PS2_KEY_UP
--        KEY_EVENT : OUT STD_LOGIC; -- high for 1 cycle on new key pressed(or repeated)/released
--        KEY_VALUE : OUT STD_LOGIC_VECTOR(7 downto 0); -- valid on event, raw scan code
--        KEY_EXTENDED : OUT STD_LOGIC;           -- valid on event, if scan code extended
--        KEY_UP : OUT STD_LOGIC                 -- value on event, if key released
    );
end generate;

gen_ps2_off : if ps2_enable=0 generate
    PS2_KEY_EVENT <= '0';
    PS2_KEY_VALUE <= (others=>'0');
    PS2_KEY_EXTENDED <= '0';
    PS2_KEY_UP <= '0';
end generate;

gen_direct_on : if direct_enable=1 generate
    direct_key_value <= input(7 downto 0);
    direct_key_extended <= input(12);
    direct_key_up <= not(input(16));
    direct_key_event <= '1';
end generate;

gen_direct_off : if direct_enable=0 generate
    DIRECT_KEY_EVENT <= '0';
    DIRECT_KEY_VALUE <= (others=>'0');
    DIRECT_KEY_EXTENDED <= '0';
    DIRECT_KEY_UP <= '0';
end generate;

    KEY_EVENT <= DIRECT_KEY_EVENT or PS2_KEY_EVENT;
    KEY_VALUE <= PS2_KEY_VALUE when PS2_KEY_EVENT='1' else DIRECT_KEY_VALUE;
    KEY_EXTENDED <= PS2_KEY_EXTENDED when PS2_KEY_EVENT='1' else DIRECT_KEY_EXTENDED;
    KEY_UP <= PS2_KEY_UP when PS2_KEY_EVENT='1' else DIRECT_KEY_UP;

    process(KEY_EVENT, KEY_VALUE, KEY_EXTENDED, KEY_UP, ps2_keys_reg)
    begin
        ps2_keys_next <= ps2_keys_reg;

    if (key_event = '1') then
        ps2_keys_next(to_integer(unsigned(KEY_EXTENDED&KEY_VALUE))) <= NOT(KEY_UP);
    end if;
end process;

-- map to atari key code
process(ps2_keys_reg)
begin
atari_keyboard <= (others=>'0');

shift_pressed <= '0';
control_pressed <= '0';
break_pressed <= '0';
consol_start_int <= '0';
consol_select_int <= '0';
consol_option_int <= '0';

atari_keyboard(63)<=ps2_keys_reg(16#1C#); -- A
atari_keyboard(21)<=ps2_keys_reg(16#32#); -- B
atari_keyboard(18)<=ps2_keys_reg(16#21#); -- C
atari_keyboard(58)<=ps2_keys_reg(16#23#); -- D
atari_keyboard(42)<=ps2_keys_reg(16#24#); -- E
atari_keyboard(56)<=ps2_keys_reg(16#2B#); -- F
atari_keyboard(61)<=ps2_keys_reg(16#34#); -- G
atari_keyboard(57)<=ps2_keys_reg(16#33#); -- H
atari_keyboard(13)<=ps2_keys_reg(16#43#); --I
atari_keyboard(1)<=ps2_keys_reg(16#3B#); -- J
atari_keyboard(5)<=ps2_keys_reg(16#42#); -- K
atari_keyboard(0)<=ps2_keys_reg(16#4B#); -- L
atari_keyboard(37)<=ps2_keys_reg(16#3A#); -- M
atari_keyboard(35)<=ps2_keys_reg(16#31#); -- N
```

```
atari_keyboard(8)<=ps2_keys_reg(16#44#); -- O
atari_keyboard(10)<=ps2_keys_reg(16#4D#); -- P
atari_keyboard(47)<=ps2_keys_reg(16#15#); -- Q
atari_keyboard(40)<=ps2_keys_reg(16#2D#); -- R
atari_keyboard(62)<=ps2_keys_reg(16#1B#); -- S
atari_keyboard(45)<=ps2_keys_reg(16#2C#); -- T
atari_keyboard(11)<=ps2_keys_reg(16#3C#); -- U
atari_keyboard(16)<=ps2_keys_reg(16#2A#); -- V
atari_keyboard(46)<=ps2_keys_reg(16#1D#); -- W
atari_keyboard(22)<=ps2_keys_reg(16#22#); -- X
atari_keyboard(43)<=ps2_keys_reg(16#35#); -- Y
atari_keyboard(23)<=ps2_keys_reg(16#1A#); -- Z
atari_keyboard(50)<=ps2_keys_reg(16#45#); -- 0
atari_keyboard(31)<=ps2_keys_reg(16#16#); -- 1
atari_keyboard(30)<=ps2_keys_reg(16#1E#); -- 2
atari_keyboard(26)<=ps2_keys_reg(16#26#); -- 3
atari_keyboard(24)<=ps2_keys_reg(16#25#); -- 4
atari_keyboard(29)<=ps2_keys_reg(16#2E#); -- 5
atari_keyboard(27)<=ps2_keys_reg(16#36#); -- 6
atari_keyboard(51)<=ps2_keys_reg(16#3D#); -- 7
atari_keyboard(53)<=ps2_keys_reg(16#3E#); -- 8
atari_keyboard(48)<=ps2_keys_reg(16#46#); -- 9
atari_keyboard(17)<=ps2_keys_reg(16#16c#) or ps2_keys_reg(16#03#); -- HELP
atari_keyboard(52)<=ps2_keys_reg(16#66#); -- BACKSPACE
atari_keyboard(28)<=ps2_keys_reg(16#76#); -- ESCAPE
atari_keyboard(39)<=ps2_keys_reg(16#111#); -- INVERSE (NO IDEA WHERE THIS CODE 111 COMES FROM)
atari_keyboard(60)<=ps2_keys_reg(16#58#); -- CAPS
atari_keyboard(44)<=ps2_keys_reg(16#0D#); -- TAB
atari_keyboard(12)<=ps2_keys_reg(16#5A#); -- RETURN
atari_keyboard(33)<=ps2_keys_reg(16#29#); -- SPACE
atari_keyboard(54)<=ps2_keys_reg(16#4E#); -- LESS THAN
atari_keyboard(55)<=ps2_keys_reg(16#55#); -- GREATER THAN
atari_keyboard(15)<=ps2_keys_reg(16#5B#); -- EQUAL
atari_keyboard(14)<=ps2_keys_reg(16#54#); -- MINUS
atari_keyboard(38)<=ps2_keys_reg(16#4A#); -- FORWARD SLASH
atari_keyboard(32)<=ps2_keys_reg(16#41#); -- COMMA
atari_keyboard(34)<=ps2_keys_reg(16#49#); -- PERIOD

-- ADD VARIABLE key_type AND SET STATE THROUGH OSD:
--
-- 0 = ISO (EUROPEAN STYLE KEYBOARD)
-- 1 = ANSI (AMERICAN STYLE KEYBOARD)

if (key_type = '0') then -- ISO KEYBOARD TYPE
     atari_keyboard(6)<=ps2_keys_reg(16#52#); -- PLUS
     atari_keyboard(7)<=ps2_keys_reg(16#5D#); -- ASTERIX
     atari_keyboard(2)<=ps2_keys_reg(16#4C#); -- SEMI-COLON
else  -- ANSI KEYBOARD TYPE
     atari_keyboard(6)<=ps2_keys_reg(16#4C#); -- PLUS (CHANGED TO 4C FROM 52 TO BTTER MATCH US KEYBOARD LAYOUT
)
     atari_keyboard(7)<=ps2_keys_reg(16#52#); -- ASTERIX (CHANGED TO 52 FROM 5D TO BETTER MATCH US KEYBOARD)
     atari_keyboard(2)<=ps2_keys_reg(16#5D#); -- SEMI-COLON (CHANGED TO 5D FROM 4C TO BETTER MATCH US KEYBOARD
)
endif;

atari_keyboard(3)<=ps2_keys_reg(16#05#); -- 1200XL F1
atari_keyboard(4)<=ps2_keys_reg(16#06#); -- 1200XL F2
atari_keyboard(19)<=ps2_keys_reg(16#04#); -- 1200XL F3
atari_keyboard(20)<=ps2_keys_reg(16#0c#); -- 1200XL F4

consol_start_int<=ps2_keys_reg(16#0B#);
consol_select_int<=ps2_keys_reg(16#83#);
consol_option_int<=ps2_keys_reg(16#0a#);
shift_pressed<=ps2_keys_reg(16#12#) or ps2_keys_reg(16#59#);
control_pressed<=ps2_keys_reg(16#14#) or ps2_keys_reg(16#114#);

break_pressed<=ps2_keys_reg(16#0E#) or ps2_keys_reg(16#77#); -- BREAK (ADDED 0E IN ADDITION TO 77)

fkeys_int(0)<=ps2_keys_reg(16#05#);
fkeys_int(1)<=ps2_keys_reg(16#06#);
fkeys_int(2)<=ps2_keys_reg(16#04#);
fkeys_int(3)<=ps2_keys_reg(16#0C#);
fkeys_int(4)<=ps2_keys_reg(16#03#);
fkeys_int(5)<=ps2_keys_reg(16#0B#);
fkeys_int(6)<=ps2_keys_reg(16#83#);
```

```
fkeys_int(7)<=ps2_keys_reg(16#0a#);
fkeys_int(8)<=ps2_keys_reg(16#01#);
fkeys_int(9)<=ps2_keys_reg(16#09#);
fkeys_int(10)<=ps2_keys_reg(16#78#);
fkeys_int(11)<=ps2_keys_reg(16#07#);

        -- use scroll lock or delete to activate freezer (same key on my keyboard + scroll lock does not seem
to work on mist!)
        freezer_activate_int <= ps2_keys_reg(16#7e#) or ps2_keys_reg(16#171#);
    end process;

    -- provide results as if we were a grid to pokey...
    process(keyboard_scan, atari_keyboard, control_pressed, shift_pressed, break_pressed)
        begin
            keyboard_response <= (others=>'1');

            if (atari_keyboard(to_integer(unsigned(not(keyboard_scan)))) = '1') then
                keyboard_response(0) <= '0';
            end if;

            if (keyboard_scan(5 downto 4)="00" and break_pressed = '1') then
                keyboard_response(1) <= '0';
            end if;

            if (keyboard_scan(5 downto 4)="10" and shift_pressed = '1') then
                keyboard_response(1) <= '0';
            end if;

            if (keyboard_scan(5 downto 4)="11" and control_pressed = '1') then
                keyboard_response(1) <= '0';
            end if;
    end process;

    -- outputs
    CONSOL_START <= CONSOL_START_INT;
    CONSOL_SELECT <= CONSOL_SELECT_INT;
    CONSOL_OPTION <= CONSOL_OPTION_INT;

    FKEYS <= FKEYS_INT;
    FREEZER_ACTIVATE <= FREEZER_ACTIVATE_INT;

    PS2_KEYS <= ps2_keys_reg;
    PS2_KEYS_NEXT_OUT <= ps2_keys_next;
END vhdl;
```

#16 - 06/22/2017 09:12 PM - foft

Yeah, that should work. You just described a multiplexor btw.

Got guests this weekend, so will do the plumbing next week.

#17 - 06/23/2017 04:31 AM - 917k

Okay, sounds great. So, is the system menu actually an Atari program or is it VHDL code?

#18 - 06/23/2017 07:27 AM - foft

It's written in c. The Atari core is built with a DMA port and the ability to freeze the 6502. Outside that there is another CPU (the zpu) that draws the menu, handles USB, se card, drive emulation etc. It has quite a lot of custom hardware support too.

I should draw an overview picture...

Files

Keyboard.zip	3.17 KB	06/20/2017	ndary