Introduction

This document describes how to add a new Mesa card to the pncconf utility.

Concepts

There are various types of Mesa cards that this configuration utility supports. Generally speaking, they need to be run with a specific firmware that configures the FPGA on the card to behave in a certain way with respect to the I/O of the card. In some cases this firmware is installed at runtime by LinuxCNC and in other cases it is loaded on the boards manually via some other process. In order for pncconf to be able to properly generate a configuration, it must know both the Mesa I/O card(s) you are configuring, as well as the firmware installed on those cards. In the case of the real time drivers (for example the `hm2_pci` driver), the pinout from message will be written to the `dmesg` facility on LinuxCNC startup. In the case of the upsace drivers (for example the `hm2_eth` driver), the pinouts are written to the `stdout` of the LinuxCNC process and so you can see the pinout by starting LinuxCNC from a command prompt.

Individual firmware images allow additional configuration, via the driver, to specify for instance the specific number of stepgens, pwmgens and encoders your specific machine requires. The remaining I/O is typically configured as GPIO. For example, a `7i93` card, can be configiured with an `SVST4_4d` firmware, which allows up to 4 steppers, 4 encoders, and 4 pwmgens. Using pncconf you can configure fewer of each of those, and the remainder will be configured as GPIO. Because things like stepgens, pwmgens, and encoders are allocated in the firmware in a particular order, typically you have to align the wiring of the machine to respect these limitations. This is a limitation of how the Mesa firmware is handled, not pncconf.

To add a board to pncconf, you must create a list data structure within the `private_data.py` `MESA_INTERNAL_FIRMWAREDATA` list variable that specifies the details of the card. There will be an entry in that data structure for each combination of card and firmware that you wish to have as configuration options.

Examples

An example of a data structure to configure the 7i93 card with a firmware capable of up to 4 pwmgens/encoders and 4 stepgens is shown below. The `7i93` has two 50-pin headers, which creates 48 total I/O pins, 24 pairs of signal and ground plus VCC and ground on each 50 pin header. This configuration is compatible with the `SVST4_4d` or `SVST4_8d` firmware available for the `7i93` card.

The expected data elements of the list are described programmatically in the `private_data.py` file. However, we will dissect this example a bit to provide some more context.

position in list	description	notes
0	board title	
1	board name	
2	firmware name for this configuration	
3	firmware directory	used when the firmware is installed at runtime
4	hal driver name	
5	max encoders	
6	number of pins per encoder	
7	max resolver gens	
8	number of pins per resolver gen	
9	max number of pwmgens	
10	number of pins per pwm gen	
11	max number of tppwmgens	
12	number of pins per tppwmgen	
13	max number of step gens	
14	number of pins per step gen	
15	max smart serial	
16	number of channels	
17	discovered sserial devices	A list
18 through 24	spare	
25	has watchdog	1 or 0
26	max GPIO	
27	low frequency rate	
28	hi frequency rate	
29	available connector numbers	A list of component type and logical number in the order they should appear. This should match the physical labels on the Mesa card.

The remainder of the data structure consists of list entries specifying the individual I/O pins of the card in order of how they will be allocated by the firmware. For instance the first six pins are used to provision two encoders with a,b and index connections: `[S.ENCB,1],[S.ENCA,1],[S.ENCB,0],[S.ENCA,0],[S.ENCI,1],[S.ENCI,0]`. The next two pins are two pwmgens:

`[S.PWMP,1],[S.PWMP,0]`. The order of these pins is not arbitrary, and must match what the specific firmware on the card is expecting. This information is supplied from Mesa in the form of pin files, typically shipped with the firmware files. In our case there are 48 entries because there are 48 I/O pins.

Allocation of the order of I/O pins to connectors is done at the Mesa hardware level and in our case the Mesa manual for the `7i93` specifies `CONECTOR P2 I/O 0..23` and `CONNECTOR P1 I/O 24..47`.

If logical number < 100 => GPIO can be changed to GPIOO or GPIOD at the start of linuxcnc, load time if you prefer (or run time) If logical number > 100 => GPIO can NOT be changed to GPIOO or GPIOD at the start of linuxcnc, load time (always input or always output) Value of number or number-100 corresponds with HAL Pin of Hostmot2 component The number 0 or 100 currently only has meaning for GPIO, SSR, INM and OUTM components. With GPIO the numbering uses the position in the firmware, starting with the first found GPIO as 0. SSR encodes the logical number within the 1xx number. ie 100 = zero component, 101 the #1 component etc.

The data structure from the code is replicated below. Using the data above, you should be able to decode this information:

```
["7i93-Internal Data", "7i93", "SVST4_4d", "7i93", "hm2_eth",
        4,3, 0,0, 4,3, 0,0, 4,2, 0,0, [],0,0,0,0,0,0,0, 1, 48, 33, 100, [2,1],
      # TAB 2
      [S.ENCB,1],[S.ENCA,1],[S.ENCB,0],[S.ENCA,0],[S.ENCI,1],[S.ENCI,0],[S.PWMP,1],
[S.PWMP,0],[S.PWMD,1],[S.PWMD,0],[S.PWME,1],[S.PWME,0],
      [S.ENCB,3],[S.ENCA,3],[S.ENCB,2],[S.ENCA,2],[S.ENCI,3],[S.ENCI,2],[S.PWMP,3],
[S.PWMP,2],[S.PWMD,3],[S.PWMD,2],[S.PWME,3],[S.PWME,2],
      # TAB 1
      [S.STEPA,0],[S.STEPB,0],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],
[S.STEPA,1],[S.STEPB,1],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],
      [S.STEPA,2],[S.STEPB,2],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],
[S.STEPA,3],[S.STEPB,3],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],[S.GPIOI,0],]
```

Knowing this, you can use a similar approach to add the specific card you want, or a new firmware configuration for an existing card, into pncconf.

Additional steps

You must also add the card to the `MESA_BOARD_META` dictionary data structure in the `private_data.py` file. For our example this entry is:

```
        '7i93':{'DRIVER':'hm2_eth','PINS_PER_CONNECTOR':24,'TOTAL_CONNECTORS':2},
```

Caveats

This approach will work for cards that conform to existing configuration patterns. If a card comes out that has new capabilities, that will of necessity require additional modification of pncconf.