Table of Contents

[<-] [^] [->]

List of figures

1 Parallel Port

1.1 Parport

Parport is a driver for the traditional PC parallel port. The port has a total of 17 physical pins. The original parallel port divided those pins into three groups: data, control, and status. The data group consists of 8 output pins, the control group consists of 4 pins, and the status group consists of 5 input pins.

In the early 1990's, the bidirectional parallel port was introduced, which allows the data group to be used for output or input. The HAL driver supports the bidirectional port, and allows the user to set the data group as either input or output. If configured as output, a port provides a total of 12 outputs and 5 inputs. If configured as input, it provides 4 outputs and 13 inputs.

In some parallel ports, the control group pins are open collectors, which may also be driven low by an external gate. On a board with open collector control pins, the "x" mode allows a more flexible mode with 8 outputs, and 9 inputs. In other parallel ports, the control group has push-pull drivers and cannot be used as an input.[1]

No other combinations are supported, and a port cannot be changed from input to output once the driver is installed. Figure [.] shows two block diagrams, one showing the driver when the data group is configured for output, and one showing it configured for input. For "x" mode, refer to the pin listing of "halcmd show pin" for pin direction assignment.

The parport driver can control up to 8 ports (defined by MAX_PORTS in hal_parport.c). The ports are numbered starting at zero.

1.1.1 Installing

```
loadrt hal_parport cfg="<config-string>"
```

1.1.1.1 Using the Port Index

I/O addresses below 16 are treated as port indexes. This is the simplest way to install the parport driver and cooperates with the Linux parport_pc driver if it is loaded.

```
loadrt hal_parport cfg="0"
```

Will use the address Linux has detected for parport0.

1.1.1.2 Using the Port Address

The configure string consists of a hex port address, followed by an optional direction, repeated for each port. The direction is "in", "out", or "x" and determines the direction of the physical pins 2 through 9, and whether to create input HAL pins for the physical control pins. If the direction is not specified, the data group defaults to output. For example:

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

This example installs drivers for one port at 0x0278, with pins 2-9 as outputs (by default, since neither "in" nor "out" was specified), one at 0x0378, with pins 2-9 as inputs, and one at 0x20A0, with pins 2-9 explicitly specified as outputs. Note that you must know the base address of the parallel port to properly configure the driver. For ISA bus ports, this is usually not a problem, since the port is almost always at a "well known" address, like 0278 or 0378 which is typically configured in the system BIOS. The address for a PCI card is usually shown in "lspci -v" in an "I/O ports" line, or in the kernel message log after executing "`sudo modprobe -a parport_pc`". There is no default address; if `<config-string>` does not contain at least one address, it is an error.

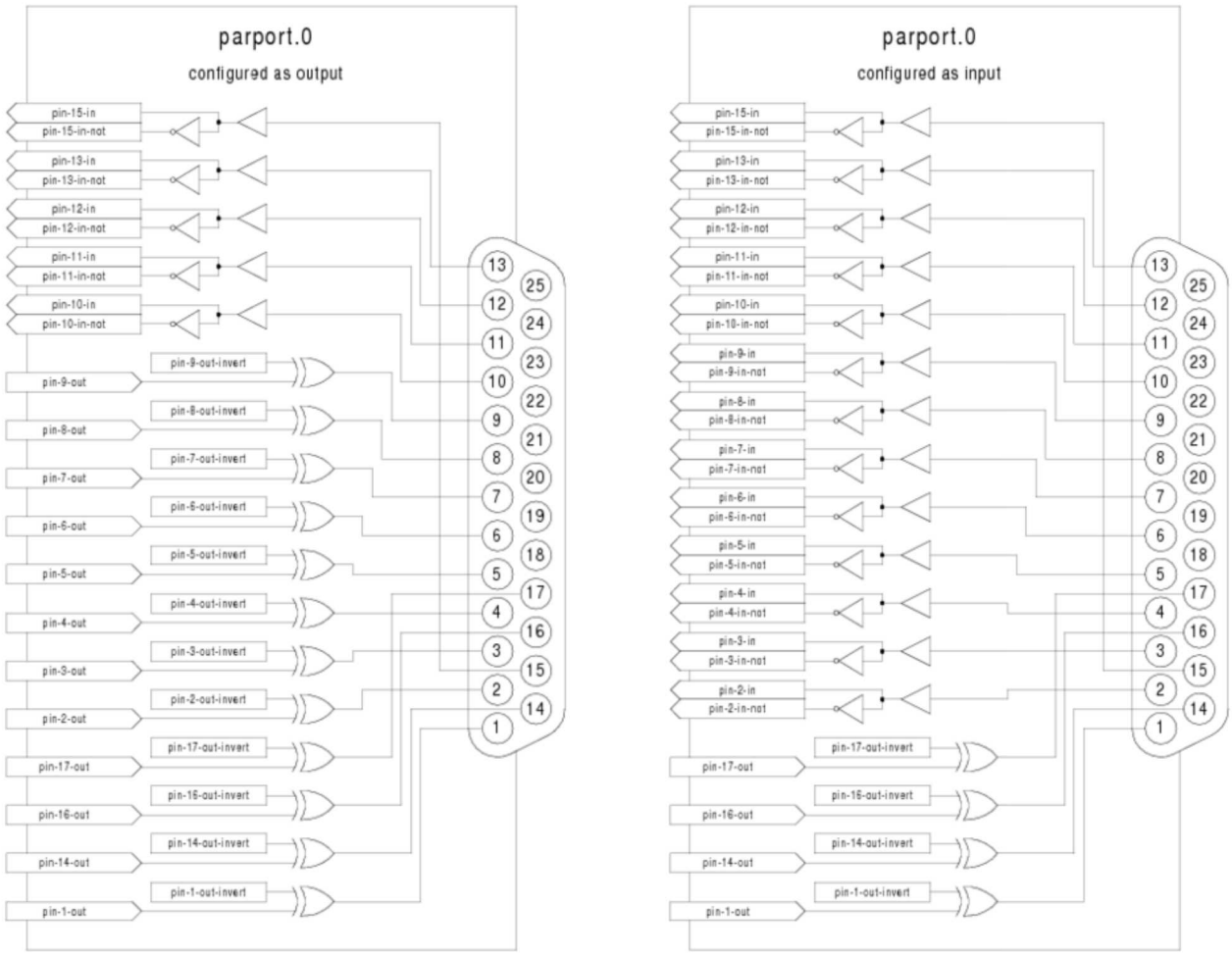

Figure: Parport Block Diagram

1.1.2 Pins

- (BIT) `parport.<portnum>.pin-<pinnum>-out` -- Drives a physical output pin.
- (BIT) `parport.<portnum>.pin-<pinnum>-in` -- Tracks a physical input pin.
- (BIT) `parport.<portnum>.pin-<pinnum>-in-not` -- Tracks a physical input pin, but inverted.

For each pin, `<portnum>` is the port number, and `<pinnum>` is the physical pin number in the 25 pin D-shell connector.

For each physical output pin, the driver creates a single HAL pin, for example `parport.0.pin-14-out`. Pins 2 through 9 are part of the data group and are output pins if the port is defined as an output port. (Output is the default.) Pins 1, 14, 16, and 17 are outputs in all modes. These HAL pins control the state of the corresponding physical pins.

For each physical input pin, the driver creates two HAL pins, for example `parport.0.pin-12-in` and `parport.0.pin-12-in-not`. Pins 10, 11, 12, 13, and 15 are always input pins. Pins 2 through 9 are input pins only if the port is defined as an input port. The `-in` HAL pin is TRUE if the physical pin is high, and FALSE if the physical pin is low. The `-in-not` HAL pin is inverted -- it is FALSE if the physical pin is high. By connecting a signal to one or the other, the user can determine the state of the input. In "x" mode, pins 1, 14, 16, and 17 are also input pins.

1.1.3 Parameters

- (BIT) `parport.<portnum>.pin-<pinnum>-out-invert` -- Inverts an output pin.
- (BIT) `parport.<portnum>.pin-<pinnum>-out-reset` (only for "out" pins) -- TRUE if this pin should be reset when the `-reset` function is executed.
- (U32) `parport.<portnum>.reset-time` -- The time (in nanoseconds) between a pin is set by `write` and reset by the `reset` function if it is enabled.

The `-invert` parameter determines whether an output pin is active high or active low. If `-invert` is FALSE, setting the HAL `-out` pin TRUE drives the physical pin high, and FALSE drives it low. If `-invert` is TRUE, then setting the HAL `-out` pin TRUE will drive the physical pin low.

1.1.4 Functions

- (FUNCT) `parport.<portnum>.read`-- Reads physical input pins of port `<portnum>` and updates HAL `-in` and `-in-not` pins.
- (FUNCT) `parport.read-all` -- Reads physical input pins of all ports and updates HAL `-in` and `-in-not` pins.
- (FUNCT) `parport.<portnum>.write` -- Reads HAL `-out` pins of port `<portnum>` and updates that port's physical output pins.
- (FUNCT) `parport.write-all` -- Reads HAL `-out` pins of all ports and updates all physical output pins.
- (FUNCT) `parport.<portnum>.reset` -- Waits until `reset-time` has elapsed since the associated `write`, then resets pins to values indicated by

`-out-invert` and `-out-invert` settings. `reset` must be later in the same thread as `write`. If `-reset` is TRUE, then the `reset` function will set the pin to the value of `-out-invert`. This can be used in conjunction with stepgen's `doublefreq` to produce one step per period. The stepgen stepspace for that pin must be set to 0 to enable doublefreq.

The individual functions are provided for situations where one port needs to be updated in a very fast thread, but other ports can be updated in a slower thread to save CPU time. It is probably not a good idea to use both an `-all` function and an individual function at the same time.

1.1.5 Common problems

If loading the module reports

> insmod: error inserting '/home/jepler/emc2/rtlib/hal_parport.ko':
> -1 Device or resource busy

then ensure that the standard kernel module `parport_pc` is not loaded[2] and that no other device in the system has claimed the I/O ports.

If the module loads but does not appear to function, then the port address is incorrect or the `probe_parport` module is required.

1.1.6 Using DoubleStep

To setup DoubleStep on the parallel port you must add the function parport.n.reset after parport.n.write and configure stepspace to 0 and the reset time wanted. So that step can be asserted on every period in HAL and then toggled off by parport after being asserted for time specificed by parport.n.reset-time.

For example:

```
loadrt hal_parport cfg="0x378 out"

setp parport.0.reset-time 5000

loadrt stepgen step_type=0,0,0

addf parport.0.read base-thread

addf stepgen.make-pulses base-thread

addf parport.0.write base-thread

addf parport.0.reset base-thread

addf stepgen.capture-position servo-thread

...

setp stepgen.0.steplen 1

setp stepgen.0.stepspace 0
```

1.2 probe_parport

In modern PCs, the parallel port may require plug and play (PNP) configuration before it can be used. The `probe_parport` module performs configuration of any PNP ports present, and should be loaded before `hal_parport`. On machines without PNP ports, it may be loaded but has no effect.

1.2.1 Installing

> loadrt probe_parport
>
> loadrt hal_parport ...

If the Linux kernel prints a message similar to

> parport: PnPBIOS parport detected.

when the parport_pc module is loaded (`sudo modprobe -a parport_pc; sudo rmmod parport_pc`) then use of this module is probably required.

Index

Footnotes

1 HAL cannot automatically determine if the "x" mode bidirectional pins are actually open collectors (OC). If they are not, they cannot be used as inputs, and attempting to drive them LOW from an external source can damage the hardware. To determine whether your port has "open collector" pins, load hal_parport in "x" mode. With no device attached, HAL should read the pin as TRUE. Next, insert a 470Ω resistor from one of the control pins to GND. If the resulting voltage on the control pin is close to 0V, and HAL now reads the pin as FALSE, then you have an OC port. If the resulting voltage is far from 0V, or HAL does not read the pin as FALSE, then your port cannot be used in "x" mode. The external hardware that drives the control pins should also use open collector gates (e.g., 74LS05). On some machines, BIOS settings may affect whether "x" mode can be used. "SPP" mode is most most likely to work. back

2 In the EMC packages for Ubuntu, the file /etc/modprobe.d/emc2 generally prevents parport_pc from being automatically loaded. back