

Introduction

This document describes "pinctrl", a simple, text-based interface to the Demand Peripherals BaseBoard series of FPGA based peripheral controller cards. Pinctrl lets you set or read individual FPGA pins as well as pause for a specified number of milliseconds.

While pinctrl is useful for prototyping and initial check-out of a BaseBoard or daughter card, it is not intended as its sole interface.

Overview

The pinctrl program can read or write any FPGA pin on a BaseBoard using a simple ASCII protocol. The protocol is described in the next section but these few commands illustrate the basic idea.

```
# Read the input button and flash LED7
p42=?      # Pin 42 goes to a button
p39=1      # Pin 39 is the MSB on the LEDs
d500       # Delay 500 milliseconds
p39=0      # Turn LED off
```

Valid commands are "#", "P", "S", and "D". Commands are case insensitive. Pin numbers are a string of digits forming a base 10 number which must be less than or equal to the highest pin number defined for that model of the BaseBoard.

Space and tab characters are completely ignored and can appear between any characters in a command.

The following two commands are equivalent:

```
P 3 9     = 1
p39=1
```

The read command returns the value of the pin on the same line as the command. That is, it replaces the question mark with the binary value of the pin.

Installation

Pinctrl is an FPGA program that you download to the BaseBoard as you would any other FPGA program. Download pinctrl from the Demand Peripherals web page that describes your BaseBoard. (Different BaseBoards have different pinctrl programs.)

Plug the BaseBoard into your PC and determine the name of the serial port assigned to it. On Linux you can (as root) do a 'tail -100 /var/log/messages'. Look for a string something like "usb 5-3.3: FTDI USB Serial Device converter now attached to ttyUSB0". In this case Linux assigned the BaseBoard to the serial port /dev/ttyUSB0. The commands to install pinctrl would be:

```
stty --file=/dev/ttyUSB0 -opost # We want raw output
cat pinctrl.bin > /dev/ttyUSB0
```

On Windows you can plug the board in and you'll be prompted with the usual "New Hardware Found" message. After loading the Windows driver for the FTDI245 you may be asked to load a driver for a USB serial port. Looking at System folder in the Control Panel will show which COM port was assigned to the USB serial port. (On the author's it was assigned to COM5 so I'll use that for the example.) The Windows command to

pinctrl is:

```
copy pinctrl.bin COM5:
```

Once pinctrl.bin is copied to the BaseBoard you can open a terminal emulator to the USB serial port and start issuing pinctrl commands.

Pinctrl Commands

Pin: The pin command sets or gets the value of a pin. When a value of one or zero is specified it sets a pin as an output and set the state. Specifying the value as a question mark set the pin as an input and reads the value of the pin. The response to a pin read replaces the question mark with the value of the pin. All pins are configured as inputs until the first write to a pin. The syntax of the write command is

```
[Pp]<pin#>=[0 | 1 | ?]\n
```

Strobe: The strobe command inverts the value of a pin, wait for one microsecond, and then restores the original value of the pin. Strobing an input pin has no effect and the pin must have previously been set as an output using a pin command. The strobe command is useful in reducing the size and complexity of programs that use output pins as clock lines. The syntax of the strobe command is

```
[Ss]<pin #>\n
```

Delay: The delay command tells pinctrl to wait a specified number of milliseconds before continuing to read commands from the input command stream. The delay command is useful to make a flashing LED on long enough to be visible. The syntax of the command is

```
[Dd]<0-65535>\n
```

Comment: The '#' is used to indicate a comment. All characters from the '#' to the end of the line are echoed but are otherwise ignored. For example:

```
# A comment can start in the first column ...
      # ... or after some ignored white space ...
P39=0 # ... or after some other command.
```

Command Notes:

The space and tab are considered to be white space and are completely ignored. Carriage return and linefeed are both equally line terminating characters.

Pinctrl indicated a syntax error by echoing the character in error and echoing all following characters to the end of the line as 'E' characters. For example, if you meant to write to pin 39 but hit a Q instead of a P, your input might be

```
q39=1 # LED7 on
```

but the echoed character stream would be

```
qEEEEEEEEEEEEEEEEEEEE
```

Pin Numbers

The pin number used in the pinctrl commands are dependent on which BaseBoard you are using. The assignment of pin numbers for the BaseBoard-4 are as follows:

Pin Number	Location
0 - 7	Connector SV1, pins 2, 4, 6, 8, 10, 12, 14, 16
8 - 15	Connector SV2, pins 2, 4, 6, 8, 10, 12, 14, 16
16 - 23	Connector SV3, pins 2, 4, 6, 8, 10, 12, 14, 16
24 - 31	Connector SV4, pins 2, 4, 6, 8, 10, 12, 14, 16
32 - 39	LED 0 to LED 7
40 - 42	Buttons S1 to S3 (always inputs)
43 - 45	Header pins GPIO_0 to GPIO_2