README

```
ArduLogic - Low Speed Logic Analyzer using the Arduino Hardware
===============================================================

ArduLogic is a simple logic analyzer solution using an Arduino UNO board (or
a plain Atmega328 chip with a serial link to a PC and the Arduino bootloader)
as probe. The following pins on the Arduino can be used for probing:

        Arduino PIN | AVR Pin | ArduLogic Pin Name
        ------------+---------+----------------------------
            2       |  PIND2  | D2 (preferred for triggering)
            3       |  PIND3  | D3 (preferred for triggering)
            4       |  PIND4  | D4
            5       |  PIND5  | D5
            6       |  PIND6  | D6
            7       |  PIND7  | D7
           A0       |  PINC0  | A0
           A1       |  PINC1  | A1
           A2       |  PINC2  | A2
           A3       |  PINC3  | A3
           A4       |  PINC4  | A4
           A5       |  PINC5  | A5

Additionally the following PINs provide additional debug information:

        Arduino PIN | AVR Pin | ArduLogic Debug Function
        ------------+---------+-------------------------------
            8       |  PINB0  | Checking trigger (non IRQ mode)
            9       |  PINB1  | Processing trigger
           10       |  PINB2  | Serial activity (pulse)
           11       |  PINB3  | Serial activity (toggle)
```

```
          12       |  PINB4  | Capture in progress
          13 (LED) |  PINB5  | Error indicator
```

ArduLogic can use any pin as trigger pin (clock signals, etc), but IRQs are
only used when all triggers are on the pins D2 and/or D3. So it is highly
recommended to connect clock lines or other trigger lines to eighter pin
D2 or pin D3.

The data is transfered from the Arduino to the PC using a 2 megabaud serial
link and the data is transfered in bit-packed form. Thus higher sampling rates
can be achieved when fewer pins are used.

The data acquired by ArduLogic is written to a VCD file that can then
be inspected using a VCD viewer such as gtkwave. Note that you need Linux
running on the PC in order to use ArduLogic.

The ArduLogic command line tool reads in a configuration file that describes
the the type of data acquisition that should be performed and auto-generates
an Arduino firmware image that can be used to acquire the data. It then
programms the Arduino with this firmware, acquires the data and stores it
in a .vcd file.

Example session:

```
        $ vi example.al
        <create or modify configuration>

        $ ./ardulogic -p -t /dev/ttyACM0 -V example.vcd example.al
        <press Ctrl-C to stop recording>

        $ gtkwave example.vcd
        <inspect signals in gtkwave gui>
```

Configuration file syntax:
==========================

In the following syntax rules the placeholder <PIN> stands for one of the
ArduLogic pin names as stated above (e.g. D2 or A5).

The placeholder <EDGE> stand eighter for `posedge' or `negedge'.

trigger <EDGE> <PIN>

Acquire data whenever the specified edge occurs on the specified pin.
Not that `trigger' statements can't be used together with `decode'
statements in the same configuration file. But it is perfectly fine
to use as many `trigger' statements in a configuration file as you want.

trigger <Frequency>

Instead of trigger on any specific event it is possible to simply

```
use a free running trigger with the specified frequency. The
frequency must be specified using the syntax <n>Hz or <n>kHz where
<n> is an integer number.

capture <PIN> [...]
-------------------

Add the specified pin(s) to the list of pins to capture. Note that
it is possible to trigger on a pin without capturing it. It's possible
to mix `capture' statements with `decode' statements, e.g. to add
information from additional sideband signals to the vcd file.

pullup <PIN> [...]
------------------

Enable the internal AVR pullup resistors on the specified pin(s).

label <PIN> "<NAME>"
--------------------

Rename a PIN to the given name. This only changes the PIN label
in the generated VCD file. All commands in the config file must still
use the regaluar "An" or "Dn" pin names to refer to the pins in question.

decode spi <EDGE> <ORDER> <PIN-SCK> [!] <PIN-CS> <PIN-MOSI> <PIN-MISO>
----------------------------------------------------------------------

This configures capturing and decoding an SPI bus. The <EDGE> specifies if data
on the bus should be captured on the positive or negative edge of SCK. The
<ORDER> parameter must be `msb' for MSB first byte ordering or `lsb' for LSB
first byte order. The `!' in front of CS specifies if the CS pin is inverted.

decode i2c <PIN-SCL> <PIN-SDA>
------------------------------

This configures capturing and decoding an I2C bus.

decode jtag <PIN-TCK> <PIN-TMS> <PIN-TDI> <PIN-TDO>
---------------------------------------------------

This configures capturing and decoding a JTAG bus.

Note that you can't mix multiple `decode' statements and you can't mix
`decode' statements with `trigger' statements in the same configuration file.
```

Releases

No releases published

Packages

No packages published

Languages