Benchmarking Raspberry Pi GPIO Speed

08/24/15

The following information was found at

http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

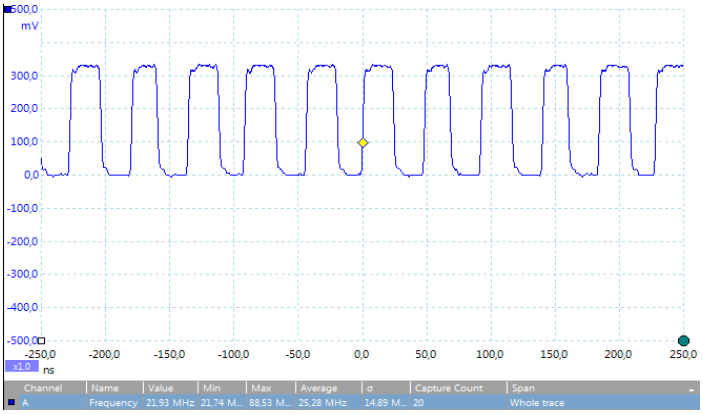
Language	Library	Tested / version	Square wave
Shell	/proc/mem access	2015-02-14	2.8 kHz
Shell / gpio utility	WiringPi gpio utility	2015-02-15 / 2.25	40 Hz
Python	RPi.GPIO	2015-02-15 / 0.5.10	70 kHz
Python	wiringpi2 bindings	2015-02-15 / latest github	28 kHz
Ruby	wiringpi bindings	2015-02-15 / latest gem (1.1.0)	21 kHz
C	Native library	2015-02-15 / latest RaspPi wiki code	22 MHz
C	BCM 2835	2015-02-15 / 1.38	5.4 MHz
C	wiringPi	2015-02-15 / 2.25	4.1 – 4.6 MHz
Perl	BCM 2835	2015-02-15 / 1.9	48 kHz

C: Maximum performance

The Raspberry Pi Wiki gives a nice C code example for true hardware-level access to the GPIO. The interfacing is slightly more difficult, but code isn't too bad. I took the example program and simplified the main method after `setup_io()` to this:

```
// Set GPIO pin 4 to output
INP_GPIO(4); // must use INP_GPIO before we can use OUT_GPIO
OUT_GPIO(4);
while(1) {
  GPIO_SET = 1<<4;
  GPIO_CLR = 1<<4;
}
```

Without any optimizations, I got an excellent 14 MHz square wave. Adding -O3 to the compile command (`gcc -O3 strobe.c -o strobe`) increases the rate to hefty 22 MHz. Measuring the waveform with oscilloscope starts to require **VERY** short wiring between probe and ground, otherwise it just looks like a sine wave due to capacitance in helper wires!



The following information was found at
http://elinux.org/RPi_Low-level_peripherals#C

Interfacing with GPIO pins

GPIO voltage levels are 3.3 V and are not 5 V tolerant. There is no over-voltage protection on the board - the intention is that people interested in serious interfacing will use an external board with buffers, level conversion and analog I/O rather than soldering directly onto the main board.

All the GPIO pins can be reconfigured to provide alternate functions, SPI, PWM, I²C and so. At reset only pins GPIO 14 & 15 are assigned to the alternate function UART, these two can be switched back to GPIO to provide a total of 17 GPIO pins[7]. Each of their functions and full details of how to access are detailed in the chipset datasheet [8].

Each GPIO can interrupt, high/low/rise/fall/change.[9][10] There is currently no support for GPIO interrupts in the official kernel, however a patch exists, requiring compilation of modified source tree. [11] The 'Raspbian "wheezy"' [12] version that is currently recommended for starters already includes GPIO interrupts.

GPIO input hysteresis (Schmitt trigger) can be on or off, output slew rate can be fast or limited, and source and sink current is configurable from 2 mA up to 16 mA. Note that chipset GPIO pins 0-27 are in the same block and these properties are set per block, not per pin. See GPIO Datasheet Addendum - GPIO Pads Control. Particular attention should be applied to the note regarding SSO (Simultaneous Switching Outputs): to avoid interference, driving currents should be kept as low as possible.

The available alternative functions and their corresponding pins are detailed below. These numbers are in reference to the chipset documentation and may not match the numbers exposed in Linux. Only fully usable functions are detailed, for some alternative functions not all the necessary pins are available for the funtionality to be actually used.

There is also some information on the Tutorial on Easy GPIO Hardware & Software.

Kernel boot messages go to the UART at 115200 bit/s - there are more details on the serial port page

P1 Header pinout, top row

Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-02	5V0		Supply through input poly fuse		
P1-04	5V0		Supply through input poly fuse		
P1-06	GND				
P1-08	GPIO 14		Boot to Alt 0 ->	UART0_TXD	ALT5 = UART1_TXD
P1-10	GPIO 15		Boot to Alt 0 ->	UART0_RXD	ALT5 = UART1_RXD
P1-12	GPIO 18			PCM_CLK	ALT4 = SPI1_CE0_N ALT5 = PWM0
P1-14	GND				
P1-16	GPIO23				ALT3 = SD1_CMD ALT4 = ARM_RTCK
P1-18	GPIO24				ALT3 = SD1_DAT0 ALT4 = ARM_TDO
P1-20	GND				
P1-22	GPIO25				ALT3 = SD1_DAT1 ALT4 = ARM_TCK
P1-24	GPIO08			SPI0_CE0_N	
P1-26	GPIO07			SPI0_CE1_N	

P1 Header pinout, bottom row

Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-01	3.3 V		50 mA max (01 & 17)		
P1-03	GPIO 0	**GPIO 2**	1K8 pull up resistor	I2C0_SDA / **I2C1_SDA**	
P1-05	GPIO 1	**GPIO 3**	1K8 pull up resistor	I2C0_SCL / **I2C1_SCL**	

P1-07	GPIO 4		GPCLK0	ALT5 = ARM_TDI
P1-09	GND			
P1-11	GPIO17			ALT3 = UART0_RTS ALT4 = SPI1_CE1_N ALT5 = UART1_RTS
P1-13	GPIO21 **GPIO27**		PCM_DOUT / **reserved**	ALT4 = SPI1_SCLK ALT5 = GPCLK1 / **ALT3 = SD1_DAT3 ALT4 = ARM_TMS**
P1-15	GPIO22			ALT3 = SD1_CLK ALT4 = ARM_TRST
P1-17	3.3 V	50 mA max (01 & 17)		
P1-19	GPIO10		SPI0_MOSI	
P1-21	GPIO9		SPI0_MISO	
P1-23	GPIO11		SPI0_SCLK	
P1-25	GND			

Colour legend
+5 V
+3.3 V
Ground, 0V
UART
GPIO
SPI
I²C

Referring to pins on the Expansion header

The header is referred to as "The GPIO Connector (P1)". To avoid nomenclature confusion between Broadcom signal names on the SoC and pin names on the expansion header, the following naming is highly recommended:

- The expansion header is referred to as "Expansion Header" or "GPIO Connector (P1)"
- Pins on the GPIO connector (P1) are referred to as P1-01, etc.
- Names GPIO0, GPIO1, GPIOx-ALTy, etc. refer to the signal names on the SoC as enumerated in the Broadcom datasheet, where "x" matches BCM2835 number (without leading zero) and "y" is the alternate number column 0 to 5 on page 102-103 of the Broadcom document. For example, depending on what you are describing, use either "GPIO7" to refer to a row of the table, and "GPIO7-ALT0" would refer to a specific cell of the table.
- When refering to signal names, the Broadcom name should be modified slightly to minimize confusion. The Broadcom SPI bus pin names are fine, such as "SPI0_*" and "SPI1_*", but they did not do the same on the I²C and UART pins. Instead of using "SDA0" and "SCL0", "I2C0_SDA" and "I2C0_SCL" should be used; and "UART0_TXD" and "UART0_RXD"

instead of "TX" or "TXD" and "RX" or "RXD".

Power pins

The maximum permitted current draw from the 3.3 V pins is 50 mA.

Maximum permitted current draw from the 5 V pin is the USB input current (usually 1 A) minus any current draw from the rest of the board.[18]

- Model A: 1000 mA - 500 mA -> max current draw: 500 mA
- Model B: 1000 mA - 700 mA -> max current draw: 300 mA

Be very careful with the 5 V pins P1-02 and P1-04, because if you short 5 V to any other P1 pin you may permanently damage your RasPi. Before probing P1, it is a good idea to strip short pieces of insulation off a wire and push them over the 5 V pins are not accidentally shorted with a probe.

GPIO hardware hacking

The complete list of chipset GPIO pins which are available on the GPIO connector is:

0, 1, 4, 7, 8, 9, 10, 11, 14, 15, 17, 18, 21, 22, 23, 24, 25

(on the Revision2.0 RaspberryPis, this list changes to: 2, 3, 4, 7, 8, 9, 10, 11, 14, 15, 17, 18, 22, 23, 24, 25, 27, with 28, 29, 30, 31 additionally available on the P5 header)

As noted above, P1-03 and P1-05 (SDA0 and SCL0 / SDA1 and SCL1) have 1.8 kohm pull-up resistors to 3.3 V.

If 17 GPIOs are not sufficient for a project, there are a few other signals potentially available, with varying levels of software and hardware (soldering iron) hackery skills:

GPIO02, 03, 05 and 27 are available on S5 (the CSI interface) when a camera peripheral is not connected to that socket, and are configured by default to provide the functions SDA1, SCL1, CAM_CLK and CAM_GPIO respectively. SDA1 and SCL1 have 1K6 pull-up resistors to 3.3 V.

GPIO06 is LAN_RUN and is available on pad 12 of the footprint for IC3 on the Model A. On Model B, it is in use for the Ethernet function.

There are a few other chipset GPIO pins accessible on the PCB but are in use:

- GPIO16 drives status LED D5 (usually SD card access indicator)
- GPIO28-31 are used by the board ID and are connected to resistors R3 to R10 (only on Rev1.0 boards).
- GPIO40 and 45 are used by analogue audio and support PWM. They connect to the analogue audio circuitry via R21 and R27 respectively.
- GPIO46 is HDMI hotplug detect (goes to pin 6 of IC1).
- GPIO47 to 53 are used by the SD card interface. In particular, GPIO47 is SD card detect (this would seem to be a good candidate for re-use). GPIO47 is connected to the SD card interface card detect switch; GPIO48 to 53 are connected to the SD card interface via resistors R45 to R50.

Internal Pull-Ups & Pull-Downs

The GPIO ports include the ability to enable and disable internal pull-up or pull-down resistors (see below for code examples/support of this):

- Pull-up is 50 kOhm - 65 kOhm
- Pull-down is 50 kOhm - 60 kOhm

Driver support

The Foundation will not include a GPIO driver in the initial release, standard Linux GPIO drivers should work with minimal modification.[19]

The community implemented SPI and I²C drivers [20], which will be integrated with the new Linux pinctrl concept in a later version of the kernel. (On Oct. 14 2012, it was already included in the latest raspbian image.) A first compiled version as Linux modules is available to install on the 19/04/2012 Debian image, including 1-wire support[21]. The I²C and SPI driver uses the hardware modules of the microcontroller and interrupts for low CPU usage, the 1-wire support uses bitbanging on the GPIO ports, which results in higher CPU usage.

GordonH[22] wrote a (mostly) Arduino compatible/style WiringPi library in C for controlling the GPIO pins.

A useful tutorial on setting up I²C driver support can be found at Robot Electronics - look for the downloadable document rpi_i2c_setup.doc

SPI

There is one SPI bus brought out to the header: RPi_SPI

I²C

There are two I²C-buses on the Raspberry Pi: One on P1, and one on P5.

Note that there's a bug concerning I²C-clock-stretching, so don't use I²C-devices which use clock-stretching directly with the Raspberry Pi, or use a workaround. Details about this bug can be found at:

- http://www.raspberrypi.org/phpBB3/viewtopic.php?f=44&t=13771
- http://www.advamation.com/knowhow/raspberrypi/rpi-i2c-bug.html