C library for Broadcom BCM 2835 as used in Raspberry Pi

This is a C library for Raspberry Pi (RPi). It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip, allowing access to the GPIO pins on the 26 pin IDE plug on the RPi board so you can control and interface with various external devices.

It provides functions for reading digital inputs and setting digital outputs, using SPI and I2C, and for accessing the system timers. Pin event detection is supported by polling (interrupts are not supported).

It is C++ compatible, and installs as a header file and non-shared library on any Linux-based distro (but clearly is no use except on Raspberry Pi or another board with BCM 2835).

The version of the package that this documentation refers to can be downloaded from http://www.airspayce.com/mikem/bcm2835/bcm2835-1.36.tar.gz You can find the latest version at http://www.airspayce.com/mikem/bcm2835

Several example programs are provided.

Based on data in http://elinux.org/RPi_Low-level_peripherals and http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf and http://www.scribd.com/doc/101830961/GPIO-Pads-Control2

You can also find online help and discussion at http://groups.google.com/group/bcm2835 Please use that group for all questions and discussions on this topic. Do not contact the author directly, unless it is to discuss commercial licensing. Before asking a question or reporting a bug, please read http://www.catb.org/esr/faqs/smart-questions.html

Tested on debian6-19-04-2012, 2012-07-15-wheezy-raspbian, 2013-07-26-wheezy-raspbian and Occidentalisv01 CAUTION: it has been observed that when detect enables such as **bcm2835_gpio_len()** are used and the pin is pulled LOW it can cause temporary hangs on 2012-07-15-wheezy-raspbian, 2013-07-26-wheezy-raspbian and Occidentalisv01. Reason for this is not yet determined, but we suspect that an interrupt handler is hitting a hard loop on those OSs. If you must use **bcm2835_gpio_len()** and friends, make sure you disable the pins with **bcm2835_gpio_clr_len()** and friends after use.

Installation

This library consists of a single non-shared library and header file, which will be installed in the usual places by make install

```
*   # download the latest version of the library, say bcm2835-1.xx.tar.gz, then:
*   tar zxvf bcm2835-1.xx.tar.gz
*   cd bcm2835-1.xx
*   ./configure
*   make
*   sudo make check
*   sudo make install
*
```

Physical Addresses

The functions **bcm2835_peri_read()**, **bcm2835_peri_write()** and **bcm2835_peri_set_bits()** are low level peripheral register access functions. They are designed to use physical addresses as described in section 1.2.3 ARM physical addresses of the BCM2835 ARM Peripherals manual. Physical addresses range from 0x20000000 to 0x20FFFFFF for peripherals. The bus addresses for peripherals are set up to map onto the peripheral bus address range starting at 0x7E000000. Thus a peripheral advertised in the manual at bus address 0x7Ennnnnn is available at physical address 0x20nnnnnn.

The base address of the various peripheral registers are available with the following externals: bcm2835_gpio bcm2835_pwm bcm2835_clk bcm2835_pads bcm2835_spio0 bcm2835_st bcm2835_bsc0 bcm2835_bsc1

Pin Numbering

The GPIO pin numbering as used by RPi is different to and inconsistent with the underlying BCM 2835 chip pin numbering. http://elinux.org/RPi_BCM2835_GPIOs

RPi has a 26 pin IDE header that provides access to some of the GPIO pins on the BCM 2835, as well as power and ground pins. Not all GPIO pins on the BCM 2835 are available on the IDE header.

RPi Version 2 also has a P5 connector with 4 GPIO pins, 5V, 3.3V and Gnd.

The functions in this library are designed to be passed the BCM 2835 GPIO pin number and *not* the RPi pin number. There are symbolic definitions for each of the available pins that you should use for convenience. See **RPiGPIOPin**.

SPI Pins

The bcm2835_spi_* functions allow you to control the BCM 2835 SPI0 interface, allowing you to send and received data by SPI (Serial Peripheral Interface). For more information about SPI, see http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

When **bcm2835_spi_begin()** is called it changes the bahaviour of the SPI interface pins from their default GPIO behaviour in order to support SPI. While SPI is in use, you will not be able to control the state of the SPI pins through the usual bcm2835_spi_gpio_write(). When **bcm2835_spi_end()** is called, the SPI pins will all revert to inputs, and can then be configured and controled with the usual bcm2835_gpio_* calls.

The Raspberry Pi GPIO pins used for SPI are:

- P1-19 (MOSI)
- P1-21 (MISO)
- P1-23 (CLK)
- P1-24 (CE0)
- P1-26 (CE1)

I2C Pins

The bcm2835_i2c_* functions allow you to control the BCM 2835 BSC interface, allowing you to send and received data by I2C ("eye-squared cee"; generically referred to as "two-wire interface") . For more

information about I?C, see http://en.wikipedia.org/wiki/I%C2%B2C

The Raspberry Pi V2 GPIO pins used for I2C are:

- P1-03 (SDA)
- P1-05 (SLC)

PWM

The BCM2835 supports hardware PWM on a limited subset of GPIO pins. This bcm2835 library provides functions for configuring and controlling PWM output on these pins.

The BCM2835 contains 2 independent PWM channels (0 and 1), each of which be connnected to a limited subset of GPIO pins. The following GPIO pins may be connected to the following PWM channels (from section 9.5):

```
*  GPIO PIN     RPi pin   PWM Channel     ALT FUN
*      12                     0             0
*      13                     1             0
*      18          1-12       0             5
*      19                     1             5
*      40                     0             0
*      41                     1             0
*      45                     1             0
*      52                     0             1
*      53                     1             1
*
```

In order for a GPIO pin to emit output from its PWM channel, it must be set to the Alt Function given above. Note carefully that current versions of the Raspberry Pi only expose one of these pins (GPIO 18 = RPi Pin 1-12) on the IO headers, and therefore this is the only IO pin on the RPi that can be used for PWM. Further it must be set to ALT FUN 5 to get PWM output.

Both PWM channels are driven by the same PWM clock, whose clock dvider can be varied using **bcm2835_pwm_set_clock()**. Each channel can be separately enabled with **bcm2835_pwm_set_mode()**. The average output of the PWM channel is determined by the ratio of DATA/RANGE for that channel. Use **bcm2835_pwm_set_range()** to set the range and **bcm2835_pwm_set_data()** to set the data in that ratio

Each PWM channel can run in either Balanced or Mark-Space mode. In Balanced mode, the hardware sends a combination of clock pulses that results in an overall DATA pulses per RANGE pulses. In Mark-Space mode, the hardware sets the output HIGH for DATA clock pulses wide, followed by LOW for RANGE-DATA clock pulses.

The PWM clock can be set to control the PWM pulse widths. The PWM clock is derived from a 19.2MHz clock. You can set any divider, but some common ones are provided by the BCM2835_PWM_CLOCK_DIVIDER_* values of bcm2835PWMClockDivider.

For example, say you wanted to drive a DC motor with PWM at about 1kHz, and control the speed in 1/1024 increments from 0/1024 (stopped) through to 1024/1024 (full on). In that case you might set the clock divider to be 16, and the RANGE to 1024. The pulse repetition frequency will be 1.2MHz/1024 = 1171.875Hz.

Real Time performance constraints

The bcm2835 is a library for user programs (i.e. they run in 'userland'). Such programs are not part of the kernel and are usually subject to paging and swapping by the kernel while it does other things besides running your program. This means that you should not expect to get real-time performance or real-time timing constraints from such programs. In particular, there is no guarantee that the **bcm2835_delay()** and **bcm2835_delayMicroseconds()** will return after exactly the time requested. In fact, depending on other activity on the host, IO etc, you might get significantly longer delay times than the one you asked for. So please dont expect to get exactly the time delay you request.

Arjan reports that you can prevent swapping on Linux with the following code fragment:

```
*    struct sched_param sp;
*    memset(&sp, 0, sizeof(sp));
*    sp.sched_priority = sched_get_priority_max(SCHED_FIFO);
*    sched_setscheduler(0, SCHED_FIFO, &sp);
*    mlockall(MCL_CURRENT | MCL_FUTURE);
*
```

Bindings to other languages

mikem has made Perl bindings available at CPAN: http://search.cpan.org/~mikem/Device-BCM2835-1.9/lib/Device/BCM2835.pm Matthew Baker has kindly made Python bindings available at: https://github.com/mubeta06/py-libbcm2835 Gary Marks has created a Serial Peripheral Interface (SPI) command-line utility for Raspberry Pi, based on the bcm2835 library. The utility, spincl, is licensed under Open Source GNU GPLv3 by iP Solutions (http://ipsolutionscorp.com), as a free download with source included: http://ipsolutionscorp.com/raspberry-pi-spi-utility/

Open Source Licensing GPL V2

This is the appropriate option if you want to share the source code of your application with everyone you distribute it to, and you also want to give them the right to share who uses it. If you wish to use this software under Open Source Licensing, you must contribute all your source code to the open source community in accordance with the GPL Version 2 when your application is distributed. See http://www.gnu.org/copyleft/gpl.html and COPYING

Acknowledgements

Some of this code has been inspired by Dom and Gert. The I2C code has been inspired by Alan Barr.

Revision History

Version

1.0 Initial release

1.1 Minor bug fixes

1.2 Added support for SPI

1.3 Added **bcm2835_spi_transfern()**

1.4 Fixed a problem that prevented SPI CE1 being used. Reported by David Robinson.

1.5 Added **bcm2835_close()** to deinit the library. Suggested by C?sar Ortiz

1.6 Document testing on 2012-07-15-wheezy-raspbian and Occidentalisv01 Functions **bcm2835_gpio_ren()**, **bcm2835_gpio_fen()**, **bcm2835_gpio_hen() bcm2835_gpio_len()**, **bcm2835_gpio_aren()** and **bcm2835_gpio_afen()** now changes only the pin specified. Other pins that were already previously enabled stay enabled. Added **bcm2835_gpio_clr_ren()**, **bcm2835_gpio_clr_fen()**, **bcm2835_gpio_clr_hen() bcm2835_gpio_clr_len()**, **bcm2835_gpio_clr_aren()**, **bcm2835_gpio_clr_afen()** to clear the enable for individual pins, suggested by Andreas Sundstrom.

1.7 Added bcm2835_spi_transfernb to support different buffers for read and write.

1.8 Improvements to read barrier, as suggested by maddin.

1.9 Improvements contributed by mikew: I noticed that it was mallocing memory for the mmaps on /dev/mem. It's not necessary to do that, you can just mmap the file directly, so I've removed the mallocs (and frees). I've also modified delayMicroseconds() to use nanosleep() for long waits, and a busy wait on a high resolution timer for the rest. This is because I've found that calling nanosleep() takes at least 100-200 us. You need to link using '-lrt' using this version. I've added some unsigned casts to the debug prints to silence compiler warnings I was getting, fixed some typos, and changed the value of BCM2835_PAD_HYSTERESIS_ENABLED to 0x08 as per Gert van Loo's doc at http://www.scribd.com/doc/101830961/GPIO-Pads-Control2 Also added a define for the passwrd value that Gert says is needed to change pad control settings.

1.10 Changed the names of the delay functions to **bcm2835_delay()** and **bcm2835_delayMicroseconds()** to prevent collisions with wiringPi. Macros to map delay()-> **bcm2835_delay()** and Macros to map delayMicroseconds()-> **bcm2835_delayMicroseconds()**, which can be disabled by defining BCM2835_NO_DELAY_COMPATIBILITY

1.11 Fixed incorrect link to download file

1.12 New GPIO pin definitions for RPi version 2 (which has a different GPIO mapping)

1.13 New GPIO pin definitions for RPi version 2 plug P5 Hardware base pointers are now available (after initialisation) externally as bcm2835_gpio bcm2835_pwm bcm2835_clk bcm2835_pads bcm2835_spi0.

1.14 Now compiles even if CLOCK_MONOTONIC_RAW is not available, uses CLOCK_MONOTONIC instead. Fixed errors in documentation of SPI divider frequencies based on 250MHz clock. Reported by Ben Simpson.

1.15 Added **bcm2835_close()** to end of examples as suggested by Mark Wolfe.

1.16 Added bcm2835_gpio_set_multi, bcm2835_gpio_clr_multi and bcm2835_gpio_write_multi to allow a mask of pins to be set all at once. Requested by Sebastian Loncar.

1.17 Added bcm2835_gpio_write_mask. Requested by Sebastian Loncar.

1.18 Added bcm2835_i2c_* functions. Changes to bcm2835_delayMicroseconds: now uses the RPi system timer counter, instead of clock_gettime, for improved accuracy. No need to link with -lrt now. Contributed by Arjan van Vught.

1.19 Removed inlines added by previous patch since they don't seem to work everywhere. Reported by olly.

1.20 Patch from Mark Dootson to close /dev/mem after access to the peripherals has been

granted.

1.21 delayMicroseconds is now not susceptible to 32 bit timer overruns. Patch courtesy Jeremy Mortis.

1.22 Fixed incorrect definition of BCM2835_GPFEN0 which broke the ability to set falling edge events. Reported by Mark Dootson.

1.23 Added bcm2835_i2c_set_baudrate and bcm2835_i2c_read_register_rs. Improvements to bcm2835_i2c_read and bcm2835_i2c_write functions to fix ocasional reads not completing. Patched by Mark Dootson.

1.24 Mark Dootson p[atched a problem with his previously submitted code under high load from other processes.

1.25 Updated author and distribution location details to airspayce.com

1.26 Added missing unmapmem for pads in bcm2835_close to prevent a memory leak. Reported by Hartmut Henkel.

1.27 **bcm2835_gpio_set_pad()** no longer needs BCM2835_PAD_PASSWRD: it is now automatically included. Added suport for PWM mode with bcm2835_pwm_* functions.

1.28 Fixed a problem where **bcm2835_spi_writenb()** would have problems with transfers of more than 64 bytes dues to read buffer filling. Patched by Peter Würtz.

1.29 Further fix to SPI from Peter Würtz.

1.30 10 microsecond delays from bcm2835_spi_transfer and bcm2835_spi_transfern for significant performance improvements, Patch by Alan Watson.

1.31 Fix a GCC warning about dummy variable, patched by Alan Watson. Thanks.

1.32 Added option I2C_V1 definition to compile for version 1 RPi. By default I2C code is generated for the V2 RPi which has SDA1 and SCL1 connected. Contributed by Malcolm Wiles based on work by Arvi Govindaraj.

1.33 Added command line utilities i2c and gpio to examples. Contributed by Shahrooz Shahparnia.

1.34 Added **bcm2835_i2c_write_read_rs()** which writes an arbitrary number of bytes, sends a repeat start, and reads from the device. Contributed by Eduardo Steinhorst.

1.35 Fix build errors when compiled under Qt. Also performance improvements with SPI transfers. Contributed by Udo Klaas.

1.36 Make automake's test runner detect that we're skipping tests when not root, the second one makes us skip the test when using fakeroot (as used when building Debian packages). Contributed by Guido Günther.

1.37 Moved confiure.in to configure.ac as receommnded by autoreconf.
Improvements to bcm2835_st_read to account for possible timer overflow, contributed by 'Ed'.
Added definitions for Raspberry Pi B+ J8 header GPIO pins.

Author

Mike McCauley (mikem@airspayce.com) DO NOT CONTACT THE AUTHOR DIRECTLY: USE THE LISTS

Generated by doxygen 1.8.5