Serbus

v1.0.2

Generated by Doxygen 1.8.9.1

Thu Oct 22 2015 12:51:46

Contents

Chapter 1

Main Page

https://github.com/graycatlabs/serbus

Serbus provides basic C APIs for the I2C and SPI serial bus protocols on GNU/Linux based systems, as well as a Python package built on top of them.

It's really just a wrapper for the ioctl commands provided by the standard Linux I2C and SPI drivers, so it should be pretty universal. That said, I've currently only tested it extensively on the BeagleBone Black, so use it at your own risk! (And let me know if it's working for you on another system)

Contributing

Have something to contribute? Great! This project follows the Contributor Covenant Code of Conduct, so be sure to read code_of_conduct.md.

License

Released under the MIT license.

Chapter 2

Contributor Code of Conduct

As contributors and maintainers of this project, and in the interest of fostering an open and welcoming community, we pledge to respect all people who contribute through reporting issues, posting feature requests, updating documentation, submitting pull requests or patches, and other activities.

We are committed to making participation in this project a harassment-free experience for everyone, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, or nationality.

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery
- Personal attacks
- Trolling or insulting/derogatory comments
- Public or private harassment
- Publishing other's private information, such as physical or electronic addresses, without explicit permission
- Other unethical or unprofessional conduct.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct. By adopting this Code of Conduct, project maintainers commit themselves to fairly and consistently applying these principles to every aspect of managing this project. Project maintainers who do not follow or enforce the Code of Conduct may be permanently removed from the project team.

This code of conduct applies both within project spaces and in public spaces when an individual is representing the project or its community.

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue or contacting one or more of the project maintainers.

This Code of Conduct is adapted from the Contributor Covenant, version 1.2.0, available at http://contributor-covenant.org/version/1/2/0/

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

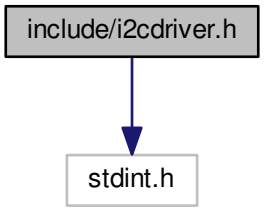
Chapter 4

File Documentation

4.1 include/i2cdriver.h File Reference

A basic driver for controlling Linux I2C interfaces.

`#include <stdint.h>`
Include dependency graph for i2cdriver.h:



Functions

- int I2C_open (uint8_t bus)

  Opens the /dev/i2c-[bus] interface.
- void I2C_close (int i2c_fd)

  Closes the given I2C interface.
- int I2C_enable10BitAddressing (int i2c_fd)

  Enables 10-bit addressing the given I2C interface.
- int I2C_disable10BitAddressing (int i2c_fd)

  Disables 10-bit addressing the given I2C interface.
- int I2C_setSlaveAddress (int i2c_fd, int addr)

  Sets the I2C slave address to communicate with.
- int I2C_read (int i2c_fd, void *rx_buffer, int n_bytes)

  Reads a block from the given I2C interface.
- int I2C_readTransaction (int i2c_fd, uint8_t byte, void *rx_buffer, int n_bytes)

  Writes the given command then reads a block from the given I2C interface.
- int I2C_write (int i2c_fd, void *tx_buffer, int n_bytes)

  Writes a block to the given I2C interface.

4.1.1 Detailed Description

A basic driver for controlling Linux I2C interfaces.

Author

Alex Hiam - alex@graycat.io

Requires an I2C Kernel driver be loaded to expose /dev/i2c-N interfaces which provide the standard Linux I2C ioctls. This driver is really just an ioctl wrapper.

4.1.2 Function Documentation

4.1.2.1 void I2C_close (int *i2c_fd*)

Closes the given I2C interface.

Parameters

i2c_fd	I2C bus file descriptor to close

4.1.2.2 int I2C_disable10BitAddressing (int *i2c_fd*)

Disables 10-bit addressing the given I2C interface.

Parameters

i2c_fd	I2C file descriptor

Returns

Returns 0 if successful, ioctl error code otherwise

4.1.2.3 int I2C_enable10BitAddressing (int *i2c_fd*)

Enables 10-bit addressing the given I2C interface.

Parameters

i2c_fd	I2C file descriptor

Returns

Returns 0 if successful, ioctl error code otherwise

4.1.2.4 int I2C_open (uint8_t *bus*)

Opens the /dev/i2c-[bus] interface.

Parameters

bus	I2C bus number

Returns

Returns the file descriptor for the I2C bus.

4.1.2.5 int I2C_read (int *i2c_fd,* void * *rx_buffer,* int *n_bytes*)

Reads a block from the given I2C interface.

Reads `n_bytes` from the current slave address on the given I2C interface. and puts them into the given buffer.

Parameters

i2c_fd	I2C file descriptor
rx_buffer	pointer to an array, already initialized to the required size
n_bytes	the number of bytes to read into rx_buffer

Returns

Returns 0 if successful, file access error code otherwise

4.1.2.6 int I2C_readTransaction (int *i2c_fd,* uint8_t *byte,* void * *rx_buffer,* int *n_bytes*)

Writes the given command then reads a block from the given I2C interface.

Writes the given byte, then immediately reads n_bytes bytes from the current slave address on the given I2C interface. Useful for things like reading register values from memory mapped devices.

Parameters

i2c_fd	I2C file descriptor
byte	the byte to write before reading
rx_buffer	pointer to an array, already initialized to the required size
n_bytes	the number of bytes to read into rx_buffer

Returns

Returns 0 if successful, file access error code otherwise

4.1.2.7 int I2C_setSlaveAddress (int *i2c_fd,* int *addr*)

Sets the I2C slave address to communicate with.

Sets the I2C slave address that's sent with all subsequent I2C transactions on the given I2C bus, until I2C_set↩ SlaveAddress is called again with a new address.

Parameters

i2c_fd	I2C file descriptor
addr	the 7- or 10-bit address of the slave device

Returns

Returns 0 if successful, ioctl error code otherwise

4.1.2.8 int I2C_write (int *i2c_fd,* void * *tx_buffer,* int *n_bytes*)

Writes a block to the given I2C interface.

Writes n_bytes bytes from the given buffer to the current slave address on the given I2C interface.

Parameters

i2c_fd	I2C file descriptor
tx_buffer	pointer to an array containing the words to be transmitted
n_bytes	the number of bytes to write from tx_buffer

Returns

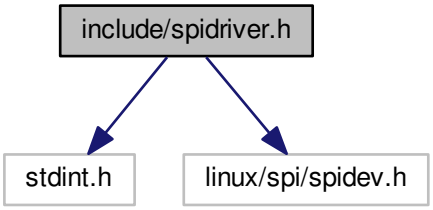
Returns 0 if successful, file access error code otherwise

4.2 include/spidriver.h File Reference

A basic driver for controlling Linux spidev interfaces.

```
#include <stdint.h>
#include <linux/spi/spidev.h>
```

Include dependency graph for spidriver.h:



Enumerations

- enum SPI_bit_order { SPI_MSBFIRST, SPI_LSBFIRST }

Functions

- int SPI_open (uint8_t bus, uint8_t cs)

  Opens the /dev/spidev[bus].[cs] interface.
- void SPI_close (int spidev_fd)

  Closes the given spidev interface.
- int SPI_read (int spidev_fd, void *rx_buffer, int n_words)

  Reads from the given spidev interface.
- int SPI_write (int spidev_fd, void *tx_buffer, int n_words)

  Writes to the given spidev interface.
- int SPI_transfer (int spidev_fd, void *tx_buffer, void *rx_buffer, int n_words)

  Writes to and reads from the given spidev interface simultaneously.
- int SPI_setBitOrder (int spidev_fd, SPI_bit_order bit_order)

  Sets the bit order of the given spidev interface.
- int SPI_setBitsPerWord (int spidev_fd, uint8_t bits_per_word)

  Sets the number of bits per word for the given spidev interface.
- int SPI_getBitsPerWord (int spidev_fd)

Gets the number of bits per word for the given spidev interface.

- int SPI_setMaxFrequency (int spidev_fd, uint32_t frequency)

  Sets the maximum clock frequency for the given spidev interface.
- int SPI_getMaxFrequency (int spidev_fd)

  Gets the maximum clock frequency for the given spidev interface.
- int SPI_setClockMode (int spidev_fd, uint8_t clock_mode)

  Sets the clock mode for the given spidev interface.
- int SPI_getClockMode (int spidev_fd)

  Gets the clock mode for the given spidev interface.
- int SPI_setCSActiveLow (int spidev_fd)

  Sets the given spidev interface's cs signal to be active low.
- int SPI_setCSActiveHigh (int spidev_fd)

  Sets the given spidev interface's cs signal to be active high.
- int SPI_enableCS (int spidev_fd)

  Enables the given spidev interface's cs output.
- int SPI_disableCS (int spidev_fd)

  Disables the given spidev interface's cs output.
- int SPI_enableLoopback (int spidev_fd)

  Puts the given spidev interface in loopback mode.
- int SPI_disableLoopback (int spidev_fd)

  Disables loopback mode for the given spidev interface.
- int SPI_enable3Wire (int spidev_fd)

  Enables 3-wire SPI mode for the given spidev interface.
- int SPI_disable3Wire (int spidev_fd)

  Enables 4-wire SPI mode for the given spidev interface.
- int SPI_setMode (int spidev_fd, uint8_t mode)

  Sets the full SPI mode byte for the given spidev interface.
- int SPI_getMode (int spidev_fd)

  Gets and returns the full SPI mode byte for the given spidev interface.

4.2.1 Detailed Description

A basic driver for controlling Linux spidev interfaces.

Author

Alex Hiam - alex@graycat.io

Requires an SPI Kernel driver be loaded to expose /dev/spidevX.Y interfaces which provide the standard Linux SPI ioctls. This driver is really just an ioctl wrapper.

4.2.2 Enumeration Type Documentation

4.2.2.1 enum SPI_bit_order

Passed to SPI_setBitOrder to specify the bit order to use for subsequent SPI transfers.

Enumerator

SPI_MSBFIRST Most significant bit first.

SPI_LSBFIRST Least significant bit first.

Definition at line 107 of file spidriver.h.

4.2.3 Function Documentation

4.2.3.1 void SPI_close (int *spidev_fd*)

Closes the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

4.2.3.2 int SPI_disable3Wire (int *spidev_fd*)

Enables 4-wire SPI mode for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.3 int SPI_disableCS (int *spidev_fd*)

Disables the given spidev interface's cs output.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.4 int SPI_disableLoopback (int *spidev_fd*)

Disables loopback mode for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.5 int SPI_enable3Wire (int *spidev_fd*)

Enables 3-wire SPI mode for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.6 int SPI_enableCS (int *spidev_fd*)

Enables the given spidev interface's cs output.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.7 int SPI_enableLoopback (int *spidev_fd*)

Puts the given spidev interface in loopback mode.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.8 int SPI_getBitsPerWord (int *spidev_fd*)

Gets the number of bits per word for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns bits per word, or -1 if error

4.2.3.9 int SPI_getClockMode (int *spidev_fd*)

Gets the clock mode for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns Returns the clock mode, or -1 if error

4.2.3.10 int SPI_getMaxFrequency (int *spidev_fd*)

Gets the maximum clock frequency for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns the frequency, or -1 if error

4.2.3.11 int SPI_getMode (int *spidev_fd*)

Gets and returns the full SPI mode byte for the given spidev interface.

Encodes current settings like the clock mode, SC active state, etc., and shouldn't typically need to be called directly.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns SPI mode if successful, or -1 if error

4.2.3.12 int SPI_open (uint8_t *bus*, uint8_t *cs*)

Opens the /dev/spidev[bus].[cs] interface.

Parameters

bus	SPI bus number
cs	chip select number

Returns

Returns the file descriptor for the spidev interface.

4.2.3.13 int SPI_read (int *spidev_fd*, void * *rx_buffer*, int *n_words*)

Reads from the given spidev interface.

Reads `n_words` from the given spidev interface and puts them into the given buffer.

Parameters

spidev_fd	spidev file descriptor
rx_buffer	pointer to an array, already initialized to the required size
n_words	the number of words to read into tx_buffer

Returns

Returns the number of bytes read, or -1 if unable to read from interface

4.2.3.14 int SPI_setBitOrder (int *spidev_fd*, SPI_bit_order *bit_order*)

Sets the bit order of the given spidev interface.

Parameters

spidev_fd	spidev file descriptor
bit_order	one of SPI_MSBFIRST or SPI_LSBFIRST

Returns

Returns 0 if successful, or -1 if error

4.2.3.15 int SPI_setBitsPerWord (int *spidev_fd,* uint8_t *bits_per_word*)

Sets the number of bits per word for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor
bits_per_word	number of bits per word

Returns

Returns 0 if successful, or -1 if error

4.2.3.16 int SPI_setClockMode (int *spidev_fd,* uint8_t *clock_mode*)

Sets the clock mode for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor
clock_mode	one of SPI_MODE_0, SPI_MODE_1, SPI_MODE_2 or SPI_MODE_3

Returns

Returns 0 if successful, -1 if error

4.2.3.17 int SPI_setCSActiveHigh (int *spidev_fd*)

Sets the given spidev interface's cs signal to be active high.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.18 int SPI_setCSActiveLow (int *spidev_fd*)

Sets the given spidev interface's cs signal to be active low.

Parameters

spidev_fd	spidev file descriptor

Returns

Returns 0 if successful, or -1 if error

4.2.3.19 int SPI_setMaxFrequency (int *spidev_fd,* uint32_t *frequency*)

Sets the maximum clock frequency for the given spidev interface.

Parameters

spidev_fd	spidev file descriptor
frequency	maximum clock frequency

Returns

Returns 0 if successful, or -1 if error

4.2.3.20 int SPI_setMode (int *spidev_fd,* uint8_t *mode*)

Sets the full SPI mode byte for the given spidev interface.

Used to set things like the clock mode, SC active state, etc., and shouldn't typically need to be called directly.

Parameters

spidev_fd	spidev file descriptor
mode	SPI mode byte

Returns

Returns 0 if successful, or -1 if error

4.2.3.21 int SPI_transfer (int *spidev_fd,* void * *tx_buffer,* void * *rx_buffer,* int *n_words*)

Writes to and reads from the given spidev interface simultaneously.

Writes n_words from the given tx buffer to the given spidev interface, while simultaneously reading words into the given rx buffer.

Parameters

spidev_fd	spidev file descriptor
tx_buffer	pointer to an array containing the words to be transmitted
rx_buffer	pointer to an array, already initialized to the required size
n_words	the number of words to be transferred

Returns

Returns the number of bytes transferred, or -1 if unable to write interface

4.2.3.22 int SPI_write (int *spidev_fd,* void * *tx_buffer,* int *n_words*)

Writes to the given spidev interface.

Writes `n_words` from the given buffer to the given spidev interface.

Parameters

spidev_fd	spidev file descriptor
tx_buffer	pointer to an array containing the words to be transmitted
n_words	the number of words to be transmitted from tx_buffer

Returns

Returns the number of bytes written, or -1 if unable to write interface

Index