

Quantum Computer Outreach Project

Generated by Doxygen 1.8.13

Contents

1	Todo List	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	LED_GLOBAL Struct Reference	7
4.1.1	Detailed Description	7
5	File Documentation	9
5.1	dspic33e/qcomp-sim-c.X/config.h File Reference	9
5.2	dspic33e/qcomp-sim-c.X/io.c File Reference	9
5.2.1	Detailed Description	10
5.2.2	Function Documentation	10
5.2.2.1	__attribute__()	10
5.2.2.2	flash_all()	10
5.2.2.3	flash_led()	11
5.2.2.4	read_btn()	11
5.2.2.5	read_external_buttons()	11
5.2.2.6	set_external_led()	12
5.2.2.7	set_led()	12
5.2.2.8	set_strobe()	12

5.2.2.9	setup_io()	13
5.2.2.10	start_strobe()	13
5.2.2.11	TLC591x_mode_switch()	13
5.2.2.12	toggle_strobe()	14
5.2.3	Variable Documentation	14
5.2.3.1	led_global	14
5.3	dspic33e/qcomp-sim-c.X/io.h File Reference	14
5.3.1	Detailed Description	15
5.3.2	Function Documentation	15
5.3.2.1	flash_all()	15
5.3.2.2	flash_led()	16
5.3.2.3	read_btn()	16
5.3.2.4	set_external_led()	16
5.3.2.5	set_led()	17
5.3.2.6	set_strobe()	17
5.3.2.7	setup_io()	18
5.3.2.8	start_strobe()	18
5.3.2.9	toggle_strobe()	18
5.4	dspic33e/qcomp-sim-c.X/main.c File Reference	18
5.4.1	Detailed Description	19
5.4.2	Function Documentation	19
5.4.2.1	main()	19
5.5	dspic33e/qcomp-sim-c.X/quantum.c File Reference	19
5.5.1	Detailed Description	20
5.5.2	Function Documentation	20
5.5.2.1	clean_state()	20
5.5.2.2	clean_state_cmplx()	21
5.5.2.3	fix_phase()	21
5.5.2.4	fix_phase_cmplx()	21
5.5.2.5	init_state()	22

5.5.2.6	init_state_cmplx()	22
5.5.2.7	make_ops()	22
5.5.2.8	make_ops_cmplx()	23
5.5.2.9	mat_mul()	23
5.5.2.10	mat_mul_cmplx()	23
5.5.2.11	show_state()	24
5.5.2.12	show_state_cmplx()	24
5.6	dspic33e/qcomp-sim-c.X/quantum.h File Reference	24
5.6.1	Detailed Description	25
5.6.2	Function Documentation	25
5.6.2.1	clean_state()	25
5.6.2.2	clean_state_cmplx()	26
5.6.2.3	fix_phase()	26
5.6.2.4	fix_phase_cmplx()	26
5.6.2.5	init_state()	27
5.6.2.6	init_state_cmplx()	27
5.6.2.7	make_ops()	27
5.6.2.8	make_ops_cmplx()	28
5.6.2.9	mat_mul()	28
5.6.2.10	mat_mul_cmplx()	28
5.6.2.11	show_state()	29
5.6.2.12	show_state_cmplx()	29
5.7	dspic33e/qcomp-sim-c.X/spi.c File Reference	29
5.7.1	Detailed Description	29
5.7.2	Function Documentation	30
5.7.2.1	send_byte_spi_1()	30
5.7.2.2	setup_spi()	30
5.8	dspic33e/qcomp-sim-c.X/spi.h File Reference	31
5.8.1	Detailed Description	31
5.8.2	Function Documentation	31
5.8.2.1	send_byte_spi_1()	31
5.8.2.2	setup_spi()	31
5.9	dspic33e/qcomp-sim-c.X/tests.c File Reference	32
5.9.1	Detailed Description	33
5.10	dspic33e/qcomp-sim-c.X/tests.h File Reference	33
5.10.1	Detailed Description	33
5.11	dspic33e/qcomp-sim-c.X/time.c File Reference	33
5.11.1	Detailed Description	34
5.12	dspic33e/qcomp-sim-c.X/time.h File Reference	34
5.12.1	Detailed Description	34

Chapter 1

Todo List

Global `read_external_buttons ()`

read buttons

Global `set_external_led (int data)`

How long should this be?

Global `TLC591x_mode_switch (int mode)`

mode switcher for LED Driver

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

LED_GLOBAL

Pin mappings Pins for LE and OE on port D OE = RD4 = uC:81 = J1:28 = J10:14 LE = RD3 =
uC:78 = J1:40 = J11:18

Chapter 3

File Index

3.1 File List

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

dspic33e/qcomp-sim-c.X/ config.h	
General config settings #pragma for microcontroller	9
dspic33e/qcomp-sim-c.X/ io.c	
Contains all the functions for reading buttons and writing to LEDs	9
dspic33e/qcomp-sim-c.X/ io.h	
Description: Header file for input output functions	14
dspic33e/qcomp-sim-c.X/ main.c	
The main function	18
dspic33e/qcomp-sim-c.X/ quantum.c	
Description: Contains matrix and vector arithmetic for simulating one qubit	19
dspic33e/qcomp-sim-c.X/ quantum.h	
Description: Header file containing all the matrix arithmetic for simulating a single qubit	24
dspic33e/qcomp-sim-c.X/ spi.c	
Description: Functions for communicating with serial devices	29
dspic33e/qcomp-sim-c.X/ spi.h	
Description: SPI communication functions	31
dspic33e/qcomp-sim-c.X/ tests.c	
Description: Contains all the tests we have performed on the micro- controller	32
dspic33e/qcomp-sim-c.X/ tests.h	
Description: Header file containing all the tests we performed	33
dspic33e/qcomp-sim-c.X/ time.c	
Description: Functions to control the on chip timers	33
dspic33e/qcomp-sim-c.X/ time.h	
Description: Header file containing all the timing functions	34

Chapter 4

Data Structure Documentation

4.1 LED_GLOBAL Struct Reference

pin mappings Pins for LE and OE on port D OE = RD4 = uC:81 = J1:28 = J10:14 LE = RD3 = uC:78 = J1:40 = J11:18

```
#include <io.h>
```

Data Fields

- int [strobe_leds](#)
Bit set the LEDs which are strobing.
- int [strobe_state](#)
Bit zero is the current state (on/off)

4.1.1 Detailed Description

pin mappings Pins for LE and OE on port D OE = RD4 = uC:81 = J1:28 = J10:14 LE = RD3 = uC:78 = J1:40 = J11:18

Pins for SH and CLK_INH on port D SH = RD5 = uC:82 = J1:25 = J10:13 CLK_INH = RD8 = uC:68 = J1:58 = J11:25Global LED strobing state parameter

The documentation for this struct was generated from the following file:

- [dspic33e/qcomp-sim-c.X/io.h](#)

Chapter 5

File Documentation

5.1 dspic33e/qcomp-sim-c.X/config.h File Reference

General config settings #pragma for microcontroller.

This graph shows which files directly or indirectly include this file:

5.2 dspic33e/qcomp-sim-c.X/io.c File Reference

Contains all the functions for reading buttons and writing to LEDs.

```
#include "io.h"  
#include "time.h"  
#include "spi.h"
```

Include dependency graph for io.c:

Macros

- #define **PERIOD** 500000

Functions

- int [setup_io](#) (void)
Set up LEDs and buttons on port D.
- void [__attribute__](#) ((__interrupt__, no_auto_psv))
Interrupt service routine for timer 4.
- void [start_strobe](#) ()
Set LEDs flashing.
- void [stop_strobe](#) ()
Stop LEDs flashing.
- void [set_strobe](#) (int color, int state)
Set an LED strobing.
- void [toggle_strobe](#) (int color)

- *Toggle LED strobe.*
- int `set_led` (int color, int state)
Turn a particular LED on or off.
- int `read_btn` (int btn)
Read the state of a push button.
- void `leds_off` (void)
Turn all the LEDs off.
- void `flash_led` (int color, int number)
Flash LED a number of times.
- void `flash_all` (int number)
Flash all the LEDs a number of times.
- int `set_external_led` (int data)
Turn on an LED via the external display driver.
- int `TLC591x_mode_switch` (int mode)
Switch between normal and special mode.
- int `read_external_buttons` ()
Read external buttons.

Variables

- `LED_GLOBAL led_global` = {0}

5.2.1 Detailed Description

Contains all the functions for reading buttons and writing to LEDs.

Author

J Scott

Date

8/11/18

5.2.2 Function Documentation

5.2.2.1 `__attribute__()`

```
void __attribute__ (
    (__interrupt__, no_auto_psv) )
```

Interrupt service routine for timer 4.

Note

I have no idea what this line means...

5.2.2.2 `flash_all()`

```
void flash_all (
    int number )
```

Flash all the LEDs a number of times.

Parameters

<i>number</i>	
---------------	--

5.2.2.3 flash_led()

```
void flash_led (
    int color,
    int number )
```

Flash LED a number of times.

Flash one LED a number of times.

5.2.2.4 read_btn()

```
int read_btn (
    int btn )
```

Read the state of a push button.

Parameters

<i>btn</i>	
------------	--

Note

How well do you know C

5.2.2.5 read_external_buttons()

```
int read_external_buttons ( )
```

Read external buttons.

The external buttons are interfaced to the microcontroller via a shift register. Data is shifted in a byte at a time using the SPI 3 module. The sequence to read the buttons is as follows:

1) Momentarily bring SH low to latch button data into the shift registers 2) Bring CLK_INH low to enable the clock input on the shift register 3) Start the SPI 3 clock and read data in via the SDI 3 line

The control lines SH and CLK_INH are on port D

Todo read buttons

5.2.2.6 set_external_led()

```
int set_external_led (
    int data )
```

Turn on an LED via the external display driver.

Set an led on the display driver.

On power on, the chip (TLC591x) is in normal mode which means that the clocked bytes sent to the chip set which LEDs are on and which are off (as opposed to setting the current of the LEDs)

To write to the device, use the SPI module to write a byte to the SDI 1 pin on the chip. Then momentarily set the [LE\(ED1\)](#) pin to latch the data onto the output register. Finally, bring the OE(ED2) pin low to enable the current sinking to turn on the LEDs. See the timing diagram on page 17 of the datasheet for details.

[LE\(ED1\)](#) and OE(ED2) will be on Port D

Parameters

<i>data</i>	the byte to send to LED driver
-------------	--------------------------------

Set [LE\(ED1\)](#) pin

Todo How long should this be?

5.2.2.7 set_led()

```
int set_led (
    int color,
    int state )
```

Turn a particular LED on or off.

Parameters

<i>color</i>	
<i>state</i>	

5.2.2.8 set_strobe()

```
void set_strobe (
    int color,
    int state )
```

Set an LED strobing.

Parameters

<i>color</i>	
<i>state</i>	

5.2.2.9 `setup_io()`

```
int setup_io (
    void )
```

Set up LEDs and buttons on port D.

Set the OE pin high

Set OE(ED2) pin

5.2.2.10 `start_strobe()`

```
void start_strobe ( )
```

Set LEDs flashing.

Set LEDs flashing indefinitely.

5.2.2.11 `TLC591x_mode_switch()`

```
int TLC591x_mode_switch (
    int mode )
```

Switch between normal and special mode.

The mode switch for the TLC591x chip is a bit tricky because it involves synchronising the control lines [LE\(ED1\)](#) and OE(ED2) on Port D with the SPI 1 clock. To initiate a mode switch, OE(ED2) must be brought low for one clock cycle, and then the value of [LE\(ED1\)](#) two clock cycles later determines the new mode. See the diagrams on page 19 of the datasheet

So long as the timing is not strict, we can probably implement the mode switch by starting a non-blocking transfer of 1 byte to the device (which starts the SPI 1 clock), followed by clearing OE(ED2) momentarily and then setting the value of [LE\(ED1\)](#) as required. So long as those two things happen before the SPI 1 clock finishes the procedure will probably work. (The reason is the lack of max timing parameters on page 9 for the setup and hold time for ED1 and ED2, which can therefore presumably be longer than one clock cycle.)

Parameters

<i>mode</i>	
-------------	--

Todo mode switcher for LED Driver

5.2.2.12 toggle_strobe()

```
void toggle_strobe (
    int color )
```

Toggle LED strobe.

Parameters

<i>color</i>	
--------------	--

5.2.3 Variable Documentation

5.2.3.1 led_global

```
LED_GLOBAL led_global = {0}
```

Parameters

<i>led_global</i>	Global LED strobing state parameter
-------------------	-------------------------------------

5.3 dspic33e/qcomp-sim-c.X/io.h File Reference

Description: Header file for input output functions.

```
#include "p33EP512MU810.h"
#include "xc.h"
```

Include dependency graph for io.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [LED_GLOBAL](#)

pin mappings Pins for LE and OE on port D OE = RD4 = uC:81 = J1:28 = J10:14 LE = RD3 = uC:78 = J1:40 = J11:18

Macros

- #define [red](#) 0
Locations of LEDs and buttons on Port D.
- #define **amber** 1
- #define **green** 2

- `#define sw1 6`
- `#define sw2 7`
- `#define sw3 13`
- `#define off 0`
- `#define on 1`
- `#define LE 3`
Control for TLC591x chip on Port D.
- `#define OE 4`
- `#define SH 5`
Control lines for SNx4HC165 chip.
- `#define CLK_INH 8`

Functions

- `int setup_io (void)`
Set up LEDs and buttons on port D.
- `int set_led (int color, int state)`
Turn a particular LED on or off.
- `int read_btn (int btn)`
Read the state of a push button.
- `void leds_off (void)`
Turn all the LEDs off.
- `void flash_led (int color, int number)`
Flash one LED a number of times.
- `void flash_all (int number)`
Flash all the LEDs a number of times.
- `void start_strobe ()`
Set LEDs flashing indefinitely.
- `void set_strobe (int color, int state)`
Set an LED strobing.
- `void toggle_strobe (int color)`
Toggle LED strobe.
- `int set_external_led (int data)`
Set an led on the display driver.

5.3.1 Detailed Description

Description: Header file for input output functions.

Include it at the top of any C source file which uses buttons and LEDs. It also defines various constants representing the positions of the buttons and LEDs on port D.

5.3.2 Function Documentation

5.3.2.1 `flash_all()`

```
void flash_all (
    int number )
```

Flash all the LEDs a number of times.

Parameters

<i>number</i>	
---------------	--

5.3.2.2 flash_led()

```
void flash_led (
    int color,
    int number )
```

Flash one LED a number of times.

Parameters

<i>color</i>	
<i>number</i>	

Flash one LED a number of times.

5.3.2.3 read_btn()

```
int read_btn (
    int btn )
```

Read the state of a push button.

Parameters

<i>btn</i>	
------------	--

Note

How well do you know C

5.3.2.4 set_external_led()

```
int set_external_led (
    int data )
```

Set an led on the display driver.

Parameters

<i>data</i>	
-------------	--

Set an led on the display driver.

On power on, the chip (TLC591x) is in normal mode which means that the clocked bytes sent to the chip set which LEDs are on and which are off (as opposed to setting the current of the LEDs)

To write to the device, use the SPI module to write a byte to the SDI 1 pin on the chip. Then momentarily set the [LE\(ED1\)](#) pin to latch the data onto the output register. Finally, bring the OE(ED2) pin low to enable the current sinking to turn on the LEDs. See the timing diagram on page 17 of the datasheet for details.

[LE\(ED1\)](#) and OE(ED2) will be on Port D

Parameters

<i>data</i>	the byte to send to LED driver
-------------	--------------------------------

Set [LE\(ED1\)](#) pin

Todo How long should this be?

5.3.2.5 set_led()

```
int set_led (
    int color,
    int state )
```

Turn a particular LED on or off.

Parameters

<i>color</i>	
<i>state</i>	

5.3.2.6 set_strobe()

```
void set_strobe (
    int color,
    int state )
```

Set an LED strobing.

Parameters

<i>color</i>	
<i>state</i>	

5.3.2.7 setup_io()

```
int setup_io (
    void )
```

Set up LEDs and buttons on port D.

Set the OE pin high

Set OE(ED2) pin

5.3.2.8 start_strobe()

```
void start_strobe ( )
```

Set LEDs flashing indefinitely.

Set LEDs flashing indefinitely.

5.3.2.9 toggle_strobe()

```
void toggle_strobe (
    int color )
```

Toggle LED strobe.

Parameters

<i>color</i>	
--------------	--

5.4 dspic33e/qcomp-sim-c.X/main.c File Reference

The main function.

```
#include "p33EP512MU810.h"
#include "xc.h"
#include "config.h"
#include "time.h"
#include "io.h"
#include "quantum.h"
#include "tests.h"
#include "spi.h"
```

Include dependency graph for main.c:

Functions

- int [main](#) (void)

5.4.1 Detailed Description

The main function.

Author

J R Scott

Date

8/11/18

Description: Contains an example of fixed precision 2x2 matrix multiplication for applying operations to a single qubit. The only operations included are H, X and Z so that everything is real (this can be extended later).

All the functions have now been moved into separate files. [io.h](#) and [io.c](#) contain functions for reading and controlling the buttons and LEDs, and [quantum.h/quantum.c](#) contain the matrix arithmetic for simulating one qubit.

Compile command: make (on linux). But if you want to program the micro- controller too or if you're using windows you're better of downloading and installing MPLAB-X <https://www.microchip.com/mplab/mplab-x-ide>.

Note

You also need the microchip xc16 compilers which are available from <https://www.microchip.com/mplab/compilers>

5.4.2 Function Documentation

5.4.2.1 main()

```
int main (
    void )
```

Loop to cycle through LEDs 0 - 15

5.5 dspic33e/qcomp-sim-c.X/quantum.c File Reference

Description: Contains matrix and vector arithmetic for simulating one qubit.

```
#include "io.h"
#include "quantum.h"
Include dependency graph for quantum.c:
```

Functions

- void **cadd** (Complex a, Complex b, Complex result)
- void **cmul** (Complex a, Complex b, Complex result)
- void **make_ops** (Matrix2 X, Matrix2 Z, Matrix2 H)
Create real? X, Z, H.
- void **make_ops_cmplx** (CMatrix2 X, CMatrix2 Y, CMatrix2 Z, CMatrix2 H)
Create complex X, Y, Z and H.
- void **init_state** (Vector V, State s)
Initialise a real state vector.
- void **init_state_cmplx** (CVector V, State s)
Initialise a complex state vector.
- void **mat_mul** (Matrix2 M, Vector V)
2x2 matrix multiplication
- void **mat_mul_cmplx** (CMatrix2 M, CVector V)
2x2 complex matrix multiplication
- void **fix_phase** (Vector V)
Add a global phase to make first amplitude positive.
- void **fix_phase_cmplx** (CVector V)
Add a global phase to make first complex amplitude positive.
- void **clean_state** (Vector V)
Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$ and $|-\rangle$
- void **clean_state_cmplx** (CVector V)
Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|D\rangle$ and $|A\rangle$
- void **show_state** (Vector V)
Show the qubit state on the LEDs.
- void **show_state_cmplx** (CVector V)
Show the qubit state on the LEDs.

5.5.1 Detailed Description

Description: Contains matrix and vector arithmetic for simulating one qubit.

5.5.2 Function Documentation

5.5.2.1 clean_state()

```
void clean_state (
    Vector V )
```

Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$ and $|-\rangle$

Parameters

V	real vector
---	-------------

5.5.2.2 clean_state_cmplx()

```
void clean_state_cmplx (
    CVector V )
```

Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|D\rangle$ and $|A\rangle$

Parameters

V	complex vector
-----	----------------

5.5.2.3 fix_phase()

```
void fix_phase (
    Vector V )
```

Add a global phase to make first amplitude positive.

Parameters

V	vector
-----	--------

5.5.2.4 fix_phase_cmplx()

```
void fix_phase_cmplx (
    CVector V )
```

Add a global phase to make first complex amplitude positive.

Note

This only works for certain states (zero, one, plus, minus, etc.)

Parameters

V	complex vector
-----	----------------

5.5.2.5 init_state()

```
void init_state (
    Vector V,
    State s )
```

Initialise a real state vector.

Parameters

<i>V</i>	vector
<i>s</i>	state

5.5.2.6 init_state_cmplx()

```
void init_state_cmplx (
    CVector V,
    State s )
```

Initialise a complex state vector.

Parameters

<i>V</i>	complex vector
<i>s</i>	complex state

5.5.2.7 make_ops()

```
void make_ops (
    Matrix2 X,
    Matrix2 Z,
    Matrix2 H )
```

Create real? X, Z, H.

Parameters

<i>X</i>	Pauli x matrix
<i>Z</i>	Pauli z matrix
<i>H</i>	Hadamard matrix

5.5.2.8 make_ops_cmplx()

```
void make_ops_cmplx (
    CMatrix2 X,
    CMatrix2 Y,
    CMatrix2 Z,
    CMatrix2 H )
```

Create complex X, Y, Z and H.

Parameters

<i>X</i>	Pauli X c-Matrix
<i>Z</i>	Pauli Z c-matrix
<i>H</i>	Hadamard c-matrix
<i>Y</i>	Pauli Y c-matrix

5.5.2.9 mat_mul()

```
void mat_mul (
    Matrix2 M,
    Vector V )
```

2x2 matrix multiplication

Parameters

<i>M</i>	real matrix
<i>V</i>	real vector

5.5.2.10 mat_mul_cmplx()

```
void mat_mul_cmplx (
    CMatrix2 M,
    CVector V )
```

2x2 complex matrix multiplication

Parameters

<i>M</i>	complex matrix
<i>V</i>	complex vector

5.5.2.11 show_state()

```
void show_state (
    Vector V )
```

Show the qubit state on the LEDs.

Parameters

V	real vector
---	-------------

5.5.2.12 show_state_cmplx()

```
void show_state_cmplx (
    CVector V )
```

Show the qubit state on the LEDs.

Parameters

V	complex vector
---	----------------

5.6 dspic33e/qcomp-sim-c.X/quantum.h File Reference

Description: Header file containing all the matrix arithmetic for simulating a single qubit.

```
#include "p33EP512MU810.h"
#include "xc.h"
```

Include dependency graph for quantum.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef signed _Fract [Q15](#)
Basic fractional time.
- typedef [Q15 Complex](#)[2]
Complex type.
- typedef [Q15 Matrix4](#)[4][4]
Matrix4 type.
- typedef [Q15 CMatrix4](#)[4][4][2]
- typedef [Q15 Matrix2](#)[2][2]
Matrix2 type.
- typedef [Q15 CMatrix2](#)[2][2][2]
- typedef [Q15 Vector](#)[2]
Vector type.
- typedef [Q15 CVector](#)[2][2]

Enumerations

- enum [State](#) {
ZERO, ONE, PLUS, MINUS,
iPLUS, iMINUS }

Basis states.

Functions

- void [make_ops](#) ([Matrix2](#) X, [Matrix2](#) Z, [Matrix2](#) H)
Create real? X, Z, H.
- void [make_ops_cmplx](#) ([CMatrix2](#) X, [CMatrix2](#) Y, [CMatrix2](#) Z, [CMatrix2](#) H)
Create complex X, Y, Z and H.
- void [init_state](#) ([Vector](#) V, [State](#) s)
Initialise a real state vector.
- void [init_state_cmplx](#) ([CVector](#) V, [State](#) s)
Initialise a complex state vector.
- void [mat_mul](#) ([Matrix2](#) M, [Vector](#) V)
2x2 matrix multiplication
- void [mat_mul_cmplx](#) ([CMatrix2](#) M, [CVector](#) V)
2x2 complex matrix multiplication
- void [fix_phase](#) ([Vector](#) V)
Add a global phase to make first amplitude positive.
- void [fix_phase_cmplx](#) ([CVector](#) V)
Add a global phase to make first complex amplitude positive.
- void [clean_state](#) ([Vector](#) V)
Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$ and $|-\rangle$
- void [clean_state_cmplx](#) ([CVector](#) V)
Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|D\rangle$ and $|A\rangle$
- void [show_state](#) ([Vector](#) V)
Show the qubit state on the LEDs.
- void [show_state_cmplx](#) ([CVector](#) V)
Show the qubit state on the LEDs.

5.6.1 Detailed Description

Description: Header file containing all the matrix arithmetic for simulating a single qubit.

5.6.2 Function Documentation

5.6.2.1 [clean_state\(\)](#)

```
void clean_state (
    Vector V )
```

Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$ and $|-\rangle$

Parameters

V	real vector
-----	-------------

5.6.2.2 clean_state_cmplx()

```
void clean_state_cmplx (
    CVector V )
```

Clean the state: return the closest state out of $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|D\rangle$ and $|A\rangle$

Parameters

V	complex vector
-----	----------------

5.6.2.3 fix_phase()

```
void fix_phase (
    Vector V )
```

Add a global phase to make first amplitude positive.

Parameters

V	vector
-----	--------

5.6.2.4 fix_phase_cmplx()

```
void fix_phase_cmplx (
    CVector V )
```

Add a global phase to make first complex amplitude positive.

Note

This only works for certain states (zero, one, plus, minus, etc.)

Parameters

V	complex vector
-----	----------------

5.6.2.5 init_state()

```
void init_state (
    Vector V,
    State s )
```

Initialise a real state vector.

Parameters

<i>V</i>	vector
<i>s</i>	state

5.6.2.6 init_state_cmplx()

```
void init_state_cmplx (
    CVector V,
    State s )
```

Initialise a complex state vector.

Parameters

<i>V</i>	complex vector
<i>s</i>	complex state

5.6.2.7 make_ops()

```
void make_ops (
    Matrix2 X,
    Matrix2 Z,
    Matrix2 H )
```

Create real? X, Z, H.

Parameters

<i>X</i>	Pauli x matrix
<i>Z</i>	Pauli z matrix
<i>H</i>	Hadamard matrix

5.6.2.8 make_ops_cmplx()

```
void make_ops_cmplx (
    CMatrix2 X,
    CMatrix2 Y,
    CMatrix2 Z,
    CMatrix2 H )
```

Create complex X, Y, Z and H.

Parameters

<i>X</i>	Pauli X c-Matrix
<i>Z</i>	Pauli Z c-matrix
<i>H</i>	Hadamard c-matrix
<i>Y</i>	Pauli Y c-matrix

5.6.2.9 mat_mul()

```
void mat_mul (
    Matrix2 M,
    Vector V )
```

2x2 matrix multiplication

Parameters

<i>M</i>	real matrix
<i>V</i>	real vector

5.6.2.10 mat_mul_cmplx()

```
void mat_mul_cmplx (
    CMatrix2 M,
    CVector V )
```

2x2 complex matrix multiplication

Parameters

<i>M</i>	complex matrix
<i>V</i>	complex vector

5.6.2.11 show_state()

```
void show_state (
    Vector V )
```

Show the qubit state on the LEDs.

Parameters

V	real vector
---	-------------

5.6.2.12 show_state_cmplx()

```
void show_state_cmplx (
    CVector V )
```

Show the qubit state on the LEDs.

Parameters

V	complex vector
---	----------------

5.7 dspic33e/qcomp-sim-c.X/spi.c File Reference

Description: Functions for communicating with serial devices.

```
#include "spi.h"
```

Include dependency graph for spi.c:

Functions

- int [setup_spi](#) (void)
Set up serial peripheral interface.
- int [send_byte_spi_1](#) (int data)
Send a byte to the SPI1 peripheral.
- int [read_byte_spi_3](#) ()
Recieve a byte from the SPI3 peripheral.

5.7.1 Detailed Description

Description: Functions for communicating with serial devices.

5.7.2 Function Documentation

5.7.2.1 send_byte_spi_1()

```
int send_byte_spi_1 (
    int data )
```

Send a byte to the SPI1 peripheral.

Parameters

<i>data</i>	byte to be sent to SPI1
-------------	-------------------------

5.7.2.2 setup_spi()

```
int setup_spi (
    void )
```

Set up serial peripheral interface.

Pin mappings — Pin mappings and codes — J10:41 = J1:91 = uC:70 = RPI74 (PPS code: 0100 1010) J10:44 = J1:93 = uC:9 = RPI52 (PPS code: 0011 0100) J10:47 = J1:101 = uC:34 = RPI42 (PPS code: 0010 1010) J10:43 = J1:95 = uC:72 = RP64 (PPS reg: RPOR0_L; code: 0100 0000) J10:46 = J1:97 = uC:69 = RPI73 (PPS code: 0100 1001) J10:7 = J1:13 = uC:3 = RP85 (PPS reg: RPOR6_L; code: 0101 0101) J10:5 = J1:7 = uC:5 = RP87 (PPS reg: RPOR6_H) J10:55 = J1:117 = uC:10 = RP118 (PPS reg: RPOR13_H)

— Pin mappings for SPI 1 module — SPI 1 Clock Out (SCK1) PPS code: 000110 (0x06) SPI 1 Data Out (SDO1) PPS code: 000101 (0x05) SPI 1 Slave Select PPS code: 000111

— Pin mappings for SPI 3 module — SPI 3 Clock Out (SCK3) PPS code: 100000 (0x20) SPI 3 Data Out (SDO3) PPS code: 011111 (0x1F) SPI 3 Slave Select PPS code: 100001

Configure the SPI 1 pins

< Put SCK1 on J10:43

< Put SDO1 on J10:55

The clock pin also needs to be configured as an input

< Set SCK1 on J10:43 as input

Configure the SPI 3 output pins

< Put SCK3 on J10:7

< Put SDO3 on J10:5

< Put SDI3 on J10:44

< Set SCK3 on J10:7 as input

@note

SPI 1 clock configuration

$SCK1 = F_CY / (\text{Primary Prescaler} * \text{Secondary Prescaler})$

Assuming that $F_CY = 50\text{MHz}$, and the prescalers are 4 and 1, the SPI clock frequency will be 12.5MHz.

5.8 dspic33e/qcomp-sim-c.X/spi.h File Reference

Description: SPI communication functions.

```
#include "p33EP512MU810.h"
#include "xc.h"
```

Include dependency graph for spi.h: This graph shows which files directly or indirectly include this file:

Functions

- int [setup_spi](#) (void)
Set up serial peripheral interface.
- int [send_byte_spi_1](#) (int data)
Send a byte to the SPI1 peripheral.
- int [read_byte_spi_3](#) ()
Recieve a byte from the SPI3 peripheral.

5.8.1 Detailed Description

Description: SPI communication functions.

5.8.2 Function Documentation

5.8.2.1 send_byte_spi_1()

```
int send_byte_spi_1 (
    int data )
```

Send a byte to the SPI1 peripheral.

Parameters

<i>data</i>	byte to be sent to SPI1
-------------	-------------------------

5.8.2.2 setup_spi()

```
int setup_spi (
    void )
```

Set up serial peripheral interface.

Pin mappings — Pin mappings and codes — J10:41 = J1:91 = uC:70 = RPI74 (PPS code: 0100 1010) J10:44 = J1:93 = uC:9 = RPI52 (PPS code: 0011 0100) J10:47 = J1:101 = uC:34 = RPI42 (PPS code: 0010 1010) J10:43 = J1:95 = uC:72 = RP64 (PPS reg: RPOR0_L; code: 0100 0000) J10:46 = J1:97 = uC:69 = RPI73 (PPS code: 0100 1001) J10:7 = J1:13 = uC:3 = RP85 (PPS reg: RPOR6_L; code: 0101 0101) J10:5 = J1:7 = uC:5 = RP87 (PPS reg: RPOR6_H) J10:55 = J1:117 = uC:10 = RP118 (PPS reg: RPOR13_H)

— Pin mappings for SPI 1 module — SPI 1 Clock Out (SCK1) PPS code: 000110 (0x06) SPI 1 Data Out (SDO1) PPS code: 000101 (0x05) SPI 1 Slave Select PPS code: 000111

— Pin mappings for SPI 3 module — SPI 3 Clock Out (SCK3) PPS code: 100000 (0x20) SPI 3 Data Out (SDO3) PPS code: 011111 (0x1F) SPI 3 Slave Select PPS code: 100001

Configure the SPI 1 pins

< Put SCK1 on J10:43

< Put SDO1 on J10:55

The clock pin also needs to be configured as an input

< Set SCK1 on J10:43 as input

Configure the SPI 3 output pins

< Put SCK3 on J10:7

< Put SDO3 on J10:5

< Put SDI3 on J10:44

< Set SCK3 on J10:7 as input

@note

SPI 1 clock configuration

$SCK1 = F_CY / (\text{Primary Prescaler} * \text{Secondary Prescaler})$

Assuming that $F_CY = 50\text{MHz}$, and the prescalers are 4 and 1, the SPI clock frequency will be 12.5MHz.

5.9 dspic33e/qcomp-sim-c.X/tests.c File Reference

Description: Contains all the tests we have performed on the micro- controller.

```
#include "tests.h"
#include "io.h"
#include "quantum.h"
#include "time.h"
Include dependency graph for tests.c:
```

Functions

- void **mat_mul_test** ()
- void **mat_mul_test_cmplx** ()
- void **one_qubit** ()
- void **one_qubit_cmplx** ()
- void **dim_leds** ()
- void **multi_led_strobe** ()

5.9.1 Detailed Description

Description: Contains all the tests we have performed on the micro- controller.

5.10 dspic33e/qcomp-sim-c.X/tests.h File Reference

Description: Header file containing all the tests we performed.

```
#include "p33EP512MU810.h"  
#include "xc.h"
```

Include dependency graph for tests.h: This graph shows which files directly or indirectly include this file:

Functions

- void **mat_mul_test** ()
- void **mat_mul_test_cmplx** ()
- void **one_qubit** ()
- void **one_qubit_cmplx** ()
- void **dim_leds** ()
- void **multi_led_strobe** ()

5.10.1 Detailed Description

Description: Header file containing all the tests we performed.

5.11 dspic33e/qcomp-sim-c.X/time.c File Reference

Description: Functions to control the on chip timers.

```
#include "time.h"
```

Include dependency graph for time.c:

Functions

- void **setup_clock** ()
- void **setup_timer** ()
- void **reset_timer** ()
- void **start_timer** ()
- void **stop_timer** ()
- unsigned long int **read_timer** ()

5.11.1 Detailed Description

Description: Functions to control the on chip timers.

5.12 dspic33e/qcomp-sim-c.X/time.h File Reference

Description: Header file containing all the timing functions.

```
#include "p33EP512MU810.h"  
#include "xc.h"
```

Include dependency graph for time.h: This graph shows which files directly or indirectly include this file:

Functions

- void **setup_clock** ()
- void **setup_timer** ()
- void **reset_timer** ()
- void **start_timer** ()
- void **stop_timer** ()
- unsigned long int **read_timer** ()

5.12.1 Detailed Description

Description: Header file containing all the timing functions.

Index

__attribute__
io.c, 10

clean_state
quantum.c, 20
quantum.h, 25
clean_state_cmplx
quantum.c, 21
quantum.h, 26

dspic33e/qcomp-sim-c.X/config.h, 9
dspic33e/qcomp-sim-c.X/io.c, 9
dspic33e/qcomp-sim-c.X/io.h, 14
dspic33e/qcomp-sim-c.X/main.c, 18
dspic33e/qcomp-sim-c.X/quantum.c, 19
dspic33e/qcomp-sim-c.X/quantum.h, 24
dspic33e/qcomp-sim-c.X/spi.c, 29
dspic33e/qcomp-sim-c.X/spi.h, 31
dspic33e/qcomp-sim-c.X/tests.c, 32
dspic33e/qcomp-sim-c.X/tests.h, 33
dspic33e/qcomp-sim-c.X/time.c, 33
dspic33e/qcomp-sim-c.X/time.h, 34

fix_phase
quantum.c, 21
quantum.h, 26

fix_phase_cmplx
quantum.c, 21
quantum.h, 26

flash_all
io.c, 10
io.h, 15

flash_led
io.c, 11
io.h, 16

init_state
quantum.c, 21
quantum.h, 27

init_state_cmplx
quantum.c, 22
quantum.h, 27

io.c
__attribute__, 10
flash_all, 10
flash_led, 11
led_global, 14
read_btn, 11
read_external_buttons, 11
set_external_led, 11

set_led, 12
set_strobe, 12
setup_io, 13
start_strobe, 13
TLC591x_mode_switch, 13
toggle_strobe, 14

io.h
flash_all, 15
flash_led, 16
read_btn, 16
set_external_led, 16
set_led, 17
set_strobe, 17
setup_io, 18
start_strobe, 18
toggle_strobe, 18

LED_GLOBAL, 7
led_global
io.c, 14

main
main.c, 19

main.c
main, 19

make_ops
quantum.c, 22
quantum.h, 27

make_ops_cmplx
quantum.c, 22
quantum.h, 27

mat_mul
quantum.c, 23
quantum.h, 28

mat_mul_cmplx
quantum.c, 23
quantum.h, 28

quantum.c
clean_state, 20
clean_state_cmplx, 21
fix_phase, 21
fix_phase_cmplx, 21
init_state, 21
init_state_cmplx, 22
make_ops, 22
make_ops_cmplx, 22
mat_mul, 23
mat_mul_cmplx, 23
show_state, 23

- show_state_cmplx, [24](#)
- quantum.h
 - clean_state, [25](#)
 - clean_state_cmplx, [26](#)
 - fix_phase, [26](#)
 - fix_phase_cmplx, [26](#)
 - init_state, [27](#)
 - init_state_cmplx, [27](#)
 - make_ops, [27](#)
 - make_ops_cmplx, [27](#)
 - mat_mul, [28](#)
 - mat_mul_cmplx, [28](#)
 - show_state, [28](#)
 - show_state_cmplx, [29](#)
- read_btn
 - io.c, [11](#)
 - io.h, [16](#)
- read_external_buttons
 - io.c, [11](#)
- send_byte_spi_1
 - spi.c, [30](#)
 - spi.h, [31](#)
- set_external_led
 - io.c, [11](#)
 - io.h, [16](#)
- set_led
 - io.c, [12](#)
 - io.h, [17](#)
- set_strobe
 - io.c, [12](#)
 - io.h, [17](#)
- setup_io
 - io.c, [13](#)
 - io.h, [18](#)
- setup_spi
 - spi.c, [30](#)
 - spi.h, [31](#)
- show_state
 - quantum.c, [23](#)
 - quantum.h, [28](#)
- show_state_cmplx
 - quantum.c, [24](#)
 - quantum.h, [29](#)
- spi.c
 - send_byte_spi_1, [30](#)
 - setup_spi, [30](#)
- spi.h
 - send_byte_spi_1, [31](#)
 - setup_spi, [31](#)
- start_strobe
 - io.c, [13](#)
 - io.h, [18](#)
- TLC591x_mode_switch
 - io.c, [13](#)
- toggle_strobe
 - io.c, [14](#)
- io.h, [18](#)