Arduino Graphic LCD Driver

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:06:44

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

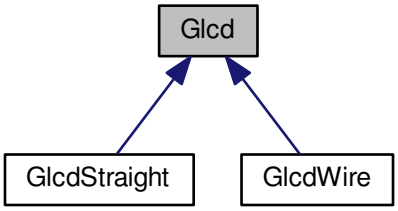
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Glcd Class Reference

`#include <Glcd.h>`

Inheritance diagram for Glcd:



Public Types

- enum Cmd {
  CMD_DISPLAY_ON_OFF = 0x3e, CMD_DISPLAY_START_LINE = 0xc0, CMD_SET_PAGE = 0xb8, CM↩
  D_SET_ADDRESS = 0x40,
  CMD_DISPLAY_ON_OFF_ON = 0x01 }
- enum Mode { MODE_OFF = 0, MODE_ON = 1 }
- enum Color { COLOR_BLACK = 0x00, COLOR_WHITE = 0xff }
- enum Chip { CHIP_1 = 0, CHIP_2 = 1, CHIP_ALL = 0xff }
- enum Rw { RW_WRITE = 0, RW_READ = 1 }
- enum ScrollDirection { SCROLL_UP = 0, SCROLL_DOWN = 1 }
- enum RegisterSelect { RS_COMMAND = 0, RS_DATA = 1 }

Public Member Functions

- virtual void init (Mode mode)=0
- virtual void reset ()=0
- bool isReseting (Chip chip)
- bool isOff (Chip chip)
- bool isBusy (Chip chip)
- void screen (unsigned char pattern)
- bool plot (unsigned char x, unsigned char y, Color color)
- bool streak (unsigned char x, unsigned char page, unsigned char streak)
- void scrollTo (Chip chip, unsigned char line)
- void scroll (Chip chip, ScrollDirection direction, unsigned char lines)
- unsigned char status (Chip chip)
- bool getWriteTimeoutFlag ()
- bool getOutOfRangeFlag ()
- bool getReadInAllChipsFlag ()
- bool isOutOfRange (unsigned char x, unsigned char y)
- unsigned char getWidth ()
- unsigned char getHeight ()
- void clear ()

Protected Member Functions

- Glcd ()
- virtual void initIo ()
- virtual bool write (Chip chip, unsigned char b, RegisterSelect rs)=0
- virtual unsigned char read (Chip chip, RegisterSelect rs)=0
- unsigned char readData (Chip chip)
- bool writeData (Chip chip, unsigned char b)
- bool command (Chip chip, unsigned char cmd)
- unsigned char getChipFromPoint (unsigned char x, unsigned char y)
- unsigned char getPageFromPoint (unsigned char x, unsigned char y)
- unsigned char getLineFromPoint (unsigned char x, unsigned char y)
- unsigned char getBitFromPoint (unsigned char x, unsigned char y)
- bool writeDataAt (Chip chip, unsigned char page, unsigned char line, unsigned char byte)
- unsigned char readDataAt (Chip chip, unsigned char page, unsigned char line)
- void setWriteTimeoutFlag ()
- void clrWriteTimeoutFlag ()
- void setOutOfRangeFlag ()
- void clrOutOfRangeFlag ()
- void setReadInAllChipsFlag ()
- void clrReadInAllChipsFlag ()

Protected Attributes

- unsigned char flags
- struct {
  unsigned char scrollTo:6
  } startLine [GLCD_CHIPS]

4.1.1 Detailed Description

Definition at line 36 of file Glcd.h.

4.1.2 Member Enumeration Documentation

4.1.2.1 enum Glcd::Chip

Enumerator

CHIP_1

CHIP_2

CHIP_ALL

Definition at line 79 of file Glcd.h.

4.1.2.2 enum Glcd::Cmd

Command	Binary								Hex
	D7	D6	D5	D4	D3	D2	D1	D0	
Display on/off	0	0	1	1	1	1	1	1/0	3e or 3f
Display start line	1	1	A	A	A	A	A	A	c0 or ff
Set page	1	0	1	1	1	A	A	A	b8 to bf
Set address	0	1	A	A	A	A	A	A	40 to 7f
Status read	B	0	S	R	0	0	0	0	

```
B: 1=Busy, 0=Not busy
S: 1=On, 0=Off
R: 1=Reset
A: Address

x = Don't care
```

Enumerator

CMD_DISPLAY_ON_OFF

CMD_DISPLAY_START_LINE

CMD_SET_PAGE

CMD_SET_ADDRESS

CMD_DISPLAY_ON_OFF_ON

Definition at line 57 of file Glcd.h.

4.1.2.3 enum Glcd::Color

Glcd color.

Enumerator

COLOR_BLACK

COLOR_WHITE

Definition at line 75 of file Glcd.h.

4.1.2.4 enum Glcd::Mode

initialization mode.

Enumerator

MODE_OFF

MODE_ON

Definition at line 68 of file Glcd.h.

4.1.2.5 enum Glcd::RegisterSelect

Rs pin modes.

Enumerator

RS_COMMAND

RS_DATA

Definition at line 100 of file Glcd.h.

4.1.2.6 enum Glcd::Rw

Rw pin modes.

Enumerator

RW_WRITE

RW_READ

Definition at line 86 of file Glcd.h.

4.1.2.7 enum Glcd::ScrollDirection

Direction of the scroll.

Enumerator

SCROLL_UP

SCROLL_DOWN

Definition at line 93 of file Glcd.h.

4.1.3 Constructor & Destructor Documentation

4.1.3.1 Glcd::Glcd ( ) [protected]

Protected constructor.

Definition at line 16 of file Glcd.cpp.

4.1.4 Member Function Documentation

4.1.4.1 void Glcd::clear ( ) [inline]

Clears the display.

Definition at line 267 of file Glcd.h.

4.1.4.2 void Glcd::clrOutOfRangeFlag ( ) [inline], [protected]

Clears the out of range flag.

Definition at line 450 of file Glcd.h.

4.1.4.3 void Glcd::clrReadInAllChipsFlag ( ) [inline], [protected]

Clears the read in all chip flag.

Definition at line 464 of file Glcd.h.

4.1.4.4 void Glcd::clrWriteTimeoutFlag ( ) [inline], [protected]

Clears the write timeout flag.

Definition at line 436 of file Glcd.h.

4.1.4.5 bool Glcd::command (Chip *chip*, unsigned char *cmd*) [inline], [protected]

Sends a command to the glcd.

Parameters

chip	The chip selector.
cmd	The command to be sent.

Returns

Definition at line 350 of file Glcd.h.

4.1.4.6 unsigned char Glcd::getBitFromPoint (unsigned char *x*, unsigned char *y*) [inline], [protected]

Gets the page bit from a point.

Parameters

x	The X position on the screen.
y	The Y position on the screen.

Returns

Definition at line 398 of file Glcd.h.

4.1.4.7 unsigned char Glcd::getChipFromPoint (unsigned char *x*, unsigned char *y*) [inline], [protected]

Gets the chip from a point.

Parameters

x	The X position on the screen.
y	The Y position on the screen.

Returns

Definition at line 361 of file Glcd.h.

4.1.4.8 unsigned char Glcd::getHeight ( ) [inline]

Gets the glcd height.

Returns

Definition at line 260 of file Glcd.h.

4.1.4.9 unsigned char Glcd::getLineFromPoint (unsigned char *x*, unsigned char *y*) [inline], [protected]

Gets the line from a point.

Parameters

x	The X position on the screen.
y	The Y position on the screen.

Returns

Definition at line 386 of file Glcd.h.

4.1.4.10 bool Glcd::getOutOfRangeFlag ( ) [inline]

Gets the out of range flag.

Returns

bool

Definition at line 222 of file Glcd.h.

4.1.4.11 unsigned char Glcd::getPageFromPoint (unsigned char *x*, unsigned char *y*) [inline], [protected]

Gets the page from a point.

Parameters

x	The X position on the screen.
y	The Y position on the screen.

Returns

Definition at line 374 of file Glcd.h.

4.1.4.12 bool Glcd::getReadInAllChipsFlag ( ) [inline]

Gets the read in all chip flag.

Returns

bool

Definition at line 231 of file Glcd.h.

4.1.4.13 unsigned char Glcd::getWidth ( ) [inline]

Gets the glcd width.

Returns

Definition at line 251 of file Glcd.h.

4.1.4.14 bool Glcd::getWriteTimeoutFlag ( ) [inline]

Gets the write timeout flag.

Returns

bool

Definition at line 213 of file Glcd.h.

4.1.4.15 virtual void Glcd::init (Mode *mode*) [pure virtual]

Initializes the glcd.

Parameters

mode	On or Off.

Implemented in GlcdStraight, and GlcdWire.

4.1.4.16 void Glcd::initIo ( ) [protected], [virtual]

Initializes the IO.

Reimplemented in GlcdStraight.

Definition at line 20 of file Glcd.cpp.

4.1.4.17 bool Glcd::isBusy (Chip *chip*) [inline]

Checks if the busy flags in set on the status.

Parameters

status	

Returns

Definition at line 144 of file Glcd.h.

4.1.4.18 bool Glcd::isOff (Chip *chip*) [inline]

Checks if the off flags in set on the status.

Parameters

status	

Returns

Definition at line 134 of file Glcd.h.

4.1.4.19 bool Glcd::isOutOfRange (unsigned char *x*, unsigned char *y*) [inline]

Checks if the given point is out of range.

Parameters

x	The X position on the screen.
y	The Y position on the screen.

Returns

Definition at line 242 of file Glcd.h.

4.1.4.20 bool Glcd::isReseting (Chip *chip*) [inline]

Checks if the reseting flags in set on the status.

Parameters

status	

Returns

Definition at line 124 of file Glcd.h.

4.1.4.21 bool Glcd::plot (unsigned char *x*, unsigned char *y*, Color *color*)

Turns a pixel on or off.

Parameters

x	The x position.
y	The y position.
color	The color.

Returns

bool

Definition at line 34 of file Glcd.cpp.

4.1.4.22 virtual unsigned char Glcd::read (Chip *chip*, RegisterSelect *rs*) [protected], [pure virtual]

Reads a byte from the glcd.

Parameters

chip	The chip selector.
rs	The register select.

Returns

Implemented in GlcdStraight, and GlcdWire.

4.1.4.23 unsigned char Glcd::readData (Chip *chip*) [inline], [protected]

Gets a byte from the glcd.

Parameters

chip	The chip selector.

Returns

Definition at line 328 of file Glcd.h.

4.1.4.24 unsigned char Glcd::readDataAt (Chip *chip*, unsigned char *page*, unsigned char *line*) [protected]

Gets a byte from the glcd.

Parameters

chip	The chip selector.
page	The page selector.
page	The line selector.

Returns

Definition at line 85 of file Glcd.cpp.

4.1.4.25 virtual void Glcd::reset ( ) [pure virtual]

Issues a resert int the glcd module.

Returns

void

Implemented in GlcdStraight, and GlcdWire.

4.1.4.26 void Glcd::screen (unsigned char *pattern*)

Fill all the buffer with the given pattern.

Parameters

Pattern	

Definition at line 23 of file Glcd.cpp.

4.1.4.27 void Glcd::scroll (Chip *chip,* ScrollDirection *direction,* unsigned char *lines*)

Scrolls the glcd.

Parameters

chip	The chip selector.
direction	The scroll direction.
lines	How many lines will scroll.

Returns

void

Definition at line 91 of file Glcd.cpp.

4.1.4.28 void Glcd::scrollTo (Chip *chip,* unsigned char *line*) [inline]

Scrolls the glcd to the given line.

Parameters

The	chip selector
The	line

Returns

bool

Definition at line 183 of file Glcd.h.

4.1.4.29 void Glcd::setOutOfRangeFlag ( ) [inline], [protected]

Sets the out of range flag.

Definition at line 443 of file Glcd.h.

4.1.4.30 void Glcd::setReadInAllChipsFlag ( ) [inline], [protected]

Sets the read in all chip flag.

Definition at line 457 of file Glcd.h.

4.1.4.31 void Glcd::setWriteTimeoutFlag ( ) [inline], [protected]

Sets the write timeout flag.

Definition at line 429 of file Glcd.h.

4.1.4.32 unsigned char Glcd::status (Chip *chip*) [inline]

Gets the status of the glcd.

Parameters

chip	The chip selector.

Returns

Byte representing the status info.

Definition at line 204 of file Glcd.h.

4.1.4.33 bool Glcd::streak (unsigned char *x*, unsigned char *page*, unsigned char *streak*)

Writes a entire byte at the page and line.

Parameters

line	
page	
chunk	

Returns

Definition at line 61 of file Glcd.cpp.

4.1.4.34 virtual bool Glcd::write (Chip *chip*, unsigned char *b*, RegisterSelect *rs*) [protected], [pure virtual]

Writes a byte into the glcd.

Parameters

chip	The chip selector.
b	The byte to be written.
rs	The register select.

Returns

Implemented in GlcdStraight, and GlcdWire.

4.1.4.35 bool Glcd::writeData (Chip *chip*, unsigned char *b*) [inline], [protected]

Sends data to the glcd.

Parameters

chip	The chip selector.
b	The data to be sent.

Returns

Definition at line 339 of file Glcd.h.

4.1.4.36 bool Glcd::writeDataAt (Chip *chip*, unsigned char *page*, unsigned char *line*, unsigned char *byte*) [protected]

Sends data to the glcd by the given chip, page and line.

Parameters

chip	The chip selector.
page	The page selector.
page	The line selector.
data	The data to be sent.

Returns

Definition at line 79 of file Glcd.cpp.

4.1.5 Member Data Documentation

4.1.5.1 unsigned char Glcd::flags [protected]

```
0b00000000
  ||||||||_ Timeout on write operation
  |||||||__ Plot out of range
  ||||||___ Read all chip at same time
  |||||_____ Unused
  ||||______ Unused
  |||_______ Unused
  ||________ Unused
  |_________ Unused
```

Definition at line 286 of file Glcd.h.

4.1.5.2 unsigned char Glcd::scrollTo

Definition at line 289 of file Glcd.h.

4.1.5.3 struct { ... } Glcd::startLine[GLCD_CHIPS] [protected]

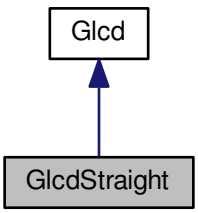
The documentation for this class was generated from the following files:

- Glcd.h
- Glcd.cpp

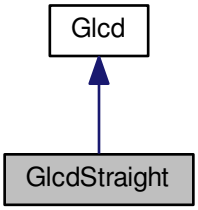
4.2 GlcdStraight Class Reference

```
#include <GlcdStraight.h>
```

Inheritance diagram for GlcdStraight:



Collaboration diagram for GlcdStraight:



Public Member Functions

- GlcdStraight ()
- void init (Mode mode)
- void reset ()

Protected Member Functions

- void initIo ()
- bool write (Chip chip, unsigned char b, RegisterSelect rs)
- unsigned char read (Chip chip, RegisterSelect rs)
- void switchRegisterSelectTo (RegisterSelect rs)
- void switchRegisterSelectToData ()
- void switchRegisterSelectToCommand ()
- void switchChipTo (Chip chip)
- void disableChips ()
- void switchRwToWrite ()
- void switchRwToRead ()
- void writeToBus (unsigned char b)
- unsigned char readFromBus ()
- void busOutputDirection ()
- void busInputDirection ()
- void setEnablePin ()
- void clrEnablePin ()

Additional Inherited Members

4.2.1 Detailed Description

Definition at line 82 of file GlcdStraight.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 GlcdStraight::GlcdStraight ( )

Definition at line 17 of file GlcdStraight.cpp.

4.2.3 Member Function Documentation

4.2.3.1 void GlcdStraight::busInputDirection ( ) [inline], [protected]

Sets all bus pin as input.

Definition at line 203 of file GlcdStraight.h.

4.2.3.2 void GlcdStraight::busOutputDirection ( ) [inline], [protected]

Sets all bus pin as output.

Definition at line 195 of file GlcdStraight.h.

4.2.3.3 void GlcdStraight::clrEnablePin ( ) [inline], [protected]

Clears the enable pin.

Definition at line 218 of file GlcdStraight.h.

4.2.3.4 void GlcdStraight::disableChips ( ) [inline], [protected]

Disable the chips.

Definition at line 154 of file GlcdStraight.h.

4.2.3.5 void GlcdStraight::init (Mode *mode*) [virtual]

Initializes the glcd.

Parameters

mode	On or Off.

Implements Glcd.

Definition at line 20 of file GlcdStraight.cpp.

4.2.3.6 void GlcdStraight::initIo ( ) [protected], [virtual]

Initializes the IO.

Reimplemented from Glcd.

Definition at line 38 of file GlcdStraight.cpp.

4.2.3.7 unsigned char GlcdStraight::read (Chip *chip*, RegisterSelect *rs*) [protected], [virtual]

Reads a byte from the glcd.

Parameters

chip	The chip selector.
rs	The register select.

Returns

Implements Glcd.

Definition at line 108 of file GlcdStraight.cpp.

4.2.3.8 unsigned char GlcdStraight::readFromBus ( ) [inline], [protected]

Reads a byte from bus.

Returns

Definition at line 188 of file GlcdStraight.h.

4.2.3.9 void GlcdStraight::reset ( ) [virtual]

Issues a resert int the glcd module.

Returns

void

Implements Glcd.

Definition at line 50 of file GlcdStraight.cpp.

4.2.3.10 void GlcdStraight::setEnablePin ( ) [inline], [protected]

Sets the enable pin.

Definition at line 211 of file GlcdStraight.h.

4.2.3.11 void GlcdStraight::switchChipTo (Chip *chip*) [protected]

Selects the given chip.

Definition at line 67 of file GlcdStraight.cpp.

4.2.3.12 void GlcdStraight::switchRegisterSelectTo (RegisterSelect *rs*) [protected]

Switch the register select pin to the given mode.

Definition at line 59 of file GlcdStraight.cpp.

4.2.3.13 void GlcdStraight::switchRegisterSelectToCommand ( ) [inline], [protected]

Switch the register select pin to command mode.

Definition at line 142 of file GlcdStraight.h.

4.2.3.14 void GlcdStraight::switchRegisterSelectToData ( ) [inline], [protected]

Switch the register select pin to data mode.

Definition at line 135 of file GlcdStraight.h.

4.2.3.15 void GlcdStraight::switchRwToRead ( ) [inline], [protected]

Switch the rs pin to read.

Definition at line 169 of file GlcdStraight.h.

4.2.3.16 void GlcdStraight::switchRwToWrite ( ) [inline], [protected]

Switch the rs pin to write.

Definition at line 162 of file GlcdStraight.h.

4.2.3.17 bool GlcdStraight::write (Chip *chip*, unsigned char *b*, RegisterSelect *rs*) [protected], [virtual]

Writes a byte into the glcd.

Parameters

chip	The chip selector.
b	The byte to be written.
rs	The register select.

Returns

Implements Glcd.

Definition at line 82 of file GlcdStraight.cpp.

4.2.3.18 void GlcdStraight::writeToBus (unsigned char *b*) [inline],[protected]

Writes a byte to bus.

Parameters

byte	

Definition at line 178 of file GlcdStraight.h.

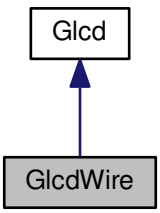
The documentation for this class was generated from the following files:

- GlcdStraight.h
- GlcdStraight.cpp

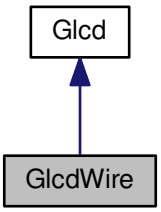
4.3 GlcdWire Class Reference

`#include <GlcdWire.h>`

Inheritance diagram for GlcdWire:



Collaboration diagram for GlcdWire:



Public Member Functions

- GlcdWire (unsigned char device)
- void init (Mode mode)
- void reset ()
- bool write (Chip chip, unsigned char b, RegisterSelect rs)
- unsigned char read (Chip chip, RegisterSelect rs)

Protected Attributes

- unsigned char device
- struct {
  unsigned char page
  unsigned char line
  } chipInfo [GLCD_CHIPS]

Private Member Functions

- unsigned char makeHeader (Chip chip, RegisterSelect rs, Rw rw)

Additional Inherited Members

4.3.1 Detailed Description

Arduino - Glcd driver.

GlcdWire.h

The header file for glcd driver, wicth implements the driver base using i2c with a PIC microcontroller.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 17 of file GlcdWire.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 GlcdWire::GlcdWire (unsigned char *device*)

Public constructor.

Parameters

address	The interface address.

Definition at line 19 of file GlcdWire.cpp.

4.3.3 Member Function Documentation

4.3.3.1 void GlcdWire::init (Mode *mode*) [virtual]

Initializes the glcd.

Parameters

mode	On or Off.

Implements Glcd.

Definition at line 23 of file GlcdWire.cpp.

4.3.3.2 unsigned char GlcdWire::makeHeader (Chip *chip*, RegisterSelect *rs*, Rw *rw*) [private]

Makes the header of the pic i2c communication.

```
header: 0b00000000
          ||||||||_ Chip b0 \: the chip, 11 means all chips
          |||||||__ Chip b1 /
          ||||||___ Register Select: 1 means data, 0 means command
          |||||____ Read/Write: 1 means read, 0 means write
          ||||_____ Unused
          |||______ Unused
          ||_______ Unused
          |________ Unused
```

Parameters

chip	The chip.
rs	The register select.
rw	Read/Write.

Returns

unsigned char

4.3.3.3 unsigned char GlcdWire::read (Chip *chip*, RegisterSelect *rs*) [virtual]

Reads a byte from the glcd.

Parameters

chip	The chip selector.
rs	The register select.

Returns

Implements Glcd.

Definition at line 73 of file GlcdWire.cpp.

4.3.3.4 void GlcdWire::reset ( ) [virtual]

Issues a reset int the glcd module.

Returns

void

Implements Glcd.

Definition at line 31 of file GlcdWire.cpp.

4.3.3.5 bool GlcdWire::write (Chip *chip*, unsigned char *b*, RegisterSelect *rs*) [virtual]

Writes a byte into the glcd.

BLOCKING!

Parameters

chip	The chip selector.
b	The byte to be written.
rs	The register select.

Returns

Implements Glcd.

Definition at line 34 of file GlcdWire.cpp.

4.3.4 Member Data Documentation

4.3.4.1 struct { ... } GlcdWire::chipInfo[GLCD_CHIPS] [protected]

4.3.4.2 unsigned char GlcdWire::device [protected]

Definition at line 20 of file GlcdWire.h.

4.3.4.3 unsigned char GlcdWire::line

Definition at line 24 of file GlcdWire.h.

4.3.4.4 unsigned char GlcdWire::page

Definition at line 23 of file GlcdWire.h.

The documentation for this class was generated from the following files:

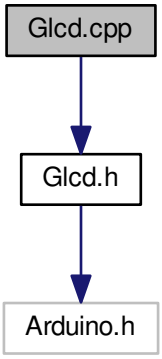
- GlcdWire.h
- GlcdWire.cpp

5 File Documentation

5.1 Glcd.cpp File Reference

```
#include "Glcd.h"
```

Include dependency graph for Glcd.cpp:



Macros

- #define __ARDUINO_DRIVER_GLCD_CPP__ 1

5.1.1 Macro Definition Documentation

5.1.1.1 #define __ARDUINO_DRIVER_GLCD_CPP__ 1

Arduino - Glcd driver.

Glcd.c

The glcd driver functions

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file Glcd.cpp.

5.2 Glcd.cpp

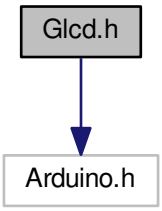
```
00001
00011 #ifndef __ARDUINO_DRIVER_GLCD_CPP__
00012 #define __ARDUINO_DRIVER_GLCD_CPP__ 1
00013
00014 #include "Glcd.h"
00015
00016 Glcd::Glcd() {
00017     flags = 0x00;
00018 }
00019
00020 void Glcd::initIo() {
00021 }
00022
00023 void Glcd::screen(unsigned char pattern) {
00024
00025     unsigned char chip, page, line;
00026
00027     for (page = 0; page < GLCD_CHIP_PAGES; page++) {
00028         for (line = 0; line < GLCD_PAGE_LINES; line++) {
00029             writeDataAt(Glcd::CHIP_ALL, page, line, pattern);
00030         }
```

```
00031     }
00032 }
00033
00034 bool Glcd::plot(unsigned char x, unsigned char y, Color color) {
00035
00036     unsigned char b;
00037
00038     unsigned char chip, page, line;
00039
00040     if (isOutOfRange(x, y)) {
00041
00042         setOutOfRangeFlag();
00043         return 0;
00044     }
00045
00046     chip = getChipFromPoint(x, y);
00047     page = getPageFromPoint(x, y);
00048     line = getLineFromPoint(x, y);
00049
00050     b = readDataAt((Chip) chip, page, line);
00051
00052     if (color) {
00053         bitSet(b, getBitFromPoint(x, y));
00054     } else {
00055         bitClear(b, getBitFromPoint(x, y));
00056     }
00057
00058     return writeDataAt((Chip) chip, page, line, b);
00059 }
00060
00061 bool Glcd::streak(unsigned char x, unsigned char page, unsigned char streak) {
00062
00063     unsigned char chip, line, y;
00064
00065     y = page * 8;
00066
00067     if (isOutOfRange(x, y)) {
00068
00069         setOutOfRangeFlag();
00070         return 0;
00071     }
00072
00073     chip = getChipFromPoint(x, y);
00074     line = getLineFromPoint(x, y);
00075
00076     return writeDataAt((Chip) chip, page, line, streak);
00077 }
00078
00079 bool Glcd::writeDataAt(Chip chip, unsigned char page, unsigned char line, unsigned
      char b) {
00080     command(chip, (unsigned char) (CMD_SET_PAGE | page));
00081     command(chip, (unsigned char) (CMD_SET_ADDRESS | line));
00082     return writeData(chip, b);
00083 }
00084
00085 unsigned char Glcd::readDataAt(Chip chip, unsigned char page, unsigned char line) {
00086     command(chip, (unsigned char) (CMD_SET_PAGE | page));
00087     command(chip, (unsigned char) (CMD_SET_ADDRESS | line));
00088     return readData(chip);
00089 }
00090
00091 void Glcd::scroll(Chip chip, ScrollDirection direction, unsigned char lines)
       {
00092     unsigned char i = 0;
00093     if (direction == SCROLL_DOWN) {
00094         lines = -(lines);
00095     }
00096     if (chip == CHIP_ALL) {
00097         for (i = 0; i < GLCD_CHIPS; i++) {
00098             startLine[i].scrollTo += lines;
00099             scrollTo((Chip) i, startLine[i].scrollTo);
00100         }
00101     } else {
00102         startLine[chip].scrollTo += lines;
00103         scrollTo((Chip) chip, startLine[chip].scrollTo);
00104     }
00105 }
00106
00107 #endif /* __ARDUINO_DRIVER_GLCD_CPP__ */
```

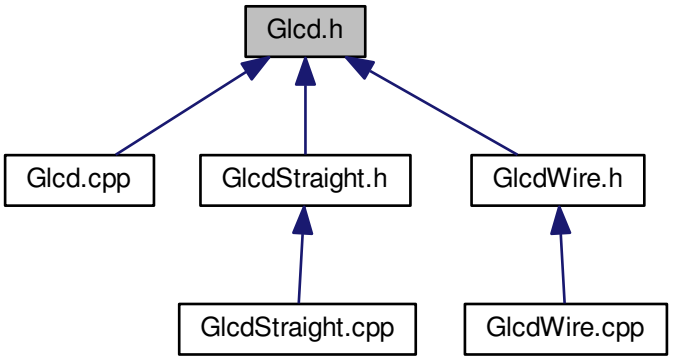
5.3 Glcd.h File Reference

```
#include <Arduino.h>
```

Include dependency graph for Glcd.h:



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



Classes

- class Glcd

Macros

- #define GLCD_CHIP_WIDTH 64
- #define GLCD_CHIP_HEIGHT 64
- #define GLCD_HORIZONTAL_CHIPS 2
- #define GLCD_VERTICAL_CHIPS 1
- #define GLCD_CHIPS (GLCD_HORIZONTAL_CHIPS * GLCD_VERTICAL_CHIPS)
- #define GLCD_WIDTH (GLCD_HORIZONTAL_CHIPS * GLCD_CHIP_WIDTH)
- #define GLCD_HEIGHT (GLCD_VERTICAL_CHIPS * GLCD_CHIP_HEIGHT)
- #define GLCD_CHIP_AREA (GLCD_CHIP_WIDTH * GLCD_CHIP_HEIGHT)
- #define GLCD_AREA (GLCD_CHIP_AREA * GLCD_CHIPS)
- #define GLCD_CHIP_PAGES (GLCD_CHIP_HEIGHT / 8)
- #define GLCD_PAGE_LINES (GLCD_CHIP_WIDTH)

- #define GLCD_STATUS_RESET_BIT 0x10
- #define GLCD_STATUS_OFF_BIT 0x20
- #define GLCD_STATUS_BUSY_BIT 0x80
- #define GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT 0x10
- #define GLCD_FLAGS_PLOT_OUT_OF_RANGE_BIT 0x20
- #define GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT 0x40

5.3.1 Macro Definition Documentation

5.3.1.1 #define GLCD_AREA (GLCD_CHIP_AREA * GLCD_CHIPS)

Definition at line 24 of file Glcd.h.

5.3.1.2 #define GLCD_CHIP_AREA (GLCD_CHIP_WIDTH * GLCD_CHIP_HEIGHT)

Definition at line 23 of file Glcd.h.

5.3.1.3 #define GLCD_CHIP_HEIGHT 64

Definition at line 17 of file Glcd.h.

5.3.1.4 #define GLCD_CHIP_PAGES (GLCD_CHIP_HEIGHT / 8)

Definition at line 25 of file Glcd.h.

5.3.1.5 #define GLCD_CHIP_WIDTH 64

Arduino - Glcd driver.

Glcd.h

The glcd driver functions

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 16 of file Glcd.h.

5.3.1.6 #define GLCD_CHIPS (GLCD_HORIZONTAL_CHIPS * GLCD_VERTICAL_CHIPS)

Definition at line 20 of file Glcd.h.

5.3.1.7 #define GLCD_FLAGS_PLOT_OUT_OF_RANGE_BIT 0x20

Definition at line 33 of file Glcd.h.

5.3.1.8 #define GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT 0x40

Definition at line 34 of file Glcd.h.

5.3.1.9 #define GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT 0x10

Definition at line 32 of file Glcd.h.

5.3.1.10 #define GLCD_HEIGHT (GLCD_VERTICAL_CHIPS * GLCD_CHIP_HEIGHT)

Definition at line 22 of file Glcd.h.

5.3.1.11 #define GLCD_HORIZONTAL_CHIPS 2

Definition at line 18 of file Glcd.h.

5.3.1.12 #define GLCD_PAGE_LINES (GLCD_CHIP_WIDTH)

Definition at line 26 of file Glcd.h.

5.3.1.13 #define GLCD_STATUS_BUSY_BIT 0x80

Definition at line 30 of file Glcd.h.

5.3.1.14 #define GLCD_STATUS_OFF_BIT 0x20

Definition at line 29 of file Glcd.h.

5.3.1.15 #define GLCD_STATUS_RESET_BIT 0x10

Definition at line 28 of file Glcd.h.

5.3.1.16 #define GLCD_VERTICAL_CHIPS 1

Definition at line 19 of file Glcd.h.

5.3.1.17 #define GLCD_WIDTH (GLCD_HORIZONTAL_CHIPS * GLCD_CHIP_WIDTH)

Definition at line 21 of file Glcd.h.

5.4 Glcd.h

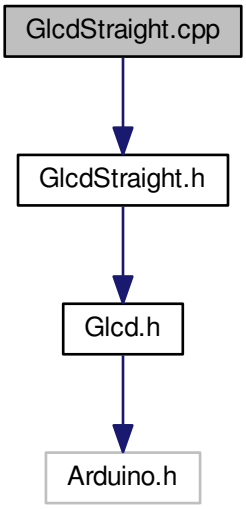
```
00001
00011 #ifndef __ARDUINO_DRIVER_GLCD_H__
00012 #define __ARDUINO_DRIVER_GLCD_H__ 1
00013
00014 #include <Arduino.h>
00015
00016 #define GLCD_CHIP_WIDTH                                 64
00017 #define GLCD_CHIP_HEIGHT                                64
00018 #define GLCD_HORIZONTAL_CHIPS                           2
00019 #define GLCD_VERTICAL_CHIPS                             1
00020 #define GLCD_CHIPS                                      (GLCD_HORIZONTAL_CHIPS * GLCD_VERTICAL_CHIPS)
00021 #define GLCD_WIDTH                                      (GLCD_HORIZONTAL_CHIPS * GLCD_CHIP_WIDTH)
00022 #define GLCD_HEIGHT                                     (GLCD_VERTICAL_CHIPS * GLCD_CHIP_HEIGHT)
00023 #define GLCD_CHIP_AREA                                  (GLCD_CHIP_WIDTH * GLCD_CHIP_HEIGHT)
00024 #define GLCD_AREA                                       (GLCD_CHIP_AREA * GLCD_CHIPS)
00025 #define GLCD_CHIP_PAGES                                 (GLCD_CHIP_HEIGHT / 8)
00026 #define GLCD_PAGE_LINES                                 (GLCD_CHIP_WIDTH)
00027
00028 #define GLCD_STATUS_RESET_BIT                           0x10
00029 #define GLCD_STATUS_OFF_BIT                             0x20
00030 #define GLCD_STATUS_BUSY_BIT                            0x80
00031
00032 #define GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT                0x10
00033 #define GLCD_FLAGS_PLOT_OUT_OF_RANGE_BIT                0x20
00034 #define GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT                0x40
00035
00036 class Glcd {
00037 public:
00038
00057     enum Cmd {
00058         CMD_DISPLAY_ON_OFF = 0x3e,
00059         CMD_DISPLAY_START_LINE = 0xc0,
00060         CMD_SET_PAGE = 0xb8,
00061         CMD_SET_ADDRESS = 0x40,
00062         CMD_DISPLAY_ON_OFF_ON = 0x01
00063     };
00064
00068     enum Mode {
00069         MODE_OFF = 0, MODE_ON = 1
00070     };
00071
00075     enum Color {
00076         COLOR_BLACK = 0x00, COLOR_WHITE = 0xff
00077     };
00078
00079     enum Chip {
00080         CHIP_1 = 0, CHIP_2 = 1, CHIP_ALL = 0xff
00081     };
00082
```

```
00086     enum Rw {
00087         RW_WRITE = 0, RW_READ = 1
00088     };
00089
00093     enum ScrollDirection {
00094         SCROLL_UP = 0, SCROLL_DOWN = 1
00095     };
00096
00100     enum RegisterSelect {
00101         RS_COMMAND = 0, RS_DATA = 1
00102     };
00103
00109     virtual void init(Mode mode) = 0;
00110
00116     virtual void reset() = 0;
00117
00124     bool isReseting(Chip chip) {
00125         return ((status(chip) & GLCD_STATUS_RESET_BIT) != 0);
00126     }
00127
00134     bool isOff(Chip chip) {
00135         return ((status(chip) & GLCD_STATUS_OFF_BIT) != 1);
00136     }
00137
00144     bool isBusy(Chip chip) {
00145         return ((status(chip) & GLCD_STATUS_BUSY_BIT) != 0);
00146     }
00147
00153     void screen(unsigned char pattern);
00154
00163     bool plot(unsigned char x, unsigned char y, Color color);
00164
00173     bool streak(unsigned char x, unsigned char page,
00174             unsigned char streak);
00175
00183     void scrollTo(Chip chip, unsigned char line) {
00184         command(chip, Glcd::CMD_DISPLAY_START_LINE | (line & 0x3f));
00185     }
00186
00195     void scroll(Chip chip, ScrollDirection direction,
00196             unsigned char lines);
00197
00204     unsigned char inline status(Chip chip) {
00205         return read(chip, Glcd::RS_COMMAND);
00206     }
00207
00213     bool inline getWriteTimeoutFlag() {
00214         return ((flags & GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT) != 0);
00215     }
00216
00222     bool inline getOutOfRangeFlag() {
00223         return ((flags & GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT) != 0);
00224     }
00225
00231     bool inline getReadInAllChipsFlag() {
00232         return ((flags & GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT) != 0);
00233     }
00234
00242     bool inline isOutOfRange(unsigned char x, unsigned char y) {
00243         return (x > GLCD_WIDTH || y > GLCD_HEIGHT);
00244     }
00245
00251     unsigned char inline getWidth() {
00252         return GLCD_WIDTH;
00253     }
00254
00260     unsigned char inline getHeight() {
00261         return GLCD_HEIGHT;
00262     }
00263
00267     void inline clear() {
00268         screen(0x00);
00269     }
00270
00271 protected:
00272
00286     unsigned char flags;
00287
00288     struct {
00289         unsigned char scrollTo :6;
00290     } startLine[GLCD_CHIPS];
00291
00295     Glcd();
00296
00300     virtual void initIo();
00301
00310     virtual bool write(Chip chip, unsigned char b,
```

```
00311             RegisterSelect rs) = 0;
00312
00320     virtual unsigned char read(Chip chip, RegisterSelect rs) = 0;
00321
00328     unsigned char inline readData(Chip chip) {
00329         return read(chip, Glcd::RS_DATA);
00330     }
00331
00339     bool inline writeData(Chip chip, unsigned char b) {
00340         return write(chip, b, Glcd::RS_DATA);
00341     }
00342
00350     bool inline command(Chip chip, unsigned char cmd) {
00351         return write(chip, cmd, Glcd::RS_COMMAND);
00352     }
00353
00361     unsigned char inline getChipFromPoint(unsigned char x,
00362             unsigned char y) {
00363         return ((y / GLCD_CHIP_HEIGHT) * GLCD_HORIZONTAL_CHIPS)
00364                 + (x / GLCD_CHIP_WIDTH);
00365     }
00366
00374     unsigned char inline getPageFromPoint(unsigned char x,
00375             unsigned char y) {
00376         return (y % GLCD_CHIP_HEIGHT) / 8;
00377     }
00378
00386     unsigned char inline getLineFromPoint(unsigned char x,
00387             unsigned char y) {
00388         return x % GLCD_CHIP_WIDTH;
00389     }
00390
00398     unsigned char inline getBitFromPoint(unsigned char x,
00399             unsigned char y) {
00400         return y % 8;
00401     }
00402
00412     bool writeDataAt(Chip chip, unsigned char page, unsigned char line,
00413             unsigned char byte);
00414
00423     unsigned char readDataAt(Chip chip, unsigned char page,
00424             unsigned char line);
00425
00429     void inline setWriteTimeoutFlag() {
00430         flags |= GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT;
00431     }
00432
00436     void inline clrWriteTimeoutFlag() {
00437         flags &= ~(GLCD_FLAGS_TIME_OUT_ON_WRITE_BIT);
00438     }
00439
00443     void inline setOutOfRangeFlag() {
00444         flags |= GLCD_FLAGS_PLOT_OUT_OF_RANGE_BIT;
00445     }
00446
00450     void inline clrOutOfRangeFlag() {
00451         flags &= ~(GLCD_FLAGS_PLOT_OUT_OF_RANGE_BIT);
00452     }
00453
00457     void inline setReadInAllChipsFlag() {
00458         flags |= GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT;
00459     }
00460
00464     void inline clrReadInAllChipsFlag() {
00465         flags &= ~(GLCD_FLAGS_READ_IN_ALL_CHIPS_BIT);
00466     }
00467 };
00468
00469 #endif /* __ARDUINO_DRIVER_GLCD_H__ */
```

5.5 GlcdStraight.cpp File Reference

`#include "GlcdStraight.h"`
Include dependency graph for GlcdStraight.cpp:



Macros

- #define __ARDUINO_DRIVER_GLCD_STRAIGHT_CPP__ 1

5.5.1 Macro Definition Documentation

5.5.1.1 #define __ARDUINO_DRIVER_GLCD_STRAIGHT_CPP__ 1

Arduino - Glcd driver.

GlcdStraight.cpp

The glcd driver functions for glcd driver, wicth implements the driver base with direct access, witout buffer.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 13 of file GlcdStraight.cpp.

5.6 GlcdStraight.cpp

```
00001
00012 #ifndef __ARDUINO_DRIVER_GLCD_STRAIGHT_CPP__
00013 #define __ARDUINO_DRIVER_GLCD_STRAIGHT_CPP__ 1
00014
00015 #include "GlcdStraight.h"
00016
00017 GlcdStraight::GlcdStraight() : Glcd() {
00018 }
00019
```

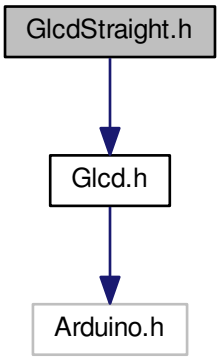
```
00020 void GlcdStraight::init(Mode mode) {
00021     unsigned char i = 0;
00022     initIo();
00023     reset();
00024     clrEnablePin();
00025     for (i = 0; i < GLCD_CHIPS; i++) {
00026         startLine[i].scrollTo = 0;
00027     }
00028     scrollTo(CHIP_ALL, 0);
00029     command(CHIP_ALL, CMD_SET_ADDRESS);
00030     command(CHIP_ALL, CMD_SET_PAGE);
00031     if (mode == MODE_ON) {
00032         command(CHIP_ALL, CMD_DISPLAY_ON_OFF |
      CMD_DISPLAY_ON_OFF_ON);
00033     } else {
00034         command(CHIP_ALL, CMD_DISPLAY_ON_OFF);
00035     }
00036 }
00037
00038 void GlcdStraight::initIo() {
00039     pinMode(GLCD_CS1_PIN, OUTPUT);
00040     pinMode(GLCD_CS2_PIN, OUTPUT);
00041     pinMode(GLCD_RS_PIN, OUTPUT);
00042     pinMode(GLCD_RW_PIN, OUTPUT);
00043     pinMode(GLCD_EN_PIN, OUTPUT);
00044
00045 #ifdef GLCD_USING_RESET
00046     pinMode(GLCD_RESET_PIN, OUTPUT);
00047 #endif
00048 }
00049
00050 void GlcdStraight::reset() {
00051 #ifdef GLCD_USING_RESET
00052     digitalWrite(GLCD_RESET_PIN, LOW);
00053     delayMicroseconds(GLCD_DELAY_RESET_US);
00054     digitalWrite(GLCD_RESET_PIN, HIGH);
00055     while (isReseting(Glcd::CHIP_1));
00056 #endif
00057 }
00058
00059 void GlcdStraight::switchRegisterSelectTo(
      RegisterSelect rs) {
00060     if (rs == RS_COMMAND) {
00061         switchRegisterSelectToCommand();
00062     } else {
00063         switchRegisterSelectToData();
00064     }
00065 }
00066
00067 void GlcdStraight::switchChipTo(Chip chip) {
00068     if (chip == CHIP_ALL) {
00069         digitalWrite(GLCD_CS1_PIN, HIGH);
00070         digitalWrite(GLCD_CS2_PIN, HIGH);
00071     } else {
00072         if (chip == CHIP_1) {
00073             digitalWrite(GLCD_CS1_PIN, HIGH);
00074             digitalWrite(GLCD_CS2_PIN, LOW);
00075         } else {
00076             digitalWrite(GLCD_CS1_PIN, LOW);
00077             digitalWrite(GLCD_CS2_PIN, HIGH);
00078         }
00079     }
00080 }
00081
00082 bool GlcdStraight::write(Chip chip, unsigned char b,
      RegisterSelect rs) {
00083
00084 #if GLCD_CHECK_FOR_BUSY_ON_WRITE == 1
00085     unsigned char attempts = GLCD_DEFAULT_ATTEMPTS_ON_BUSY;
00086     while (isBusy(chip) && attempts--) {
00087         if (attempts == 0) {
00088             setWriteTimeoutFlag();
00089             return 0;
00090         }
00091     }
00092 #endif
00093
00094     switchRegisterSelectTo(rs);
00095     switchRwToWrite();
00096     switchChipTo(chip);
00097     setEnablePin();
00098     delayMicroseconds(GLCD_DELAY_TDSU_US);
00099     busOutputDirection();
00100     writeToBus(b);
00101     delayMicroseconds(GLCD_DELAY_TDHW_US);
00102     clrEnablePin();
00103     disableChips();
```

```
00104
00105     return 1;
00106 }
00107
00108 unsigned char GlcdStraight::read(Chip chip, RegisterSelect rs) {
00109
00110     unsigned char b = 0;
00111
00112     unsigned char i, howManyReads = 1;
00113
00114     // In some cases is necessary to write data in all chips,
00115     // But, to know if the module is not busy is necessary a read
00116     // operation to get the glcd status. But read in all chips will
00117     // cause conflicts in the bus.
00118     // I decided to choose the first chip in this case and go on, but
00119     // you can make another decision.
00120     if (chip == CHIP_ALL) {
00121         setReadInAllChipsFlag();
00122         // BUG? was ==
00123         chip = CHIP_1;
00124     }
00125
00126     busInputDirection();
00127     clrEnablePin();
00128     switchChipTo(chip);
00129     switchRegisterSelectTo(rs);
00130     switchRwToRead();
00131
00132     // To read the contents of display data RAM, twice access of read
00133     // instruction is needed. In first access, data in display data RAM
00134     // is latched into output register. In second access, MPU can read
00135     // data which is latched. That is, to read the data in display data
00136     // RAM, it needs dummy read. But status read is not needed dummy
00137     // read.
00138     if (rs == RS_DATA) {
00139         howManyReads = 2;
00140     }
00141
00142     for (i = 0; i < howManyReads; i++) {
00143         setEnablePin();
00144         delayMicroseconds(GLCD_DELAY_TD_US);
00145         b = readFromBus();
00146         clrEnablePin();
00147     }
00148
00149     disableChips();
00150
00151     return b;
00152 }
00153 #endif /* __ARDUINO_DRIVER_GLCD_STRAIGHT_CPP__ */
```

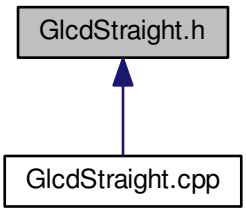
5.7 GlcdStraight.h File Reference

```
#include <Glcd.h>
```

Include dependency graph for GlcdStraight.h:



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



Classes

- class GlcdStraight

Macros

- #define GLCD_CS1_PIN 12
- #define GLCD_CS2_PIN 13
- #define GLCD_RS_PIN 3
- #define GLCD_RW_PIN 2
- #define GLCD_EN_PIN A0
- #define GLCD_BUS_PIN_NIBBLE_LOW PIND
- #define GLCD_BUS_PIN_NIBBLE_HIGH PINB
- #define GLCD_BUS_DDR_NIBBLE_LOW DDRD
- #define GLCD_BUS_DDR_NIBBLE_HIGH DDRB
- #define GLCD_BUS_PORT_NIBBLE_LOW PORTD
- #define GLCD_BUS_PORT_NIBBLE_HIGH PORTB

- #define GLCD_DELAY_TDSU_US 0x0a
- #define GLCD_DELAY_TDHW_US 0x0a
- #define GLCD_DELAY_TD_US 0x0a
- #define GLCD_DELAY_RESET_US 0x0a
- #define GLCD_CHECK_FOR_BUSY_ON_WRITE 0x00
- #define GLCD_DEFAULT_ATTEMPTS_ON_BUSY 0x0a

5.7.1 Macro Definition Documentation

5.7.1.1 #define GLCD_BUS_DDR_NIBBLE_HIGH DDRB

Definition at line 61 of file GlcdStraight.h.

5.7.1.2 #define GLCD_BUS_DDR_NIBBLE_LOW DDRD

Definition at line 60 of file GlcdStraight.h.

5.7.1.3 #define GLCD_BUS_PIN_NIBBLE_HIGH PINB

Definition at line 58 of file GlcdStraight.h.

5.7.1.4 #define GLCD_BUS_PIN_NIBBLE_LOW PIND

Definition at line 57 of file GlcdStraight.h.

5.7.1.5 #define GLCD_BUS_PORT_NIBBLE_HIGH PORTB

Definition at line 64 of file GlcdStraight.h.

5.7.1.6 #define GLCD_BUS_PORT_NIBBLE_LOW PORTD

Definition at line 63 of file GlcdStraight.h.

5.7.1.7 #define GLCD_CHECK_FOR_BUSY_ON_WRITE 0x00

Definition at line 78 of file GlcdStraight.h.

5.7.1.8 #define GLCD_CS1_PIN 12

Arduino - Glcd driver.

GlcdStraight.h

The header file for glcd driver, wicth implements the driver base with direct access, witout buffer.

01 - GND 02 - VDD 03 - V0 04 - D/I 05 - R/W 06 - E 07 - D0 08 - D1 09 - D2 10 - D3 11 - D4 12 - D5 13 - D6 14 - D7 15 - CS1 16 - CS2 17 - RST 18 - VEE 19 - LED 5v 20 - LED 0v

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 39 of file GlcdStraight.h.

5.7.1.9 #define GLCD_CS2_PIN 13

Definition at line 40 of file GlcdStraight.h.

5.7.1.10 #define GLCD_DEFAULT_ATTEMPTS_ON_BUSY 0x0a

Definition at line 79 of file GlcdStraight.h.

5.7.1.11 #define GLCD_DELAY_RESET_US 0x0a

Definition at line 76 of file GlcdStraight.h.

5.7.1.12 #define GLCD_DELAY_TD_US 0x0a

Definition at line 73 of file GlcdStraight.h.

5.7.1.13 #define GLCD_DELAY_TDHW_US 0x0a

Definition at line 70 of file GlcdStraight.h.

5.7.1.14 #define GLCD_DELAY_TDSU_US 0x0a

Definition at line 67 of file GlcdStraight.h.

5.7.1.15 #define GLCD_EN_PIN A0

Definition at line 44 of file GlcdStraight.h.

5.7.1.16 #define GLCD_RS_PIN 3

Definition at line 42 of file GlcdStraight.h.

5.7.1.17 #define GLCD_RW_PIN 2

Definition at line 43 of file GlcdStraight.h.

5.8 GlcdStraight.h

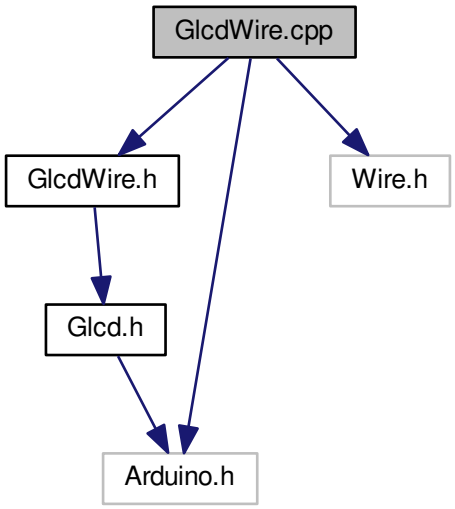
```
00001
00034 #ifndef __ARDUINO_DRIVER_GLCD_STRAIGHT_H__
00035 #define __ARDUINO_DRIVER_GLCD_STRAIGHT_H__ 1
00036
00037 #include <Glcd.h>
00038
00039 #define GLCD_CS1_PIN                                    12
00040 #define GLCD_CS2_PIN                                    13
00041
00042 #define GLCD_RS_PIN                                     3
00043 #define GLCD_RW_PIN                                     2
00044 #define GLCD_EN_PIN                                     A0
00045
00046 #ifdef GLCD_USING_RESET
00047 #define GLCD_RESET_PIN                                  13
00048 #endif
00049
00050 /*
00051  * Arduino layout
00052  *
00053  * B (digital pin 8 to 13)
00054  * C (analog input pins)
00055  * D (digital pins 0 to 7)
00056  */
00057 #define GLCD_BUS_PIN_NIBBLE_LOW                         PIND
00058 #define GLCD_BUS_PIN_NIBBLE_HIGH                        PINB
00059
00060 #define GLCD_BUS_DDR_NIBBLE_LOW                         DDRD
00061 #define GLCD_BUS_DDR_NIBBLE_HIGH                        DDRB
00062
00063 #define GLCD_BUS_PORT_NIBBLE_LOW                        PORTD
00064 #define GLCD_BUS_PORT_NIBBLE_HIGH                       PORTB
00065
00066 // Data setup time TDSU ~= 300 ns (at 20Mhz it will be about 15 cycles)
00067 #define GLCD_DELAY_TDSU_US                              0x0a
00068
00069 // Data hold time (write) TDHW ~= 15 ns (at 20Mhz it will be about 1 cycles)
00070 #define GLCD_DELAY_TDHW_US                              0x0a
00071
00072 // Data delay time TDDR ~= 480 ns (at 20Mhz it will be about 25 cycles)
00073 #define GLCD_DELAY_TD_US                                0x0a
00074
00075 // I don't know exactly how many cycles :/
00076 #define GLCD_DELAY_RESET_US                             0x0a
```

```
00077
00078 #define GLCD_CHECK_FOR_BUSY_ON_WRITE                    0x00
00079 #define GLCD_DEFAULT_ATTEMPTS_ON_BUSY                   0x0a
00080
00081
00082 class GlcdStraight : public Glcd {
00083 public:
00084
00085     GlcdStraight();
00086
00092     void init(Mode mode);
00093
00099     void reset();
00100
00101 protected:
00102
00106     void initIo();
00107
00116     bool write(Chip chip, unsigned char b, RegisterSelect rs);
00117
00125     unsigned char read(Chip chip, RegisterSelect rs);
00126
00130     void switchRegisterSelectTo(RegisterSelect rs);
00131
00135     void switchRegisterSelectToData() {
00136         digitalWrite(GLCD_RS_PIN, HIGH);
00137     }
00138
00142     void switchRegisterSelectToCommand() {
00143         digitalWrite(GLCD_RS_PIN, LOW);
00144     }
00145
00149     void switchChipTo(Chip chip);
00150
00154     void disableChips() {
00155         digitalWrite(GLCD_CS1_PIN, LOW);
00156         digitalWrite(GLCD_CS2_PIN, LOW);
00157     }
00158
00162     void switchRwToWrite() {
00163         digitalWrite(GLCD_RW_PIN, LOW);
00164     }
00165
00169     void switchRwToRead() {
00170         digitalWrite(GLCD_RW_PIN, HIGH);
00171     }
00172
00178     void writeToBus(unsigned char b) {
00179         GLCD_BUS_PORT_NIBBLE_LOW = (
      GLCD_BUS_PORT_NIBBLE_LOW & 0x0f) | (b & 0xf0);
00180         GLCD_BUS_PORT_NIBBLE_HIGH = (
      GLCD_BUS_PORT_NIBBLE_HIGH & 0xf0) | (b & 0x0f);
00181     }
00182
00188     unsigned char readFromBus() {
00189         return (GLCD_BUS_PIN_NIBBLE_LOW & 0xf0) | (
      GLCD_BUS_PIN_NIBBLE_HIGH & 0x0f);
00190     }
00191
00195     void busOutputDirection() {
00196         GLCD_BUS_DDR_NIBBLE_LOW |= 0xf0;
00197         GLCD_BUS_DDR_NIBBLE_HIGH |= 0x0f;
00198     }
00199
00203     void busInputDirection() {
00204         GLCD_BUS_DDR_NIBBLE_LOW &= 0x0f;
00205         GLCD_BUS_DDR_NIBBLE_HIGH &= 0xf0;
00206     }
00207
00211     void setEnablePin() {
00212         digitalWrite(GLCD_EN_PIN, HIGH);
00213     }
00214
00218     void clrEnablePin() {
00219         digitalWrite(GLCD_EN_PIN, LOW);
00220     }
00221 };
00222
00223 #endif /* __ARDUINO_DRIVER_GLCD_STRAIGHT_H__ */
```

5.9 GlcdWire.cpp File Reference

```
#include <GlcdWire.h>
#include <Wire.h>
#include <Arduino.h>
```

Include dependency graph for GlcdWire.cpp:



Macros

- #define __ARDUINO_DRIVER_GLCD_WIRE_CPP__ 1

5.9.1 Macro Definition Documentation

5.9.1.1 #define __ARDUINO_DRIVER_GLCD_WIRE_CPP__ 1

Arduino - Glcd driver.

GlcdWire.h

The header file for glcd driver, wicth implements the driver base using i2c with a PIC microcontroller.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 13 of file GlcdWire.cpp.

5.10 GlcdWire.cpp

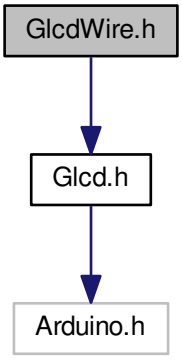
```
00001
00012 #ifndef __ARDUINO_DRIVER_GLCD_WIRE_CPP__
00013 #define __ARDUINO_DRIVER_GLCD_WIRE_CPP__ 1
00014
00015 #include <GlcdWire.h>
00016 #include <Wire.h>
```

```
00017 #include <Arduino.h>
00018
00019 GlcdWire::GlcdWire(unsigned char device) : Glcd() {
00020     this->device = device;
00021 }
00022
00023 void GlcdWire::init(Mode mode) {
00024     Wire.begin();
00025     for (unsigned char i = 0; i < GLCD_CHIPS; i++) {
00026         chipInfo[i].page = 0x00;
00027         chipInfo[i].line = 0x00;
00028     }
00029 }
00030
00031 void GlcdWire::reset() {
00032 }
00033
00034 bool GlcdWire::write(Chip chip, unsigned char b,
      RegisterSelect rs) {
00035     unsigned char cmd, page, line;
00036     bool success = false;
00037     cmd = b & 0xc0;
00038     if (rs == Glcd::RS_COMMAND) {
00039         if (cmd == (Glcd::CMD_SET_PAGE & 0xc0)) {
00040             if (chip == Glcd::CHIP_1 || chip == Glcd::CHIP_ALL) {
00041                 chipInfo[0].page = b & 0x07;
00042             }
00043             if (chip == Glcd::CHIP_2 || chip == Glcd::CHIP_ALL) {
00044                 chipInfo[1].page = b & 0x07;
00045             }
00046         } else if(cmd == (Glcd::CMD_SET_ADDRESS & 0xc0)) {
00047             if (chip == Glcd::CHIP_1 || chip == Glcd::CHIP_ALL) {
00048                 chipInfo[0].line = b & 0x3f;
00049             }
00050             if (chip == Glcd::CHIP_2 || chip == Glcd::CHIP_ALL) {
00051                 chipInfo[1].line = b & 0x3f;
00052             }
00053         }
00054     } else {
00055         if (chip == Glcd::CHIP_1 || chip == Glcd::CHIP_ALL) {
00056             page = (chipInfo[0].page << 2) | (chip & 0x03);
00057             line = chipInfo[0].line;
00058         } else {
00059             page = (chipInfo[1].page << 2) | (chip & 0x03);
00060             line = chipInfo[1].line;
00061         }
00062         Wire.beginTransmission((int)device);
00063         Wire.write(page);
00064         Wire.write(line);
00065         Wire.write(b);
00066         if (Wire.endTransmission() == 0) {
00067             success = true;
00068         }
00069     }
00070     return success;
00071 }
00072
00073 unsigned char GlcdWire::read(Chip chip, RegisterSelect rs) {
00074     unsigned char page, line;
00075     if (rs == Glcd::RS_COMMAND) {
00076         return 0x20;
00077     }
00078     if (chip == Glcd::CHIP_1 || chip == Glcd::CHIP_ALL) {
00079         page = (chipInfo[0].page << 2) | (chip & 0x03);
00080         line = chipInfo[0].line;
00081     } else {
00082         page = (chipInfo[1].page << 2) | (chip & 0x03);
00083         line = chipInfo[1].line;
00084     }
00085     Wire.beginTransmission((int)device);
00086     Wire.write(page);
00087     Wire.write(line);
00088     Wire.endTransmission();
00089     Wire.requestFrom((int)device, (int)1);
00090     while (!Wire.available());
00091     return Wire.read();
00092 }
00093
00094 #endif /* __ARDUINO_DRIVER_GLCD_WIRE_CPP__ */
```

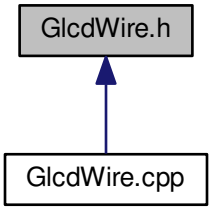
5.11 GlcdWire.h File Reference

```
#include <Glcd.h>
```

Include dependency graph for GlcdWire.h:



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



Classes

- class GlcdWire

5.12 GlcdWire.h

```
00001
00012 #ifndef __ARDUINO_DRIVER_GLCD_WIRE_H__
00013 #define __ARDUINO_DRIVER_GLCD_WIRE_H__ 1
00014
00015 #include <Glcd.h>
00016
00017 class GlcdWire : public Glcd {
00018 protected:
00019
00020     unsigned char device;
00021
00022     struct {
00023         unsigned char page;
00024         unsigned char line;
00025     } chipInfo[GLCD_CHIPS];
00026
00027 public:
```

```
00028
00034     GlcdWire(unsigned char device);
00035
00041     void init(Mode mode);
00042
00048     void reset();
00049
00058     bool write(Chip chip, unsigned char b, RegisterSelect rs);
00059
00067     unsigned char read(Chip chip, RegisterSelect rs);
00068
00069 private:
00070
00091     unsigned char makeHeader(Chip chip, RegisterSelect rs,
      Rw rw);
00092 };
00093
00094 #endif /* __ARDUINO_DRIVER_GLCD_WIRE_H__ */
00095
```

Index