NAME

LC100.cpp −

Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

SYNOPSIS

```
#include 'LC100.h'
#include <Arduino.h>
#include <Print.h>
#include <stdint.h>
#include <string.h>
```

Detailed Description

Copyright 2012 Digital Aggregates Corporation, Colorado, USA
Licensed under the terms in **README.h**
Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

Definition in file **LC100.cpp**.

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

LC100.h −

Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

SYNOPSIS

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

Classes

struct **com::diag::amigo::Display**

This pure (abstract) class defines the interface that the **LC100** *software expects to be implemented by any display that it uses.*

class **com::diag::amigo::LC100Base**

This is the common base (super) class for the **LC100** *software that does all of the heavy lifting for any* **LC100** *template instantiation regardless of the display dimensions.*

class **com::diag::amigo::LC100< _COLS_, _ROWS_ >**

This is the derived (sub) class for the **LC100** *software that contains the actual data structures whose sizes depend on the actual size of the display.*

Detailed Description

Copyright 2012 Digital Aggregates Corporation, Colorado, USA
Licensed under the terms in **README.h**
Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

Definition in file **LC100.h**.

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

Mock −

This is a mock display that merely traces itself on the serial output and implements the movement keys used by the visual editor (vi).

SYNOPSIS

Inherits **com::diag::amigo::Display**.

Public Types

enum **Read**

Public Member Functions

Mock ()
virtual **˜Mock** ()
virtual void **begin** (byte cols, byte rows)
virtual void **home** ()
virtual void **clear** ()
virtual void **setCursor** (byte col, byte row)
virtual size_t **write** (uint8_t ch)
virtual **Read read** ()

Detailed Description

This is a mock display that merely traces itself on the serial output and implements the movement keys used by the visual editor (vi).

Definition at line 26 of file TinyTerminal.ino.

Member Function Documentation

virtual void Mock::begin (byte cols, byte rows) `[inline, virtual]`

Initialize. This needs to be done only once.

Parameters:

cols is the number of columns in this display.
rows is the number of rows in this display.

Implements **com::diag::amigo::Display**.

Definition at line 47 of file TinyTerminal.ino.

virtual Read Mock::read () `[inline, virtual]`

Read the joystick. The returned value will be an enumerated value indicating movement left, down, up, or right, a select indication, or no input.

Returns:

an enumerated value.

Implements **com::diag::amigo::Display**.

Definition at line 106 of file TinyTerminal.ino.

virtual void Mock::setCursor (byte col, byte row) `[inline, virtual]`

Place the cursor at the specified position. **Parameters:**

col is the zero-based column number.
row is the zero-based row number.

Implements **com::diag::amigo::Display**.

Definition at line 73 of file TinyTerminal.ino.

virtual size_t Mock::write (uint8_t ch) `[inline, virtual]`

Write a character to the display. The signed value that is returned will typically be one, but can be zero if the specified character was somehow invalid, or negative if an error writing the character occurred.

Parameters:

ch is the character that is written to the display.

Returns:
the number of characters written.

Implements **com::diag::amigo::Display**.

Definition at line 87 of file TinyTerminal.ino.

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

README.h −

This is the README for this project.

SYNOPSIS

Detailed Description

This is the README for this project.

Definition in file **README.h**.

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

TinyTerminal.ino −

Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

SYNOPSIS

```
#include <LiquidCrystal.h>
#include 'LC100.h'
```

Classes

class **Mock**

This is a mock display that merely traces itself on the serial output and implements the movement keys used by the visual editor (vi).

Defines

#define **DEBUG** (0)

Functions

static void **test** ()
void **setup** ()
void **loop** ()

Variables

static **Mock display**
static const byte **COLS** = 16
static const byte **ROWS** = 2
static **com::diag::amigo::LC100**< **COLS**, **ROWS** > **lc100** (**display**, false, true, 1000/8)

Detailed Description

Copyright 2012 Digital Aggregates Corporation, Colorado, USA
Licensed under the terms in **README.h**
Chip Overclock mailto:coverclock@diag.com
http://www.diag.com/navigation/downloads/Amigo.html

Definition in file **TinyTerminal.ino**.

Define Documentation

#define DEBUG (0)

If DEBUG is defined, TinyTerminal replaces the interface to the actual LCD hardware with a mock object that implements the same interface. The mock object just traces itself on the serial output.

Definition at line 18 of file TinyTerminal.ino.

Function Documentation

void loop ()

Loop is executed continuously by the Arduino run-time software, which also does some other housekeeping each iteration, like polling the serial port for activity. The timing of the execution of this function is asynchronous. This function checks for input on the serial port and if it exists passes all of the buffered input characters to the LC100 software. It also checks for input from the LC100 software from the joystick and if it exists passes it to the serial port.

Definition at line 475 of file TinyTerminal.ino.

References lc100.

void setup ()

Setup is executed once by the Arduino run-time software. This function initializes the serial port, the LC100 software (which in turn initializes the underlying display hardware), and runs a unit test which may or may not do anything.

Definition at line 460 of file TinyTerminal.ino.

References lc100, and test().

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

com::diag::amigo::Display −

This pure (abstract) class defines the interface that the **LC100** software expects to be implemented by any display that it uses.

SYNOPSIS

```
#include <LC100.h>
```

Inherited by **Mock**.

Public Types

enum **Read**

Public Member Functions

virtual void **begin** (byte cols, byte rows)=0
virtual void **home** ()=0
virtual void **clear** ()=0
virtual void **setCursor** (byte col, byte row)=0
virtual size_t **write** (uint8_t ch)=0
virtual **Read read** ()=0

Detailed Description

This pure (abstract) class defines the interface that the **LC100** software expects to be implemented by any display that it uses.

Definition at line 24 of file LC100.h.

Member Function Documentation

virtual void com::diag::amigo::Display::begin (byte cols, byte rows) [pure virtual]

Initialize. This needs to be done only once.

Parameters:

cols is the number of columns in this display.
rows is the number of rows in this display.

Implemented in **Mock**.

Referenced by com::diag::amigo::LC100Base::begin().

virtual Read com::diag::amigo::Display::read () [pure virtual]

Read the joystick. The returned value will be an enumerated value indicating movement left, down, up, or right, a select indication, or no input.

Returns:

an enumerated value.

Implemented in **Mock**.

Referenced by com::diag::amigo::LC100Base::read().

virtual void com::diag::amigo::Display::setCursor (byte col, byte row) [pure virtual]

Place the cursor at the specified position. The column and row coordinates are taken modulo of the actual display dimensions.

Parameters:

col is the zero-based column number.
row is the zero-based row number.

Implemented in **Mock**.

Referenced by com::diag::amigo::LC100Base::down(), com::diag::amigo::LC100Base::setCursor(), and com::diag::amigo::LC100Base::up().

virtual size_t com::diag::amigo::Display::write (uint8_t ch) `[pure virtual]`

Write a character to the display. The signed value that is returned will typically be one, but can be zero if the specified character was somehow invalid, or negative if an error writing the character occurred.

Parameters:

ch is the character that is written to the display.

Returns:

the number of characters written.

Implemented in **Mock**.

Referenced by com::diag::amigo::LC100Base::down(), com::diag::amigo::LC100Base::emit(), and com::diag::amigo::LC100Base::up().

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

com::diag::amigo::LC100 −

This is the derived (sub) class for the **LC100** software that contains the actual data structures whose sizes depend on the actual size of the display.

SYNOPSIS

```
#include <LC100.h>
```

Inherits **com::diag::amigo::LC100Base**.

Public Types

enum **Ascii**

Public Member Functions

LC100 (**Display** &**display**, boolean sample=false, boolean debug=false, int ms=0)
virtual **˜LC100** ()
void **begin** ()
void **setCursor** (byte col, byte row)
void **home** ()
void **down** ()
void **up** ()
void **erase** (byte colFrom, byte rowFrom, byte colTo, byte rowTo)
void **clear** ()
int **read** (char *buffer)
size_t **write** (uint8_t ch)

Public Attributes

const byte **COLS**
const byte **ROWS**

Protected Types

enum **Constant**
enum **State**
enum **Action**

Protected Member Functions

byte **index** (byte col, byte row)
byte **index** (byte row)
size_t **emit** (uint8_t ch)
size_t **frame** (uint8_t ch)
byte **one** (byte value)

Detailed Description

template<byte _COLS_, byte _ROWS_> class com::diag::amigo::LC100< _COLS_, _ROWS_ >

This is the derived (sub) class for the **LC100** software that contains the actual data structures whose sizes depend on the actual size of the display.

It is templatized so that the display dimensions can be passed as arguments at compile time.

Definition at line 368 of file LC100.h.

Member Enumeration Documentation

enum com::diag::amigo::LC100Base::Action `[protected, inherited]`

Actions which the push down automaton may execute. CONSUMED is the default action. All of the enumerated values are printable, making it easy to trace the PDA as it executes.

Definition at line 124 of file LC100.h.

enum com::diag::amigo::LC100Base::State `[protected, inherited]`

States in which the push down automaton may be. DATA is the start state. There is no end state. All of the enumerated values are printable, making it easy to trace the PDA as it executes.

Definition at line 109 of file LC100.h.

Constructor & Destructor Documentation

template<byte _COLS_, byte _ROWS_> com::diag::amigo::LC100< _COLS_, _ROWS_ >::LC100 (Display & display, boolean sample = `false`**, boolean debug** = `false`**, int ms** = `0`**)** `[inline]`

Ctor. **Parameters:**

display refers to the object that implements the **Display** interface.
sample if true causes the joystick value to be returned continuously as long as a button is pressed; if false, the joystick value is returned intermittently when the button is released.
debug enables debug output if debugging was compiled in.
ms is the number of milliseconds to delay between each individual update of the display, which can make debugging a lot easier.

Definition at line 390 of file LC100.h.

Member Function Documentation

void com::diag::amigo::LC100Base::begin () `[inherited]`

Initialize this object. This only needs to be called once.

Definition at line 724 of file LC100.cpp.

References com::diag::amigo::Display::begin(), com::diag::amigo::LC100Base::clear(), com::diag::amigo::LC100Base::COLS, and com::diag::amigo::LC100Base::ROWS.

void com::diag::amigo::LC100Base::down () `[inherited]`

Scroll the scroll area down, leaving the cursor placed at a blank line at the top of the scroll area. By default, the scroll area is the entire display.

Definition at line 176 of file LC100.cpp.

References com::diag::amigo::Display::clear(), com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::ROWS, com::diag::amigo::LC100Base::setCursor(), com::diag::amigo::Display::setCursor(), and com::diag::amigo::Display::write().

Referenced by com::diag::amigo::LC100Base::write().

size_t com::diag::amigo::LC100Base::emit (uint8_t ch) `[protected, inherited]`

Emit a character, which both displays it on the actual display and stores it appropriately in the frame buffer. The returned value is the number of characters processed, which is nominally one but may be zero if the character is somehow invalid or negative if an error occurred.

Parameters:

ch is the character to be emitted.

Returns:

the number of characters emitted.

Definition at line 69 of file LC100.cpp.

References com::diag::amigo::LC100Base::COLS, com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::ROWS, and com::diag::amigo::Display::write().

Referenced by com::diag::amigo::LC100Base::erase(), com::diag::amigo::LC100Base::frame(), and com::diag::amigo::LC100Base::write().

void com::diag::amigo::LC100Base::erase (byte colFrom, byte rowFrom, byte colTo, byte rowTo) `[inherited]`

Erase a square bordered by the specified upper left and lower right corners inclusive. Erasing is done by writing blanks into the display left to right, top to bottom. The cursor is placed back in its original position.

Parameters:

colFrom is the zero-based column of the upper left corner of the erased square.
rowFrom is the zero-based row of the upper left corner of the erased square.
colTo is the zero-based column of the lower right corner of the erased square.
rowTo is the zero-based row of the lower right corner of the erased square.

Definition at line 109 of file LC100.cpp.

References com::diag::amigo::LC100Base::COLS, com::diag::amigo::LC100Base::emit(), com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::ROWS, and com::diag::amigo::LC100Base::setCursor().

Referenced by com::diag::amigo::LC100Base::frame(), and com::diag::amigo::LC100Base::write().

size_t com::diag::amigo::LC100Base::frame (uint8_t ch) `[protected, inherited]`

Frame a character appropriately by dealing with line wrapping (if enabled) and screen scrolling (ditto).
Parameters:
ch is the character to be framed.

Returns:
the number of characters framed.

Definition at line 144 of file LC100.cpp.

References com::diag::amigo::LC100Base::COLS, com::diag::amigo::LC100Base::emit(), com::diag::amigo::LC100Base::erase(), com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::ROWS, com::diag::amigo::LC100Base::setCursor(), and com::diag::amigo::LC100Base::up().

Referenced by com::diag::amigo::LC100Base::write().

byte com::diag::amigo::LC100Base::index (byte row) `[protected, inherited]`

Concert a zero-based row value into an array index. **Parameters:**
row is the zero-based row value.

Returns:
an array index.

Definition at line 35 of file LC100.cpp.

References com::diag::amigo::LC100Base::ROWS.

byte com::diag::amigo::LC100Base::index (byte col, byte row) `[protected, inherited]`

Convert a zero-based column and row coordinate into a frame buffer index. **Parameters:**
col is the zero-based column value.
row is the zero-based row value.

Returns:
a frame buffer index.

Definition at line 39 of file LC100.cpp.

References com::diag::amigo::LC100Base::COLS.

Referenced by com::diag::amigo::LC100Base::down(), com::diag::amigo::LC100Base::emit(), com::diag::amigo::LC100Base::erase(), com::diag::amigo::LC100Base::frame(), com::diag::amigo::LC100Base::up(), and com::diag::amigo::LC100Base::write().

byte com::diag::amigo::LC100Base::one (byte value) `[protected, inherited]`

Convert a one-based column or row value into a zero-based column or row value. **Parameters:**
value is the one-based value.

Returns:
the zero-based value.

Definition at line 43 of file LC100.cpp.

Referenced by com::diag::amigo::LC100Base::write().

int com::diag::amigo::LC100Base::read (char * buffer) `[inherited]`

Read the current joy stick stimulus into a buffer of at least four bytes in length. If the joy stick is indicating movement, the buffer will contain a VT100 (ANSI) arrow escape sequence indicating the direction of movement. The buffer will be nul-terminated, allowing it to be written directly to the serial port. Return the number of bytes placed into the buffer, zero indicating that there is no joy stick stimulus at this time.

Parameters:

buffer points to a buffer of at least four bytes in length.

Returns:

the number of bytes placed in the buffer.

Definition at line 667 of file LC100.cpp.

References com::diag::amigo::Display::read().

void com::diag::amigo::LC100Base::setCursor (byte col, byte row) `[inherited]`

Place the cursor at the specified position. The column and row coordinates are taken modulo of the actual display dimensions.

Parameters:

col is the zero-based column number.
row is the zero-based row number.

Definition at line 51 of file LC100.cpp.

References com::diag::amigo::LC100Base::COLS, com::diag::amigo::LC100Base::ROWS, and com::diag::amigo::Display::setCursor().

Referenced by com::diag::amigo::LC100Base::clear(), com::diag::amigo::LC100Base::down(), com::diag::amigo::LC100Base::erase(), com::diag::amigo::LC100Base::frame(), com::diag::amigo::LC100Base::up(), and com::diag::amigo::LC100Base::write().

void com::diag::amigo::LC100Base::up () `[inherited]`

Scroll the scroll area up, leaving the cursor placed at a blank line at the bottom of the scroll area. By default, the scroll area is the entire display.

Definition at line 197 of file LC100.cpp.

References com::diag::amigo::Display::clear(), com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::ROWS, com::diag::amigo::LC100Base::setCursor(), com::diag::amigo::Display::setCursor(), and com::diag::amigo::Display::write().

Referenced by com::diag::amigo::LC100Base::frame(), and com::diag::amigo::LC100Base::write().

size_t com::diag::amigo::LC100Base::write (uint8_t ch) `[inherited]`

Write the current character to the display. This character may be part of a VT100 (ANSI) escape sequence, in which case it is not actually written to the display, but will be executed once the complete escape sequence is captured.

Parameters:

ch is the character to be written to the display.

Returns:

the number of characters processed, which could be zero if the character is somehow invalid in the context of the current escape sequence, or negative is an error occurred.

Definition at line 246 of file LC100.cpp.

References com::diag::amigo::LC100Base::clear(), com::diag::amigo::LC100Base::COLS, com::diag::amigo::LC100Base::down(), com::diag::amigo::LC100Base::emit(), com::diag::amigo::LC100Base::erase(), com::diag::amigo::LC100Base::frame(), com::diag::amigo::Display::home(), com::diag::amigo::LC100Base::index(), com::diag::amigo::LC100Base::one(), com::diag::amigo::LC100Base::ROWS, com::diag::amigo::LC100Base::setCursor(), and com::diag::amigo::LC100Base::up().

Author

Generated automatically by Doxygen for Amigo from the source code.

NAME

com::diag::amigo::LC100Base −

This is the common base (super) class for the **LC100** software that does all of the heavy lifting for any **LC100** template instantiation regardless of the display dimensions.

SYNOPSIS

`#include <LC100.h>`

Inherited by **com::diag::amigo::LC100< _COLS_, _ROWS_ >**.

Public Types

enum **Ascii**

Public Member Functions

LC100Base (**Display** &**display**, byte cols, byte rows, byte *lineLengthArray, uint8_t *frameBufferArray, boolean *tabSettingsArray, boolean sample=false, boolean debug=false, int ms=0)
virtual **~LC100Base** ()
void **begin** ()
void **setCursor** (byte col, byte row)
void **home** ()
void **down** ()
void **up** ()
void **erase** (byte colFrom, byte rowFrom, byte colTo, byte rowTo)
void **clear** ()
int **read** (char *buffer)
size_t **write** (uint8_t ch)

Public Attributes

const byte **COLS**
const byte **ROWS**

Protected Types

enum **Constant**
enum **State**
enum **Action**

Protected Member Functions

byte **index** (byte col, byte row)
byte **index** (byte row)
size_t **emit** (uint8_t ch)
size_t **frame** (uint8_t ch)
byte **one** (byte value)

Detailed Description

This is the common base (super) class for the **LC100** software that does all of the heavy lifting for any **LC100** template instantiation regardless of the display dimensions.

It derives from (extends) the Arduino Print class so that any of the usual Print methods can be used by an application. This class is not intended to be used by itself, although there is no reason why you couldn't do so.

Definition at line 91 of file LC100.h.

Member Enumeration Documentation

enum com::diag::amigo::LC100Base::Action [protected]

Actions which the push down automaton may execute. CONSUMED is the default action. All of the enumerated values are printable, making it easy to trace the PDA as it executes.

Definition at line 124 of file LC100.h.

enum com::diag::amigo::LC100Base::State `[protected]`

States in which the push down automaton may be. DATA is the start state. There is no end state. All of the enumerated values are printable, making it easy to trace the PDA as it executes.

Definition at line 109 of file LC100.h.

Constructor & Destructor Documentation

com::diag::amigo::LC100Base::LC100Base (Display & display, byte cols, byte rows, byte * lineLengthArray, uint8_t * frameBufferArray, boolean * tabSettingsArray, boolean sample = `false`**, boolean debug** = `false`**, int ms** = `0`**)** `[inline]`

Ctor. **Parameters:**

display refers to the object that implements the **Display** interface.
cols is the number of columns in the display.
rows is the number of rows in the display.
lineLengthArray points to an array of dimension [rows] which will be used to store the length in bytes of the displayed line in each row of the display.
frameBufferArray points to an array of dimensions [rows][cols] that is used as a frame buffer to support scrolling.
tabSettingsArray points to an array of dimension [cols] that is used to keep track of the tab settings for the display.
sample if true causes the joystick value to be returned continuously as long as a button is pressed; if false, the joystick value is returned intermittently when the button is released.
debug enables debug output if debugging was compiled in.
ms is the number of milliseconds to delay between each individual update of the display, which can make debugging a lot easier.

Definition at line 162 of file LC100.h.

Member Function Documentation

void com::diag::amigo::LC100Base::begin ()

Initialize this object. This only needs to be called once.

Definition at line 724 of file LC100.cpp.

References com::diag::amigo::Display::begin(), clear(), COLS, and ROWS.

void com::diag::amigo::LC100Base::down ()

Scroll the scroll area down, leaving the cursor placed at a blank line at the top of the scroll area. By default, the scroll area is the entire display.

Definition at line 176 of file LC100.cpp.

References com::diag::amigo::Display::clear(), index(), ROWS, setCursor(), com::diag::amigo::Display::setCursor(), and com::diag::amigo::Display::write().

Referenced by write().

size_t com::diag::amigo::LC100Base::emit (uint8_t ch) `[protected]`

Emit a character, which both displays it on the actual display and stores it appropriately in the frame buffer. The returned value is the number of characters processed, which is nominally one but may be zero if the character is somehow invalid or negative if an error occurred.

Parameters:

ch is the character to be emitted.

Returns:

the number of characters emitted.

Definition at line 69 of file LC100.cpp.

References COLS, index(), ROWS, and com::diag::amigo::Display::write().

Referenced by erase(), frame(), and write().

void com::diag::amigo::LC100Base::erase (byte colFrom, byte rowFrom, byte colTo, byte rowTo)

Erase a square bordered by the specified upper left and lower right corners inclusive. Erasing is done by writing blanks into the display left to right, top to bottom. The cursor is placed back in its original position.

Parameters:

colFrom is the zero-based column of the upper left corner of the erased square.
rowFrom is the zero-based row of the upper left corner of the erased square.
colTo is the zero-based column of the lower right corner of the erased square.
rowTo is the zero-based row of the lower right corner of the erased square.

Definition at line 109 of file LC100.cpp.

References COLS, emit(), index(), ROWS, and setCursor().

Referenced by frame(), and write().

size_t com::diag::amigo::LC100Base::frame (uint8_t ch) `[protected]`

Frame a character appropriately by dealing with line wrapping (if enabled) and screen scrolling (ditto).
Parameters:

ch is the character to be framed.

Returns:

the number of characters framed.

Definition at line 144 of file LC100.cpp.

References COLS, emit(), erase(), index(), ROWS, setCursor(), and up().

Referenced by write().

byte com::diag::amigo::LC100Base::index (byte row) `[protected]`

Concert a zero-based row value into an array index. **Parameters:**

row is the zero-based row value.

Returns:

an array index.

Definition at line 35 of file LC100.cpp.

References ROWS.

byte com::diag::amigo::LC100Base::index (byte col, byte row) `[protected]`

Convert a zero-based column and row coordinate into a frame buffer index. **Parameters:**

col is the zero-based column value.
row is the zero-based row value.

Returns:

a frame buffer index.

Definition at line 39 of file LC100.cpp.

References COLS.

Referenced by down(), emit(), erase(), frame(), up(), and write().

byte com::diag::amigo::LC100Base::one (byte value) `[protected]`

Convert a one-based column or row value into a zero-based column or row value. **Parameters:**

value is the one-based value.

Returns:

the zero-based value.

Definition at line 43 of file LC100.cpp.

Referenced by write().

int com::diag::amigo::LC100Base::read (char * buffer)

Read the current joy stick stimulus into a buffer of at least four bytes in length. If the joy stick is indicating movement, the buffer will contain a VT100 (ANSI) arrow escape sequence indicating the direction of movement. The buffer will be nul-terminated, allowing it to be written directly to the serial port. Return the number of bytes placed into the buffer, zero indicating that there is no joy stick stimulus at this time.

Parameters:

buffer points to a buffer of at least four bytes in length.

Returns:

the number of bytes placed in the buffer.

Definition at line 667 of file LC100.cpp.

References com::diag::amigo::Display::read().

void com::diag::amigo::LC100Base::setCursor (byte col, byte row)

Place the cursor at the specified position. The column and row coordinates are taken modulo of the actual display dimensions.

Parameters:

col is the zero-based column number.
row is the zero-based row number.

Definition at line 51 of file LC100.cpp.

References COLS, ROWS, and com::diag::amigo::Display::setCursor().

Referenced by clear(), down(), erase(), frame(), up(), and write().

void com::diag::amigo::LC100Base::up ()

Scroll the scroll area up, leaving the cursor placed at a blank line at the bottom of the scroll area. By default, the scroll area is the entire display.

Definition at line 197 of file LC100.cpp.

References com::diag::amigo::Display::clear(), index(), ROWS, setCursor(), com::diag::amigo::Display::setCursor(), and com::diag::amigo::Display::write().

Referenced by frame(), and write().

size_t com::diag::amigo::LC100Base::write (uint8_t ch)

Write the current character to the display. This character may be part of a VT100 (ANSI) escape sequence, in which case it is not actually written to the display, but will be executed once the complete escape sequence is captured.

Parameters:

ch is the character to be written to the display.

Returns:

the number of characters processed, which could be zero if the character is somehow invalid in the context of the current escape sequence, or negative is an error occurred.

Definition at line 246 of file LC100.cpp.

References clear(), COLS, down(), emit(), erase(), frame(), com::diag::amigo::Display::home(), index(), one(), ROWS, setCursor(), and up().

Author

Generated automatically by Doxygen for Amigo from the source code.