Amigo

0.2.0

Generated by Doxygen 1.6.3

Tue Feb 21 09:26:21 2012

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

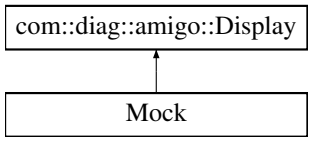
Class Documentation

4.1 com::diag::amigo::Display Struct Reference

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

`#include <LC100.h>`

Inheritance diagram for com::diag::amigo::Display:



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

4.1.1 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.

4.1.2 Member Function Documentation

4.1.2.1 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().

4.1.2.2 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().

4.1.2.3 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().

4.1.2.4 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().

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

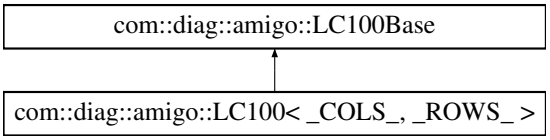
- LC100.h

4.2 com::diag::amigo::LC100< _COLS_, _ROWS_ > Class Template Reference

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.

`#include <LC100.h>`

Inheritance diagram for com::diag::amigo::LC100< _COLS_, _ROWS_ >:



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)

4.2.1 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.

4.2.2 Member Enumeration Documentation

4.2.2.1 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.

4.2.2.2 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.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 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.

4.2.4 Member Function Documentation

4.2.4.1 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.

4.2.4.2 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().

4.2.4.3 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().

4.2.4.4 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().

4.2.4.5 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().

4.2.4.6 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.

4.2.4.7 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().

4.2.4.8 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().

4.2.4.9 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().

4.2.4.10 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().

4.2.4.11 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().

4.2.4.12 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().

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

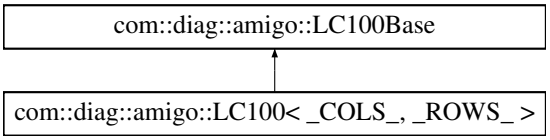
- LC100.h

4.3 com::diag::amigo::LC100Base Class Reference

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.

`#include <LC100.h>`

Inheritance diagram for com::diag::amigo::LC100Base:



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)

4.3.1 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.

4.3.2 Member Enumeration Documentation

4.3.2.1 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.

4.3.2.2 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.

4.3.3 Constructor & Destructor Documentation

4.3.3.1 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.

4.3.4 Member Function Documentation

4.3.4.1 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.

4.3.4.2 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().

4.3.4.3 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().

4.3.4.4 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().

4.3.4.5 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().

4.3.4.6 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.

4.3.4.7 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().

4.3.4.8 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().

4.3.4.9 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().

4.3.4.10 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().

4.3.4.11 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().

4.3.4.12 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().

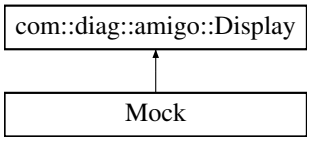
The documentation for this class was generated from the following files:

- LC100.h
- LC100.cpp

4.4 Mock Class Reference

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

Inheritance diagram for Mock:



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 ()

4.4.1 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.

4.4.2 Member Function Documentation

4.4.2.1 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.

4.4.2.2 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.

4.4.2.3 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.

4.4.2.4 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.

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

- TinyTerminal.ino

Chapter 5

File Documentation

5.1 LC100.cpp File Reference

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

.

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

5.1.1 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.

5.2 LC100.h File Reference

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

.

```
#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.

5.2.1 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.

5.3 README.h File Reference

This is the README for this project.

5.3.1 Detailed Description

This is the README for this project.

Definition in file README.h.

5.4 TinyTerminal.ino File Reference

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

.

```
#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)

5.4.1 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.

5.4.2 Define Documentation

5.4.2.1 #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.

5.4.3 Function Documentation

5.4.3.1 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.

5.4.3.2 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().