GUD STM32-Discovery Code Library
Reference Manual

Cody Johnson
Date of Issue: 7/14/2016 (V 1.0)
Revision: 7/14/2016 (V 1.0)

File Index

File List

Here is a list of all files with brief descriptions:

File Documentation

PROJ_TEST_2/GUD900.c File Reference

```
#include "GUD900.h"
#include "GUD_I2C.h"
#include "GUD_SPI.h"
#include "GUD_USART.h"
#include <stddef.h>
```

Macros

- #define **ASYNCH** 0
- #define **SPI** 1
- #define **I2C** 2

Functions

- void **GUD_Init** (unsigned width, unsigned height)
- void **GUD_print** (char data)
- void **GUD_back** ()
- void **GUD_forward** ()
- void **GUD_lineFeed** ()
- void **GUD_home** ()
- void **GUD_carriageReturn** ()
- void **GUD_crlf** ()
- void **GUD_XY** (unsigned x, unsigned y)
- void **GUD_XY1** (unsigned x, unsigned y)
- void **GUD_usCommand** ()
- void **GUD_setCursor** (unsigned x, unsigned y)
- void **GUD_clearScreen** ()
- void **GUD_cursorOn** ()
- void **GUD_cursorOff** ()
- void **GUD_reset** ()
- void **GUD_8by16DotMode** ()
- void **GUD_useMultiByteChars** (uint8_t enable)
- void **GUD_setMultiByteCharSet** (uint8_t code)
- void **GUD_useCustomChars** (uint8_t enable)
- uint8_t **GUD_getColumn** (uint8_t *src, int col)
- void **GUD_defineCustomChar** (uint8_t code, uint8_t format, uint8_t *data)
- void **GUD_deleteCustomChar** (uint8_t code)
- void **GUD_setAsciiVariant** (uint8_t code)
- void **GUD_setCharSet** (uint8_t code)
- void **GUD_setScrollMode** (uint8_t mode)
- void **GUD_setHorizScrollSpeed** (uint8_t speed)
- void **GUD_invertOff** ()
- void **GUD_invertOn** ()
- void **GUD_setCompositionMode** (uint8_t mode)
- void **GUD_setScreenBrightness** (unsigned level)
- void **GUD_wait** (uint8_t wait)
- void **GUD_scrollScreen** (unsigned x, unsigned y, unsigned times, uint8_t speed)
- void **GUD_blinkScreenOff** ()

- void **GUD_blinkScreenOn** (uint8_t enable, uint8_t reverse, uint8_t onTime, uint8_t offTime, uint8_t cycles)
- void **GUD_displayOff** ()
- void **GUD_displayOn** ()
- void **GUD_screenSaver** (uint8_t mode)
- void **GUD_setFontStyle** (uint8_t proportional, uint8_t evenSpacing)
- void **GUD_setFontSize** (uint8_t x, uint8_t y, uint8_t tall)
- void **GUD_selectWindow** (uint8_t window)
- void **GUD_defineWindow** (uint8_t window, unsigned x, unsigned y, unsigned width, unsigned height)
- void **GUD_deleteWindow** (uint8_t window)
- void **GUD_joinScreens** ()
- void **GUD_separateScreens** ()
- int **min** (int x, int y)
- void **GUD_fillRect** (unsigned x0, unsigned y0, unsigned x1, unsigned y1, uint8_t on)
- void **GUD_drawImage** (unsigned width, uint8_t height, uint8_t *data)
- void **GUD_drawDotUnitImage** (unsigned x, uint8_t y, unsigned width, uint8_t height, uint8_t *data)
- void **GUD_printDotUnitChar** (unsigned x, uint8_t y, uint8_t *buffer, uint8_t len)
- void **GUD_FROMImageDefinition** (uint8_t aL, uint8_t aH, uint8_t aE, unsigned length, uint8_t lE, uint8_t *data)
- void **GUD_drawFROMImage** (unsigned x, unsigned y, uint8_t memory, uint8_t aL, uint8_t aH, uint8_t aE, uint8_t yL, uint8_t yH, unsigned xOffset, unsigned yOffset, unsigned xSize, unsigned ySize)
- void **GUD_enterUserSetupMode** ()
- void **GUD_endUserSetupMode** ()
- void **GUD_I2C_touchStatusReadAll** ()
- void **GUD_I2C_touchStatusRead** (uint8_t switchNum)
- void **GUD_I2C_touchSet** (uint8_t mode)
- void **GUD_I2C_touchLevelRead** ()
- void **GUD_I2C_touchChangeParam** (uint8_t mode, uint8_t value)
- void **GUD_IOPortSetting** (uint8_t portSetting)
- void **GUD_IOPortOutput** (uint8_t portValue)
- void **GUD_IOPortInput** ()

Variables

- unsigned **WIDTH**
- unsigned **HEIGHT**
- unsigned **LINES**
- int **interface**

Macro Definition Documentation

#define ASYNCH 0

#define I2C 2

#define SPI 1

Function Documentation

void GUD_8by16DotMode ()

void GUD_back ()

Performs a backspace on the GUD display.

Returns:

none

void GUD_blinkScreenOff ()

Resets the blink settings on the module to all zeros.

Returns:

none

void GUD_blinkScreenOn (uint8_t *enable*, uint8_t *reverse*, uint8_t *onTime*, uint8_t *offTime*, uint8_t *cycles*)

Blinks the GUD series module screen based on user input parameters.

Parameters:

enable	Enables or disables screen blinking. 0x00 - Normal display 0x01 - Blink display (alternatively Normal and Blank display) 0x02 - Blink display (alternatively Normal and Reverse display)
reverse	Enables or disables the reverse blinking pattern.
onTime	The time that the display stays ON during blinking
offTtime	The time that the displays stays OFF during blinking.

Returns:

none

void GUD_carriageReturn ()

Performs a carriage return on the GUD display.

Returns:

none

void GUD_clearScreen ()

Clears the screen of the GUD module.

Returns:

none

void GUD_crlf ()

Performs a carriage return and line feed on the GUD display.

Returns:

none

void GUD_cursorOff ()

Turns the cursor off.

Returns:

none

void GUD_cursorOn ()

Turns the cursor on.

Returns:

none

void GUD_defineCustomChar (uint8_t *code*, uint8_t *format*, uint8_t * *data*)

Character address starts at 0x20 and ends at 0x27.

- 0 - GUD900 5x7 Format
- 1 - GUD900 7x8 Format
- 2 - CUU Format

To print custom character, write the code of the character to the display after using this function.

Data format for custom characters:

5x7 character

- 0bX0000000
- 0bX0000000
- 0bX0000000
- 0bX0000000
- 0bX0000000

7x8 character

- 0b00000000
- 0b00000000
- 0b00000000
- 0b00000000
- 0b00000000
- 0b00000000
- 0b00000000

- X = don't care
- LSB is the bottom of the character.
- MSB is the top of the character.
- First byte of data is left most column of the character.
- Last byte of data is the right most column of the character.

Parameters:

code	The code for the character being defined.
format	The format of the character being defined.
**data*	The data for the character being defined.

Returns:

none

void GUD_defineWindow (uint8_t *window*, unsigned *x*, unsigned *y*, unsigned *width*, unsigned *height*)

Defines a specific window for the GUD series module.

Parameters:

window	The number of the window to be created. (1-4)

x	The x coordinate of the new window.
y	The y coordinate of the new window.
width	The width of the new window.
height	The height of the new window.

Returns:

none

void GUD_deleteCustomChar (uint8_t *code*)

Deletes a previously defined custom character.

Parameters:

code	Address of the custom character to be deleted.

Returns:

none

void GUD_deleteWindow (uint8_t *window*)

Deletes a previously defined window.

Parameters:

window	The number of the window to be deleted. (1-4)

Returns:

none

void GUD_displayOff ()

Turns the display OFF.

Returns:

none

void GUD_displayOn ()

Turns the display ON.

Returns:

none

void GUD_drawDotUnitImage (unsigned *x*, uint8_t *y*, unsigned *width*, uint8_t *height*, uint8_t * *data*)

Draw a bitmap image on the display. This function is dot specific, not character line specific.

Parameters:

x	The x coordinate for the top left hand corner of the image.
y	The y coordinate for the top left hand corner of the image.
width	The width of the image.
height	The height of the image.
data	The byte data of the image.

Returns:

none

void GUD_drawFROMImage (unsigned *x*, unsigned *y*, uint8_t *memory*, uint8_t *aL*, uint8_t *aH*, uint8_t *aE*, uint8_t *yL*, uint8_t *yH*, unsigned *xOffset*, unsigned *yOffset*, unsigned *xSize*, unsigned *ySize*)

Prints an image that was stored in FROM.

Parameters:

x	The x coordinate of the top left hand corner of the image.
y	The y coordinate of the top left hand corner of the image.
memory	The type of memory the image is stored in.
aL	Bit image data memory address, lower byte.
aH	Bit image data memory address, upper byte.
aE	Bit image data memory address, extension byte.
yL	Bit image defined, Y size, lower byte.
yH	Bit image defined, Y size, upper byte.
xOffset	Image data offset x.
yOffset	Image data offset y.
xSize	Bit image display X size.
ySize	Bit image display Y size.

Returns:

none

void GUD_drawImage (unsigned *width*, uint8_t *height*, uint8_t * *data*)

Draw a bitmap image on the display. This function is character line specific, not dot specific.

Parameters:

width	The width of the bitmap image.
height	The height of the bitmap image.
data	The byte data of the bitmap image.

Returns:

none

void GUD_endUserSetupMode ()

End user setup mode.

Returns:

none

void GUD_enterUserSetupMode ()

Enter user setup mode.

Returns:

none

void GUD_fillRect (unsigned *x0*, unsigned *y0*, unsigned *x1*, unsigned *y1*, uint8_t *on*)

Draw a filled rectangle on the display.

Parameters:

x0	The x coordinate of the top left hand corner of the rectangle.
y0	The y coordinate of the top left hand corner of the rectangle
x1	The x coordinate of the bottom right hand corner of the rectangle.
y1	The y coordinate of the bottom right hand corner of the rectangle.
on	Enable or disable the rectangle drawing.

Returns:

none

void GUD_forward ()

Moves the cursor forward one position on the GUD display.

Returns:

none

void GUD_FROMImageDefinition (uint8_t *aL*, uint8_t *aH*, uint8_t *aE*, unsigned *length*, uint8_t *lE*, uint8_t * *data*)

This function stores an image into a specified address in FROM.

Definable area:

- 0x00000000 <= bit image data memory address <= 0x0007ffff
- 0x00000000 <= bit image data length <= 0x00080000

Parameters:

aL	Bit image data memory address, lower byte
aH	Bit image data memory address, upper byte
aE	Bit image data memory address, extension byte
length	Bit image data length
lE	Bit image data length, extension byte
data	Bit image data

Returns:

none

uint8_t GUD_getColumn (uint8_t * *src*, int *col*)

Helper function for GUD900_defineCustomChar.

Parameters:

src	The source data array to be used.
col	The column of pixels to get.

Returns:

uint8_t The desired column.

void GUD_home ()

Brings the cursor to its home position (top left) on the GUD display.

Returns:

none

void GUD_I2C_touchChangeParam (uint8_t *mode*, uint8_t *value*)

Function to set the touch switch internal parameters.

Settings:

- mode = 0x00 : Touch Sensitivity Level Setting
  - 0x00 <= value <= 0x07
    - value = 0x00 : Threshold value is 12.5%
    - value = 0x01 : Threshold value is 25.0%
    - value = 0x02 : Threshold value is 37.5%
    - value = 0x03 : Threshold value is 50.0%
    - value = 0x04 : Threshold value is 62.5%

    - value = 0x05 : Threshold value is 75.0%
    - value = 0x06 : Threshold value is 87.5%
    - value = 0x07 : Threshold value is 100%
- mode = 0x01 : Sampling Time Setting (ON decision)
    - 0x00 <= value <= 0xfe
- mode = 0x02 : Sampling Time Setting (OFF decision)
    - 0x00 <= value <= 0xfe
- mode = 0x03 : Calibration Period Setting
    - 0x00 <= value <= 0x64

Parameters:

mode	The desired touch setting to change.
value	The value used to change the touch setting.

Returns:

none

void GUD_I2C_touchLevelRead ()

Function to send the response level for all switches.

Module response will be:

1. Identifier - 0x14 - 1 byte
2. Switch number - SwNumMax - 1 byte
3. Information ON/OFF - ResLevel - SwMax bytes

Returns:

none

void GUD_I2C_touchSet (uint8_t *mode*)

Function to set the Touch Switch status read mode

Settings:

- mode = 0x00 : Manual transmit mode (Send only in response to read command)
- mode = 0x01 : Automatic transmit mode 1 (All Touch Switch status)
- mode = 0x02 : Automatic transmit mode 2 (Individual Touch Switch status)

Parameters:

mode	The touch mode to be used.

Returns:

none

void GUD_I2C_touchStatusRead (uint8_t *switchNum*)

Function to show the ON/OFF status for an individual switch.

- 0x00 <= switchNum <= MAX_NUM_SWITCHES
- switchNum = 0x00 : Switch 1
- switchNum = 0x01 : Switch 2
  |
  |
- switchNum = n - 1 : Switch n

Module response will be:

1. Identifier - 0x11 - 1 byte
2. Switch number - 0x00 through SwNumMax - 1 byte
3. Information ON/OFF - 0x00 through 0xff - 1 byte

- 0x00 : Switch OFF
- 0x01 : Switch ON

Parameters:

switchNum	The number of the switch to be read.

Returns:

none

void GUD_I2C_touchStatusReadAll ()

Function to show the ON/OFF status for all touch switches.

Module response will be:

1. Identifier - 0x10 - 1 byte
2. Switch number - InfoDatLen - 1 byte
3. Information ON/OFF - 0x00 through 0xff - InfoDatLen bytes

Returns:

none

void GUD_Init (unsigned *width*, unsigned *height*)

Initializes the Noritake GUD series module.

Parameters:

width	The width of the display in dots.
height	The height of the display in dots.

Returns:

none

void GUD_invertOff ()

Turns the display inversion OFF.

Returns:

none

void GUD_invertOn ()

Turns the display inversion ON.

Returns:

none

void GUD_IOPortInput ()

Function to request the GUD module to read the values present on the GPIO ports.

Returns:

none

void GUD_IOPortOutput (uint8_t *portValue*)

Function to set the output value on the GPIO ports.

Settings:

- 0x00 < output data value < 0x0f

Parameters:

portValue	The desired value to put on the port.

Returns:

none

void GUD_IOPortSetting (uint8_t *portSetting*)

Function to set the GPIO ports on the GUD unit to be inputs or outputs.

Settings:

- portSetting = 0x00 (all input) to 0x0f (all output)
- The portSetting parameter is bit-wise to the ports present on the module.

Parameters:

portSetting	The desired port setting, input or output.

Returns:

none

void GUD_joinScreens ()

Joins all of the screens into one base screen.

Returns:

none

void GUD_lineFeed ()

Performs a line feed on the GUD display.

Returns:

none

void GUD_print (char *data*)

The universal print function for this library. Prints bytes based on the interface value set by each interface's initialization function.

Parameters:

data	The data to be sent.

Returns:

none

void GUD_printDotUnitChar (unsigned *x*, uint8_t *y*, uint8_t * *buffer*, uint8_t *len*)

Print a string of characters at any give XY coordinate on screen.

Parameters:

x	X coordinate of the top left hand corner of the first character.
y	Y coordinate of the top left hand corner of the first character.
buffer	The string of characters to be printed.
len	The length of the passed in string of characters.

Returns:

none

void GUD_reset ()

Resets the GUD series module.

Returns:

none

void GUD_screenSaver (uint8_t *mode*)

Sets the screen saver mode for the GUD series module.

Screen saver modes:

- 0x00 - Display power OFF
- 0x01 - Display power ON
- 0x02 - All dots OFF
- 0x03 - All dots ON
- 0x04 - Repeat blink display with normal and reverse display

Parameters:

mode	The desired screensaver mode. (see ScreenSaver)

Returns:

none

void GUD_scrollScreen (unsigned *x*, unsigned *y*, unsigned *times*, uint8_t *speed*)

Scrolls the screen based on these parameters.

Parameters:

x	The amount of pixels for the screen to scroll horizontally.
y	The amount of pixels for the screen to scroll vertically.
times	The number of times the screen will scroll?
speed	The speed at which scrolling occurs.

Returns:

none

void GUD_selectWindow (uint8_t *window*)

Selects the current window.

Parameters:

window	The desired window to select. (0-4)

Returns:

none

void GUD_separateScreens ()

Separates all of the screens into four separate screens.

Returns:

none

void GUD_setAsciiVariant (uint8_t *code*)

Sets the desired ASCII variant to be used. Also called "International font select" in the GUD software manual.

Font codes:

- 0x00 - America
- 0x01 - France
- 0x02 - Germany
- 0x03 - England

- 0x04 - Denmark 1
- 0x05 - Sweden
- 0x06 - Italy
- 0x07 - Spain 1
- 0x08 - Japan
- 0x09 - Norway
- 0x0A - Denmark 2
- 0x0B - Spain 2
- 0x0C - Latin America
- 0x0D - Korea

Parameters:

code	The code of the ASCII variant to be used.

Returns:

none

void GUD_setCharSet (uint8_t *code*)

Sets the desired character set.

Character set codes:

- 0x00 - PC437(USA-Euro std)
- 0x01 - Katakana - Japanese
- 0x02 - PC850 (Multilingual)
- 0x03 - PC860 (Portuguese)
- 0x04 - PC863 (Canadian-French)
- 0x05 - PC865 (Nordic)
- 0x10 - WPC1252 (Latin)
- 0x11 - PC866 (Cryllic #2)
- 0x12 - PC852 (Latin 2)
- 0x13 - PC858 (Eastern European)

Parameters:

code	The code for the desired character set.

Returns:

none

void GUD_setCompositionMode (uint8_t *mode*)

Sets the composition mode of the GUD module.

Composition modes:

- 0x00 - Normal display write (not mixture display)
- 0x01 - OR display write
- 0x02 - AND display write
- 0x03 - XOR display write

Parameters:

mode	The desired composition mode.

Returns:

none

void GUD_setCursor (unsigned *x*, unsigned *y*)

Moves the cursor to a specific location on the GUD module.

Parameters:

x	The desired x coordinate.
y	The desired y coordinate.

Returns:

none

void GUD_setFontSize (uint8_t *x*, uint8_t *y*, uint8_t *tall*)

Sets the font size for the GUD series module.

Parameters:

x	
y	
tall	

Returns:

none

void GUD_setFontStyle (uint8_t *proportional*, uint8_t *evenSpacing*)

Sets the font style for the GUD series module.

Parameters:

proportional	Enable or disable a proportional font style.
evenSpacing	Enable or disable even spacing between characters.

Returns:

none

void GUD_setHorizScrollSpeed (uint8_t *speed*)

Sets the horizontal scroll speed on the GUD series module.

Speed parameter specifics:

- 0x00 <= speed <= 0x1f
- speed = 0x00 : Instantaneous Speed
- speed = 0x01 : IntTime / 2 dots
- speed = 0x02 - 0x1f : (n-1) * IntTime / dot

Parameters:

speed	The desired horizontal scroll speed.

Returns:

none

void GUD_setMultiByteCharSet (uint8_t *code*)

Sets the multiple byte character set to be used.

Code for each character set:

- 0x00 - Japanese
- 0x01 - Korean
- 0x02 - Simplified Chinese
- 0x03 - Traditional Chinese

Parameters:

code	The character set code to be used.

Returns:

none

void GUD_setScreenBrightness (unsigned *level*)

Sets the screen brightness of the GUD module.

Screen brightness levels:

- 0x01 - 12.5%
- 0x02 - 25.0%
- 0x03 - 37.5%
- 0x04 - 50.0%
- 0x05 - 62.5%
- 0x06 - 75.0%
- 0x07 - 87.5%
- 0x08 - 100%

Parameters:

level	The desired brightness level.

Returns:

none

void GUD_setScrollMode (uint8_t *mode*)

Sets the scroll mode on the GUD module.

Scroll modes:

- 0x01 - Wrapping mode
- 0x02 - Vertical scroll mode
- 0x03 - Horizontal scroll mode

Parameters:

mode	The desired scroll mode.

Returns:

none

void GUD_usCommand ()

Returns:

none

void GUD_useCustomChars (uint8_t *enable*)

Enables or disables the use of custom characters.

Parameters:

enable	Enable of disable custom characters.

Returns:

none

void GUD_useMultiByteChars (uint8_t *enable*)

Enables the use of multiple-byte characters.

Parameters:

enable	Enable or disable multiple byte characters.

Returns:

none

void GUD_wait (uint8_t *wait*)

Suspends the program and waits for a user defined amount of time.

Parameters:

wait	The amount of time for the MCU to wait.

Returns:

none

void GUD_XY (unsigned *x*, unsigned *y*)

Sends an x and y coordinate to the GUD display.

Parameters:

x	The desired x coordinate.
y	The desired y coordinate.

Returns:

none

void GUD_XY1 (unsigned *x*, unsigned *y*)

Sends an x and y coordinate to the GUD display.

Parameters:

x	The desired x coordinate.
y	The desired y coordinate.

Returns:

none

int min (int *x*, int *y*)

Calculate the minimum value between two given values.

Parameters:

x	Value 1.
y	Value 2.

Returns:

The smaller integer.

Variable Documentation

unsigned HEIGHT

int interface

unsigned LINES

unsigned WIDTH

PROJ_TEST_2/GUD_I2C.c File Reference

```
#include "GUD_I2C.h"
#include <stddef.h>
```

Functions

- void **GUD_I2C_Init** (uint8_t GUDaddress)
- void **GUD_I2C_Start** (uint8_t direction)
- uint8_t **GUD_I2C_SingleReadAck** ()
- uint8_t **GUD_I2C_ReadAck** ()
- uint8_t **GUD_I2C_SingleReadNack** ()
- uint8_t **GUD_I2C_ReadNack** ()
- void **GUD_I2C_MultiRead** (uint8_t *data, int count)
- void **GUD_I2C_MultiReadTRDY** (uint8_t *data)
- void **GUD_I2C_Write** (uint8_t data)
- void **GUD_I2C_MultiWrite** (uint8_t *data)
- void **GUD_I2C_Stop** ()

Variables

- uint8_t **address**

Function Documentation

void GUD_I2C_Init (uint8_t *GUDaddress*)

Initializes I2C communication for STM32F4xx to work with Noritake GUD series modules with the I2C daughter board.

Parameters:

GUDaddress	The I2C address for the GUD module.

Returns:

void

void GUD_I2C_MultiRead (uint8_t * *data*, int *count*)

Reads multiple bytes from the selected slave device. The number of bytes read is set by parameters.

Parameters:

data	The data storage location for read data.
count	The number of bytes to be read.

Returns:

none

void GUD_I2C_MultiReadTRDY (uint8_t * *data*)

Reads multiple bytes from the selected slave device. This function reads bytes based on the /TRDY signal.

Parameters:

data	The data storage location for read data.

Returns:

none

void GUD_I2C_MultiWrite (uint8_t * *data*)

Writes a string to the selected slave device. The number of byte transmitted is determined by parameters.

Parameters:

data	The data storage location for read data.

Returns:

none

uint8_t GUD_I2C_ReadAck ()

Reads a single byte from the selected slave device and then sends an acknowledge to request another byte of data. This function does not start the I2C communication. It is meant to be used in succession.

Returns:

read data

uint8_t GUD_I2C_ReadNack ()

Reads a single byte from the selected slave device and does not request another byte of data. This function does not start the I2C communication. It is meant to be used in MultiRead.

Returns:

read data

uint8_t GUD_I2C_SingleReadAck ()

Reads a single byte from the selected slave device and then sends an acknowledge to request another byte of data.

Returns:

read data

uint8_t GUD_I2C_SingleReadNack ()

Reads a single byte from the selected slave device and does not request another byte of data.

Returns:

read data

void GUD_I2C_Start (uint8_t *direction*)

Starts the I2C communication on STM32F4xx.

Parameters:

direction	The master's communication direction.

Returns:

void

void GUD_I2C_Stop ()

Stops the I2C communication.

Returns:

none

void GUD_I2C_Write (uint8_t *data*)

Writes a byte to the selected slave device.

Parameters:

data	Data to be written to the device.

Returns:

none

Variable Documentation

uint8_t address

PROJ_TEST_2/GUD_SPI.c File Reference

```
#include "GUD_SPI.h"
#include "stdint.h"
#include <stddef.h>
```

Functions

- void **Delay** (__IO uint32_t time)
- void **GUD_SPI_Init** ()
- void **startSPI** ()
- void **endSPI** ()
- void **initGUD** ()
- void **writeSPI** (uint8_t data)
- void **writeSPI2** (uint8_t data)
- uint8_t **readSPI** ()
- void **writeString** (char *data)
- uint8_t **readSPIstatus** ()
- void **startSPIRead** ()
- uint8_t **readSPIData** ()

Variables

- __IO uint32_t **timingDelay**
  This function handles SysTick Handler.

Function Documentation

void Delay (__IO uint32_t *time*)

void endSPI ()

End SPI communication by raising CS.

Returns:

none

void GUD_SPI_Init ()

Initialize SPI communication.

Returns:

none

void initGUD ()

Initialize the GUD unit.

Returns:

none

uint8_t readSPI ()

Read a byte from the display.

Returns:

The data byte that was read.

uint8_t readSPIData ()

Reads a byte of data from the display. This function should be used after **startSPIRead()** has been called.

Returns:

The data byte that was read.

uint8_t readSPIstatus ()

Read the status byte from the display. This is required to read data. The status data indicates the number of bytes to be read.

Returns:

none

void startSPI ()

Start SPI communication by lowering CS and sending 0x44.

Returns:

none

void startSPIRead ()

Start the read sequence by lowering CS and transmitting 0x54.

Returns:

none

void writeSPI (uint8_t *data*)

Write a byte to the display.

Parameters:

data	The data to be written.

Returns:

none

void writeSPI2 (uint8_t *data*)

Write a byte to the display.

This function does not check to see if MBUSY goes high

Parameters:

data	The data to be written.

Returns:

none

void writeString (char * *data*)

Write an array of bytes to the display.

Parameters:

data	The data to be written.

Returns:

none

PROJ_TEST_2/GUD_USART.c File Reference

```
#include "GUD_USART.h"
#include "stdint.h"
#include <stddef.h>
```

Functions

- void **GUD_UART_Init** (uint32_t baudrate)
  UART library for STM32F4xx and Noritake GU-D Series Modules.
- void **writeUART** (uint8_t data)
- void **printUART** (char *data)
- uint8_t **readUART** ()
- uint8_t **receivedUART** ()
- void **setHBUSY** ()
- void **clearHBUSY** ()

Function Documentation

void clearHBUSY ()

Clear the HBUSY signal.

Returns:

none

void GUD_UART_Init (uint32_t *baudrate*)

Initialize UART communication.

Parameters:

baudrate	The desired baudrate for UART communication.

Returns:

none

void printUART (char * *data*)

Write an array of bytes to the display.

Parameters:

data	The data to be written.

Returns:

none

uint8_t readUART ()

Read a byte from the display.

Returns:

The byte read from the display.

uint8_t receivedUART ()

Indicate if a byte has been received from the module.

Returns:

1 if a byte has been received, 0 otherwise.

void setHBUSY ()

Set the HBUSY signal.

Returns:

none

void writeUART (uint8_t *data*)

Write a byte to the display.

Parameters:

data	The data to be written.

Returns:

none

Revision Note

Manual Number	Date	Revision
ME-N55-0	7/14/2016	Initial release.