GT-CP STM32f4Discovery Code Library
Reference Manual

Noritake Co., Inc.
Cody Johnson
Version 1.1
Date of Issue: 8/16/16 (V1.0)
Revision: 8/8/18 (V1.1)

File Index

File List

File Documentation

GTCP.c File Reference

```
#include "GTCP.h"
#include "GTCP_I2C.h"
#include "GTCP_SPI.h"
#include "GTCP_USART.h"
#include <stddef.h>
```

Macros

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

Functions

- void **GTCP_Init** (unsigned width, unsigned height)
- void **GTCP_print** (char data)
- void **GTCP_back** ()
- void **GTCP_forward** ()
- void **GTCP_lineFeed** ()
- void **GTCP_home** ()
- void **GTCP_carriageReturn** ()
- void **GTCP_crlf** ()
- void **GTCP_XY** (unsigned x, unsigned y)
- void **GTCP_XY1** (unsigned x, unsigned y)
- void **GTCP_usCommand** ()
- void **GTCP_setCursor** (unsigned x, unsigned y)
- void **GTCP_clearScreen** ()
- void **GTCP_lineClear** ()
- void **GTCP_lineEndClear** ()
- void **GTCP_cursorOn** ()
- void **GTCP_cursorOff** ()
- void **GTCP_reset** ()
- void **GTCP_8by16DotMode** ()
- void **GTCP_writeScreenMode** (uint8_t mode)
- void **GTCP_writeMixtureDisplayMode** (uint8_t mode)
- void **GTCP_useMultiByteChars** (uint8_t enable)
- void **GTCP_setMultiByteCharSet** (uint8_t code)
- void **GTCP_setFontSize** (uint8_t size)
- void **GTCP_setOutlineFontType** (uint8_t fontType)
- void **GTCP_useCustomChars** (uint8_t enable)
- uint8_t **GTCP_getColumn** (uint8_t *src, int col)
- void **GTCP_defineCustomChar** (uint8_t codeStart, uint8_t codeEnd, uint8_t format, uint8_t xDir, uint8_t *data)
- void **GTCP_deleteCustomChar** (uint8_t type, uint8_t code)
- void **GTCP_setAsciiVariant** (uint8_t code)
- void **GTCP_setCharSet** (uint8_t code)
- void **GTCP_setScrollMode** (uint8_t mode)
- void **GTCP_setHorizScrollSpeed** (uint8_t speed)

- void **GTCP_invertOff** ()
- void **GTCP_invertOn** ()
- void **GTCP_setCompositionMode** (uint8_t mode)
- void **GTCP_setScreenBrightness** (unsigned level)
- void **GTCP_wait** (uint8_t wait)
- void **GTCP_shortWait** (uint8_t time)
- void **GTCP_scrollScreen** (unsigned x, unsigned y, unsigned times, uint8_t speed)
- void **GTCP_curtainAction** (uint8_t direction, uint8_t speed, uint8_t red, uint8_t green, uint8_t blue)
- void **GTCP_springAction** (uint8_t direction, uint8_t speed, unsigned x, unsigned y)
- void **GTCP_randomAction** (uint8_t actionType, uint8_t speed, unsigned x, unsigned y)
- void **GTCP_fadeInAction** (uint8_t speed, unsigned x, unsigned y)
- void **GTCP_fadeOutAction** (uint8_t speed)
- void **GTCP_blinkScreenOff** ()
- void **GTCP_blinkScreenOn** (uint8_t enable, uint8_t reverse, uint8_t onTime, uint8_t offTime, uint8_t cycles)
- void **GTCP_displayOff** ()
- void **GTCP_displayOn** ()
- void **GTCP_screenSaver** (uint8_t mode)
- void **GTCP_setFontStyle** (uint8_t proportional, uint8_t evenSpacing)
- void **GTCP_setFontMagnification** (uint8_t x, uint8_t y, uint8_t tall)
- void **GTCP_selectWindow** (uint8_t window)
- void **GTCP_defineWindow** (uint8_t window, unsigned x, unsigned y, unsigned width, unsigned height)
- void **GTCP_deleteWindow** (uint8_t window)
- void **GTCP_joinScreens** ()
- void **GTCP_separateScreens** ()
- void **GTCP_drawImage** (unsigned width, uint8_t height, uint8_t format, uint8_t *data)
- void **GTCP_pixelDrawing** (uint8_t pen, unsigned x, unsigned y)
- void **GTCP_drawLineBox** (uint8_t mode, uint8_t pen, unsigned xStart, unsigned yStart, unsigned xEnd, unsigned yEnd)
- void **GTCP_storedBitImageDraw** (uint8_t memory, unsigned **address**, unsigned extension, unsigned xDefine, unsigned xSize, unsigned ySize, uint8_t format)
- void **GTCP_setCharacterStyle** (uint8_t style)
- void **GTCP_setCharacterColor** (uint8_t red, uint8_t green, uint8_t blue)
- void **GTCP_setBackgroundColor** (uint8_t red, uint8_t green, uint8_t blue)
- void **GTCP_enterUserSetupMode** ()
- void **GTCP_endUserSetupMode** ()
- void **GTCP_touchDataEnable** (uint8_t enable)
- void **GTCP_touchChannel** (uint8_t channel)
- void **GTCP_touchModeSelect** (uint8_t mode)
- void **GTCP_coordinatesMode** (uint8_t channel)
- void **GTCP_switchMatrixMode** (uint8_t channel, uint8_t switchesX, uint8_t switchesY, uint8_t clearanceX, uint8_t clearanceY)
- void **GTCP_IOPortSetting** (uint8_t portSetting)
- void **GTCP_IOPortOutput** (uint8_t portValue)
- void **GTCP_IOPortInput** ()
- void **GTCP_MemorySWSetting** (uint8_t memorySW, uint8_t data)
- void **GTCP_MemorySWSettingMulti** (uint8_t numSettings, uint8_t *data)
- void **GTCP_MemorySWDataSend** (uint8_t switchNum)
- void **GTCP_MemorySWDataSendMulti** (uint8_t numReads, uint8_t *data)
- void **GTCP_16x16CharacterDefinition** (uint8_t codeUpper, uint8_t codeLower, uint8_t *data)
- void **GTCP_16x16CharacterDelete** (uint8_t codeUpper, uint8_t codeLower)
- void **GTCP_32x32CharacterDefinition** (uint8_t codeUpper, uint8_t codeLower, uint8_t *data)
- void **GTCP_32x32CharacterDelete** (uint8_t codeUpper, uint8_t codeLower)
- void **GTCP_downloadCharacterSave** (uint8_t fontSize)

- void **GTCP_downloadCharacterRestore** (uint8_t fontSize)
- void **GTCP_FROMUserFontDefinition** (uint8_t table, uint8_t *data)
- void **GTCP_RAMMacroDefineDelete** (uint8_t lengthUpper, uint8_t lengthLower, uint8_t *data)
- void **GTCP_FROMMacroDefineDelete** (uint8_t registrationNum, uint8_t lengthUpper, uint8_t lengthLower, uint8_t interval, uint8_t idleTime, uint8_t *data)
- void **GTCP_macroExecution** (uint8_t definitionNum, uint8_t interval, uint8_t idleTime)
- void **GTCP_macroEndCondition** (uint8_t endCodeEnable, uint8_t endCode, uint8_t endScreenSetting)
- void **GTCP_memoryStore** (uint8_t sizeUpper, uint8_t sizeLower, uint8_t sizeExtension, uint8_t memorySelect, uint8_t addressUpper, uint8_t addressLower, uint8_t addressExtension, uint8_t *data)
- void **GTCP_memoryTransfer** (uint8_t sizeUpper, uint8_t sizeLower, uint8_t sizeExtension, uint8_t destMemory, uint8_t destAddressUpper, uint8_t destAddressLower, uint8_t destAddressExtension, uint8_t srcMemory, uint8_t srcAddressUpper, uint8_t srcAddressLower, uint8_t srcAddressExtension)
- void **GTCP_memorySend** (uint8_t sizeUpper, uint8_t sizeLower, uint8_t sizeExtension, uint8_t memorySelect, uint8_t addressUpper, uint8_t addressLower, uint8_t addressExtension)
- void **GTCP_displayStatusSend** (uint8_t statusType, uint8_t **address**, uint8_t length)
- void **GTCP_touchSettingPackageDataStore** (uint8_t packageNum, uint8_t *data)
- void **GTCP_touchSettingPackageSelection** (uint8_t packageNum)

Variables

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

Macro Definition Documentation

#define ASYNCH 0

#define I2C 2

#define SPI 1

Function Documentation

void GTCP_16x16CharacterDefinition (uint8_t *codeUpper*, uint8_t *codeLower*, uint8_t * *data*)

Defines a 16x16 pixel downloaded character (2-byte character) in the code specified by codeUpper and codeLower.

codeUpper, codeLower: Depends on currently selected language.

- codeUpper = 0xec && 0x40 <= codeLower <= 0x4f : Japanese - JIS X0208 (SHIFT-JIS)
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Korean - KSC5601-87
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Simplified Chinese - GB2312-80
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Traditional Chinese - Big-5

Parameters:

codeUpper	Character code, upper byte
codeLower	Character code, lower byte
data	Character definition data

Returns:

none

void GTCP_16x16CharacterDelete (uint8_t *codeUpper*, uint8_t *codeLower*)

Deletes a defined 16x16 download character in code specified by codeUpper and codeLower.

codeUpper, codeLower: Depends on currently selected language.

- codeUpper = 0xec && 0x40 <= codeLower <= 0x4f : Japanese - JIS X0208 (SHIFT-JIS)
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Korean - KSC5601-87
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Simplified Chinese - GB2312-80
- codeUpper = 0xfe && 0xa1 <= codeLower <= 0xb0 : Traditional Chinese - Big-5

Parameters:

codeUpper	Character code, upper byte
codeLower	Character code, lower byte

Returns:

none

void GTCP_32x32CharacterDefinition (uint8_t *codeUpper*, uint8_t *codeLower*, uint8_t * *data*)

Defines a 32x32 pixel downloaded character (2-byte character) in character code specified by codeUpper and codeLower.

- codeUpper, codeLower: Depends on currently selected language.
- codeUpper = 0xec && 0x40 <= codeLower <= 0x4f : Japanese - JIS X0208 (SHIFT-JIS)

Parameters:

codeUpper	Character code, upper byte
codeLower	Character code, lower byte
data	Character definition data.

Returns:

none

void GTCP_32x32CharacterDelete (uint8_t *codeUpper*, uint8_t *codeLower*)

Delete a defined 32x32 download character in code specified by codeUpper and codeLower. This command is invalid if Japanese is not the selected language.

- codeUpper, codeLower: Depends on currently selected language.
- codeUpper = 0xec && 0x40 <= codeLower <= 0x4f : Japanese - JIS X0208 (SHIFT-JIS)

Parameters:

codeUpper	Character code, upper byte
codeLower	Character code, lower byte

Returns:

none

void GTCP_8by16DotMode ()

void GTCP_back ()

Performs a backspace on the GTCP display.

Returns:

none

void GTCP_blinkScreenOff ()

Resets the blink settings on the module to all zeros.

Returns:

none

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

Blinks the GTCP 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 GTCP_carriageReturn ()

Performs a carriage return on the GTCP display.

Returns:

none

void GTCP_clearScreen ()

Clears the screen of the GTCP module.

Returns:

none

void GTCP_coordinatesMode (uint8_t *channel*)

Set the coordinate touch mode.

- 0x00 <= channel <= 0x03

Parameters:

channel	The desired touch data channel.

Returns:

none

void GTCP_crlf ()

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

Returns:

none

void GTCP_cursorOff ()

Turns the cursor off.

Returns:

none

void GTCP_cursorOn ()

Turns the cursor on.

Returns:

none

void GTCP_curtainAction (uint8_t *direction*, uint8_t *speed*, uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Start a curtain display action on-screen.

- direction = 0x00: To the right from the left edge.
- direction = 0x01: To the left from the right edge.
- direction = 0x02: To the left and right separately from the center.
- direction = 0x03: To the center from the left and right edge.

- Curtain action speed = 60 * speed * IntTime
- 0x00 <= speed <= 0xff

- 0x00 <= red <= 0xff
- 0x00 <= green <= 0xff
- 0x00 <= blue <= 0xff

Parameters:

direction	Direction of curtain action.
speed	Curtain action speed.
red	Red brightness
green	Green brightness
blue	Blue brightness

Returns:

none

void GTCP_defineCustomChar (uint8_t *codeStart*, uint8_t *codeEnd*, uint8_t *format*, uint8_t *xDir*, uint8_t * *data*)

code starts at 0x20 and ends at 0x27

- format = 0x01: 6x8 pixel character
- format = 0x02: 8x16 pixel character
- format = 0x03: 12x24 pixel character
- format = 0x04: 16x32 pixel character

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

Data format for custom characters: 6x8 character

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

8x16 character

- 0b00000000, 0b00000000

- 0b00000000, 0b00000000
- 0b00000000, 0b00000000
- 0b00000000, 0b00000000
- 0b00000000, 0b00000000
- 0b00000000, 0b00000000
- 0b00000000, 0b00000000
- 0b00000000, 0b00000000

12x24 character

- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000

16x32 character

- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 0b00000000, 0b00000000, 0b00000000
- 0b00000000, 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 GTCP_defineWindow (uint8_t *window*, unsigned *x*, unsigned *y*, unsigned *width*, unsigned *height*)

Defines a specific window for the GTCP 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 GTCP_deleteCustomChar (uint8_t *type*, uint8_t *code*)

Deletes a previously defined custom character.

Parameters:

code	Address of the custom character to be deleted.

Returns:

none

void GTCP_deleteWindow (uint8_t *window*)

Deletes a previously defined window.

Parameters:

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

Returns:

none

void GTCP_displayOff ()

Turns the display OFF.

Returns:

none

void GTCP_displayOn ()

Turns the display ON.

Returns:

none

void GTCP_displayStatusSend (uint8_t *statusType*, uint8_t *address*, uint8_t *length*)

Tell the display to send the desired status information.

- statusType = 0x01: Boot version information (address, length not used)
- statusType = 0x02: Firmware version information (address, length not used)
- statusType = 0x10: 2-byte character code information (address, length not used)
- statusType = 0x11: Language type information (address, length not used)
- statusType = 0x20: Memory checksum information
  - 0x00 <= address <= 0xff: Start address (Effective address = address * 0x10000)
  - 0x00 <= length <= 0xff: Data length (Effective data length = length * 0x10000)
- statusType = 0x30: Product type information (address, length not used)

- statusType = 0x40: Display x pixel information (address, length not used)
- statusType = 0x41: Display y pixel information (address, length not used)

The following data is transmitted:

1. Header | 0x28 | 1 byte
2. Identifier | 0x65 | 1 byte
3. Identifier | 0x40 | 1 byte
4. Data | 0x00-0xff | statusType = 0x01: 4 bytes
   statusType = 0x02: 4 bytes
   statusType = 0x10: 15 bytes
   statusType = 0x11: 15 bytes
   statusType = 0x20: 4 bytes
   statusType = 0x30: 15 bytes
   statusType = 0x40: 3 bytes
   statusType = 0x41: 3 bytes

Parameters:

statusType	The desired module status to read.
address	Checksum address (optional)
address	Length of data (optional)

Returns:

none

void GTCP_downloadCharacterRestore (uint8_t *fontSize*)

Transfer the downloaded characters saved in FROM to RAM.

- fontSize = 0x01: 6x8 pixels
- fontSize = 0x02: 8x16 pixels
- fontSize = 0x03: 16x16 pixels
- fontSize = 0x04: 16x32 pixels
- fontSize = 0x05: 32x32 pixels
- fontSize = 0x06: 12x24 pixels

Parameters:

fontSize	The desired character font to be restored.

Returns:

none

void GTCP_downloadCharacterSave (uint8_t *fontSize*)

Save the download characters defined in RAM to FROM (RAM -> FROM)

- fontSize = 0x01: 6x8 pixels
- fontSize = 0x02: 8x16 pixels
- fontSize = 0x03: 16x16 pixels
- fontSize = 0x04: 16x32 pixels
- fontSize = 0x05: 32x32 pixels
- fontSize = 0x06: 12x24 pixels

Parameters:

fontSize	The desired character font size to be saved.

Returns:

none

void GTCP_drawImage (unsigned *width*, uint8_t *height*, uint8_t *format*, uint8_t * *data*)

Real-time bit image display.

- format = 0x81: Monochrome (1-bit) format
- format = 0x86: Color 6-bit format
- format = 0x8c: Color 12-bit format
- format = 0x90: Color 16-bit format
- format = 0x98: Color 24-bit format
- format = 0xf0: BMP

Parameters:

width	The width of the image to display.
height	The height of the image to display.
format	The desired image format.
data	The raw data of the image to display.

Returns:

none

void GTCP_drawLineBox (uint8_t *mode*, uint8_t *pen*, unsigned *xStart*, unsigned *yStart*, unsigned *xEnd*, unsigned *yEnd*)

Draw a line or box on the display.

- mode = 0x00: Line
- mode = 0x01: Box
- mode = 0x02: Box Fill

- pen = 0x00: Pen color is set to the background color.
- pen = 0x01: Pen color is set to the character color.

Parameters:

mode	The desired drawing mode.
pen	Set pen color to character or background color.
xStart	Line/Box drawing x-axis start position.
yStart	Line/Box drawing y-axis start position.
xEnd	Line/Box drawing x-axis end position.
yEnd	Line/Box drawing y-axis end position.

Returns:

none

void GTCP_endUserSetupMode ()

End user setup mode.

Returns:

none

void GTCP_enterUserSetupMode ()

Enter user setup mode.

Returns:

none

void GTCP_fadeInAction (uint8_t *speed*, unsigned *x*, unsigned *y*)

Start a fade-in display action.

- Fade-in time = speed * approximately 0.5s

Parameters:

speed	Fade-in action speed.
x	Display memory X position.
y	Display memory Y position.

Returns:

none

void GTCP_fadeOutAction (uint8_t *speed*)

Start a fade-out display action.

- Fade-out time = speed * approximately 0.5s

Parameters:

speed	Fade-out action speed.

Returns:

none

void GTCP_forward ()

Moves the cursor forward one position on the GTCP display.

Returns:

none

void GTCP_FROMMacroDefineDelete (uint8_t *registrationNum*, uint8_t *lengthUpper*, uint8_t *lengthLower*, uint8_t *interval*, uint8_t *idleTime*, uint8_t * *data*)

Define or delete FROM Macro or FROM Program Macro.

- 0x01 <= registrationNum <= 0x04: FROM Macro number 1-4
- 0x0000 <= (lengthLower + lengthUpper * 0x0100) <= 0x2000 (if using 4 Macros), 0x8000 (if using 1 Macro)
- (lengthLower + lengthUpper * 0x0100) > 0: Supplied data is stored as a Macro
- (lengthLower + lengthUpper * 0x0100) = 0: Macro is deleted.
- 0x00 <= interval <= 0xff
- 0x00 <= idleTime <= 0xff
- 0x00 <= data <= 0xff

Parameters:

registrationNum	FROM Macro registration number
lengthUpper	FROM Macro data length, upper byte
lengthLower	FROM Macro data length, lower byte
interval	Display time interval (interval * IntTime)
idleTime	Idle time for macro repetition (idleTime * IntTime)
data	FROM Macro data

Returns:

none

void GTCP_FROMUserFontDefinition (uint8_t *table*, uint8_t * *data*)

Define the user font table for a specified character size.

- table = 0x01: 6x8 pixel user table
  - P(0x80 - 1) ... P(0x80 - 6) ... P(0xff - 6)
  - 6 Bytes / font * 128 characters (768 bytes)
- table = 0x02: 8x16 pixel user table
  - P(0x80 - 1) ... P(0x80 - 16) ... P(0xff - 16)

    - 16 Bytes / font * 128 characters (2048 bytes)
- table = 0x03: 12x24 pixel user table
    - P(0x80 - 1) ... P(0x80 - 36) ... P(0xff - 36)
    - 36 Bytes / font * 128 characters (4608 bytes)
- table = 0x04: 16x32 pixel user table
    - P(0x80 - 1) ... P(0x80 - 64) ... P(0xff - 64)
    - 64 Bytes / font * 128 characters (8192 bytes)

It is not possible to only define a part of the character code space.

ONLY VALID IN USER SETUP MODE.

Parameters:

table	The desired user table font size.
data	The user table data.

Returns:

none

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

Helper function for GTCP_defineCustomChar.

Parameters:

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

Returns:

uint8_t The desired column.

void GTCP_home ()

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

Returns:

none

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

Initializes the Noritake GTCP series module.

Parameters:

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

Returns:

none

void GTCP_invertOff ()

Turns the display inversion OFF.

Returns:

none

void GTCP_invertOn ()

Turns the display inversion ON.

Returns:

none

void GTCP_IOPortInput ()

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

Returns:

none

void GTCP_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 GTCP_IOPortSetting (uint8_t *portSetting*)

Function to set the GPIO ports on the GTCP 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 GTCP_joinScreens ()

Joins all of the screens into one base screen.

Returns:

none

void GTCP_lineClear ()

Clear the current line. Cursor will return to left end.

Returns:

none

void GTCP_lineEndClear ()

Clear the current line. Cursor will move to right end.

Returns:

none

void GTCP_lineFeed ()

Performs a line feed on the GTCP display.

Returns:

none

void GTCP_macroEndCondition (uint8_t *endCodeEnable*, uint8_t *endCode*, uint8_t *endScreenSetting*)

Macro end condition set.

- endCodeEnable = 0x00: Macro end code disabled
- endCodeEnable = 0x01: Macro end code enabled
- 0x00 <= endCode <= 0xff
- endScreenSetting = 0x00: Clear screen at Macro end
- endScreenSetting = 0x01: Do not clear screen at Macro end

Parameters:

endCodeEnable	Macro end code Enable/Disable
endCode	Macro end code
endScreenSetting	Macro end clear screen setting

Returns:

none

void GTCP_macroExecution (uint8_t *definitionNum*, uint8_t *interval*, uint8_t *idleTime*)

Continuously execute contents of defined Macro 'definitionNum'.

- 0x00 <= definitionNum <= 0x04, 0x80 <= definitionNum <= 0x84
- definitionNum = 0x00: RAM Macro 0
  - 0x01 <= definitionNum <= 0x04: FROM Macro 1-4
- definitionNum = 0x80: RAM Program Macro 0
  - 0x81 <= definitionNum <= 0x84: FROM Program Macro 1-4
- 0x00 <= interval <= 0xff
- 0x00 <= idleTime <= 0xff

Parameters:

definitionNum	Macro processing definition number
interval	Display time interval (interval * IntTime)
idleTime	Idle time for Macro repetition (idleTime * IntTime)

Returns:

none

void GTCP_memorySend (uint8_t *sizeUpper*, uint8_t *sizeLower*, uint8_t *sizeExtension*, uint8_t *memorySelect*, uint8_t *addressUpper*, uint8_t *addressLower*, uint8_t *addressExtension*)

Send data stored in general-purpose memory.

- 0x10 <= memorySelect <= 0x17, memorySelect = 0x30, 0x31
- memorySelect = 0x10: FROM2 base address 0x0000.0000
- memorySelect = 0x11: FROM2 base address 0x0100.0000
- memorySelect = 0x12: FROM2 base address 0x0200.0000
- memorySelect = 0x13: FROM2 base address 0x0300.0000
- memorySelect = 0x14: FROM2 base address 0x0400.0000
- memorySelect = 0x15: FROM2 base address 0x0500.0000
- memorySelect = 0x16: FROM2 base address 0x0600.0000
- memorySelect = 0x17: FROM2 base address 0x0700.0000
- memorySelect = 0x18: FROM2 base address 0x0800.0000
- memorySelect = 0x19: FROM2 base address 0x0900.0000
- memorySelect = 0x1A: FROM2 base address 0x0A00.0000
- memorySelect = 0x1B: FROM2 base address 0x0B00.0000
- memorySelect = 0x1C: FROM2 base address 0x0C00.0000

- memorySelect = 0x1D: FROM2 base address 0x0D00.0000
- memorySelect = 0x1E: FROM2 base address 0x0E00.0000
- memorySelect = 0x1F: FROM2 base address 0x0F00.0000
  - 0x000000 <= (addressLower + addressUpper * 0x100 + addressExtension * 0x10000) <= 0xFFFFFF
  - 0x000000 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0xFFFFFF
- memorySelect = 0x30 (General-purpose RAM):
  - 0x000001 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0x000400
  - 0x000000 <= (addressLower + addressUpper * 0x100 + addressExtension * 0x10000) <= 0x0003FF
- memorySelect = 0x31 (General-purpose FROM):
  - 0x000001 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0x001000
  - 0x000000 <= (addressLower + addressUpper * 0x100 + addressExtension * 0x10000) <= 0x00FFFF

The following data is transmitted:

1. Header | 0x28 | 1 byte
2. Identifier | 0x65 | 1 byte
3. Identifier | 0x28 | 1 byte
4. Data | 0x00-0xff | n bytes

Parameters:

sizeUpper	Data size, upper byte
sizeLower	Data size, lower byte
sizeExtension	Data size, extension byte
memorySelect	Desired memory select
addressUpper	Memory address, upper byte
addressLower	Memory address, lower byte
addressExtension	Memory address, extension byte

Returns:

none

void GTCP_memoryStore (uint8_t *sizeUpper*, uint8_t *sizeLower*, uint8_t *sizeExtension*, uint8_t *memorySelect*, uint8_t *addressUpper*, uint8_t *addressLower*, uint8_t *addressExtension*, uint8_t * *data*)

Store the supplied data into general-purpose memory / FROM2.

- memorySelect = 0x10: FROM2 base address 0x0000.0000
- memorySelect = 0x11: FROM2 base address 0x0100.0000
- memorySelect = 0x12: FROM2 base address 0x0200.0000
- memorySelect = 0x13: FROM2 base address 0x0300.0000
- memorySelect = 0x14: FROM2 base address 0x0400.0000
- memorySelect = 0x15: FROM2 base address 0x0500.0000
- memorySelect = 0x16: FROM2 base address 0x0600.0000
- memorySelect = 0x17: FROM2 base address 0x0700.0000
- memorySelect = 0x18: FROM2 base address 0x0800.0000
- memorySelect = 0x19: FROM2 base address 0x0900.0000
- memorySelect = 0x1A: FROM2 base address 0x0A00.0000
- memorySelect = 0x1B: FROM2 base address 0x0B00.0000
- memorySelect = 0x1C: FROM2 base address 0x0C00.0000
- memorySelect = 0x1D: FROM2 base address 0x0D00.0000
- memorySelect = 0x1E: FROM2 base address 0x0E00.0000
- memorySelect = 0x1F: FROM2 base address 0x0F00.0000
  - 0x000001 <= (sizeLower + sizeUpper * 0x0100 + sizeExtension * 0x00010000) <= 0x00FFFFFF
  - 0x000000 <= (addressLower + addressUpper * 0x0100 + addressExtension * 0x00010000h) <= 0x00FFFFFF
- memorySelect = 0x30 (General-purpose RAM):

    - 0x000001 <= (sizeLower + sizeUpper * 0x0100 + sizeExtension * 0x00010000) <= 0x00000400
    - 0x000000 <= (addressLower + addressUpper * 0x0100 + addressExtension * 0x00010000) <= 0x000003FF
- memorySelect = 0x31 (General-purpose FROM):
    - 0x000001h <= (sizeLower + sizeUpper * 0x0100h + sizeExtension * 0x00010000) <= 0x00001000
    - 0x000000h <= (addressLower + addressUpper * 0x0100 + addressExtension * 0x00010000) <= 0x0000FFFF
- 0x00 <= data <= 0xFF

Parameters:

sizeUpper	Data size, upper byte
sizeLower	Data size, lower byte
sizeExtension	Data size, extension byte
memorySelect	Desired memory to use.
addressUpper	Memory address, upper byte
addressLower	Memory address, lower byte
addressExtension	Memory address, extension byte
data	Data to store

Returns:

none

void GTCP_MemorySWDataSend (uint8_t *switchNum*)

Send the contents of memory SW data.

The following data is transmitted:

1. Header | 0x28 | 1 byte
2. Identifier 1 | 0x65 | 1 byte
3. Identifier 2 | 0x04 | 1 byte
4. Data | 0x00-0xff | 1 byte / n bytes

Parameters:

switchNum	The memory SW to be read.

Returns:

none

void GTCP_MemorySWDataSendMulti (uint8_t *numReads*, uint8_t * *data*)

Send the contents of memory Sw data.

- 0x01 <= numReads <= 0xff

Parameters:

numReads	The number of reads to perform.
data	The memory SW numbers to be read.

Returns:

none

void GTCP_MemorySWSetting (uint8_t *memorySW*, uint8_t *data*)

Sets a memory switch to the passed in value.

Parameters:

memorySW	The memory SW to be changed.
data	The data to be written to the memory SW.

Returns:

none

void GTCP_MemorySWSettingMulti (uint8_t *numSettings*, uint8_t * *data*)

Set multiple memory switch values.

- 0x01 <= numSettings <= 0xff
- data should be set up like:
- memory SW number [1], setting value [1],
- memory SW number [2], setting value [2],
- ... memory SW number [n], setting value [n]

Parameters:

numSettings	The number of SW settings to be set.
data	The setting data.

Returns:

none

void GTCP_memoryTransfer (uint8_t *sizeUpper*, uint8_t *sizeLower*, uint8_t *sizeExtension*, uint8_t *destMemory*, uint8_t *destAddressUpper*, uint8_t *destAddressLower*, uint8_t *destAddressExtension*, uint8_t *srcMemory*, uint8_t *srcAddressUpper*, uint8_t *srcAddressLower*, uint8_t *srcAddressExtension*)

Transfer data between general-purpose memory / FROM2 areas.

- 0x10 <= destMemory, srcMemory <= 0x17, destMemory, srcMemory = 0x30, 0x31
- destMemory, srcMemory = 0x10: FROM2 base address 0x0000.0000
- destMemory, srcMemory = 0x11: FROM2 base address 0x0100.0000
- destMemory, srcMemory = 0x12: FROM2 base address 0x0200.0000
- destMemory, srcMemory = 0x13: FROM2 base address 0x0300.0000
- destMemory, srcMemory = 0x14: FROM2 base address 0x0400.0000
- destMemory, srcMemory = 0x15: FROM2 base address 0x0500.0000
- destMemory, srcMemory = 0x16: FROM2 base address 0x0600.0000
- destMemory, srcMemory = 0x17: FROM2 base address 0x0700.0000
- destMemory, srcMemory = 0x18: FROM2 base address 0x0800.0000
- destMemory, srcMemory = 0x19: FROM2 base address 0x0900.0000
- destMemory, srcMemory = 0x1A: FROM2 base address 0x0A00.0000
- destMemory, srcMemory = 0x1B: FROM2 base address 0x0B00.0000
- destMemory, srcMemory = 0x1C: FROM2 base address 0x0C00.0000
- destMemory, srcMemory = 0x1D: FROM2 base address 0x0D00.0000
- destMemory, srcMemory = 0x1E: FROM2 base address 0x0E00.0000
- destMemory, srcMemory = 0x1F: FROM2 base address 0x0F00.0000
  - 0x000000 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0xFFFFFF
  - 0x000000 <= (destAddressLower + destAddressUpper * 0x100 + destAddressExtension * 0x10000) <= 0xFFFFFF
  - 0x000000 <= (srcAddressLower + srcAddressUpper * 0x100 + srcAddressExtension * 0x10000) <= 0xFFFFFF
- destMemory, srcMemory = 0x30 (General-purpose RAM):
  - 0x000001 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0x000400
  - 0x000000 <= (destAddressLower + destAddressUpper * 0x100 + destAddressExtension * 0x10000) <= 0x0003FF
  - 0x000000 <= (srcAddressLower + srcAddressUpper * 0x100 + srcAddressExtension * 0x10000) <= 0x0003FF
- destMemory, srcMemory = 0x31 (General-purpose FROM):
  - 0x000001 <= (sizeLower + sizeUpper * 0x100 + sizeExtension * 0x10000) <= 0x001000
  - 0x000000 <= (destAddressLower + destAddressUpper * 0x100h + destAddressExtension * 0x10000) <= 0x00FFFF

- 0x000000 <= (srcAddressLower + srcAddressUpper * 0x100h + srcAddressExtension * 0x10000) <= 0x00FFFF

- 0x00 <= data <= 0xFF

Parameters:

sizeUpper	Transfer data size, upper byte
sizeLower	Transfer data size, lower byte
sizeExtension	Transfer data size, extension byte
destMemory	Destination memory select
destAddressUpper	Destination address, upper byte
destAddressLower	Destination address, lower byte
destAddressExtension	Destination address, extension byte
srcMemory	Source memory select
srcAddressUpper	Source address, upper byte
srcAddressLower	Source address, lower byte
srcAddressExtension	Source address, extension byte

Returns:

none

void GTCP_pixelDrawing (uint8_t *pen*, unsigned *x*, unsigned *y*)

Draw a pixel on the display.

- pen = 0x00: Pen color is set to the background color.
- pen = 0x01: Pen color is set to the character color.

Parameters:

pen	Set pen color to character or background color.
x	Pixel position X.
y	Pixel position Y.

Returns:

none

void GTCP_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 GTCP_RAMMacroDefineDelete (uint8_t *lengthUpper*, uint8_t *lengthLower*, uint8_t * *data*)

Define or delete RAM Macro or RAM Program Macro.

- 0x0000 <= (lengthLower + lengthUpper * 0x0100) <= 0x0400
- (lengthLower + lengthUpper * 0x0100) > 0x0000: Supplied data is stored as Macro
- (lengthLower + lengthUpper * 0x0100) = 0x0000: Macro is deleted

Parameters:

lengthUpper	RAM Macro data length, upper byte
lengthLower	RAM Macro data length, lower byte
data	RAM Macro data

Returns:

none

void GTCP_randomAction (uint8_t *actionType*, uint8_t *speed*, unsigned *x*, unsigned *y*)

Start a checkerboard display action.

- 0x00 <= actionType <= 0x02
- actionType = 0x00: Transition pattern 1
- actionType = 0x01: Transition pattern 2
- actionType = 0x02: Transition pattern 3
- 0x00 <= speed <= 0xff

Parameters:

direction	Direction of curtain action.
speed	Checkerboard action speed.
x	Display memory X position.
y	Display memory y position.

Returns:

none

void GTCP_reset ()

Resets the GTCP series module.

Returns:

none

void GTCP_screenSaver (uint8_t *mode*)

Sets the screen saver mode for the GTCP 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 GTCP_scrollScreen (unsigned *x*, unsigned *y*, unsigned *times*, uint8_t *speed*)

Sets the scroll parameters and scrolls the screen.

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 GTCP_selectWindow (uint8_t *window*)

Selects the current window.

Parameters:

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

Returns:

none

void GTCP_separateScreens ()

Separates all of the screens into four separate screens.

Returns:

none

void GTCP_setAsciiVariant (uint8_t *code*)

Sets the desired ASCII variant to be used. Also called "International font select" in the GTCP 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 GTCP_setBackgroundColor (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Set the character background (highlight color) color.

- 0x00 = No brightness
- 0x7f = 50% brightness
- 0xff = Maximum brightness

Parameters:

red	Red brightness
green	Green brightness
blue	Blue brightness

Returns:

none

void GTCP_setCharacterColor (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Set the character color.

- 0x00 = No brightness
- 0x7f = 50% brightness
- 0xff = Maximum brightness

Parameters:

red	Red brightness
green	Green brightness
blue	Blue brightness

Returns:

none

void GTCP_setCharacterStyle (uint8_t *style*)

Set the character style.

- style = 0x00: Normal
- style = 0x01: Bold
- style = 0x02: Shadow
- style = 0x03: Bordering
- Default: style = 0x00

Parameters:

style	The desired character style.

Returns:

none

void GTCP_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 GTCP_setCompositionMode (uint8_t *mode*)

Sets the composition mode of the GTCP 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 GTCP_setCursor (unsigned *x*, unsigned *y*)

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

Parameters:

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

Returns:

none

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

Sets the font size for the GTCP series module.

Parameters:

x	X magnification factor
y	Y magnification factor

Returns:

none

void GTCP_setFontSize (uint8_t *size*)

Set the desired 1-byte character font size.

- size = 0x00: Outline font
- size = 0x01: 6x8 font
- size = 0x02: 8x16 font
- size = 0x03: 12x24 font
- size = 0x04: 16x32 font

Parameters:

size	The desired character size.

Returns:

none

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

Sets the font style for the GTCP series module.

Parameters:

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

Returns:

none

void GTCP_setHorizScrollSpeed (uint8_t *speed*)

Sets the horizontal scroll speed on the GTCP 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 GTCP_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 GTCP_setOutlineFontType (uint8_t *fontType*)

Set the outline font type. This needs to be set in order to use outline fonts.

- fontType = 0x00: Japanese
- fontType = 0x01: Korean
- fontType = 0x02: Simplified Chinese
- fontType = 0x03: Traditional Chinese
- fontType = 0x80: None (no outline font selected)
- fontType = 0xff: User-supplied font file
- Default: fontType = 0x80

Parameters:

fontType	The desired outline font type.

Returns:

none

void GTCP_setScreenBrightness (unsigned *level*)

Sets the screen brightness of the GTCP 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 GTCP_setScrollMode (uint8_t *mode*)

Sets the scroll mode on the GTCP module.

Scroll modes:

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

Parameters:

mode	The desired scroll mode.

Returns:

none

void GTCP_shortWait (uint8_t *time*)

Suspends the program and waits for a user defined amount of time. Wait time = t * IntTime

Parameters:

time	The amount of time to wait.

Returns:

none

void GTCP_springAction (uint8_t *direction*, uint8_t *speed*, unsigned *x*, unsigned *y*)

Start a curtain display action on-screen.

- direction = 0x00: To the right from the left edge.
- direction = 0x01: To the left from the right edge.
- direction = 0x02: To the left and right separately from the center.
- direction = 0x03: To the center from the left and right edge.

- Curtain action speed = 60 * speed * IntTime
- 0x00 <= speed <= 0xff

Parameters:

direction	Direction of curtain action.
speed	Spring action speed.
x	Display memory X position.
y	Display memory y position.

Returns:

none

void GTCP_storedBitImageDraw (uint8_t *memory*, unsigned *address*, unsigned *extension*, unsigned *xDefine*, unsigned *xSize*, unsigned *ySize*, uint8_t *format*)

Software datasheet funtion: Downloaded bit image display Display an image stored in system memory.

- memory = 0x00: RAM bit image
- memory = 0x01: FROM1 bit image
- memory = 0x02: Display memory
- memory = 0x10: FROM2 base address 0x0000.0000
- memory = 0x11: FROM2 base address 0x0100.0000

- memory = 0x12: FROM2 base address 0x0200.0000
- memory = 0x13: FROM2 base address 0x0300.0000
- memory = 0x14: FROM2 base address 0x0400.0000
- memory = 0x15: FROM2 base address 0x0500.0000
- memory = 0x16: FROM2 base address 0x0600.0000
- memory = 0x17: FROM2 base address 0x0700.0000
- memory = 0x18: FROM2 base address 0x0800.0000
- memory = 0x19: FROM2 base address 0x0900.0000
- memory = 0x1a: FROM2 base address 0x0a00.0000
- memory = 0x1b: FROM2 base address 0x0b00.0000
- memory = 0x1c: FROM2 base address 0x0c00.0000
- memory = 0x1d: FROM2 base address 0x0d00.0000
- memory = 0x1e: FROM2 base address 0x0e00.0000
- memory = 0x1f: FROM2 base address 0x0f00.0000

- format = 0x81: Monochrome (1-bit) format
- format = 0x86: Color 6-bit format
- format = 0x8c: Color 12-bit format
- format = 0x90: Color 16-bit format
- format = 0x91: Color 16-bit high-speed format
- format = 0x98: Color 24-bit format
- format = 0xf0: BMP file format

Parameters:

memory	The type of memory where the image is stored.
address	The memory address of the image.
extension	The address extension.
xDefine	The defined width to be displayed of the image.
xSize	The width of the image.
ySize	The height of the image.
format	The desired image format.

Returns:

none

void GTCP_switchMatrixMode (uint8_t *channel*, uint8_t *switchesX*, uint8_t *switchesY*, uint8_t *clearanceX*, uint8_t *clearanceY*)

Set the switch matrix mode.

- 0x00 <= channel <= 0x03
- 0x01 <= switchesX <= 0x10
- 0x01 <= switchesY <= 0x10
- 0x01 <= clearanceX <= 0x10
- 0x01 <= clearanceY <= 0x10

Parameters:

channel	The desired touch data channel.
switchesX	The number of switches on the x-axis.
switchesY	The number of switches on the y-axis.
clearanceX	The size of the non-responsive area in-between switches on the x-axis.
clearanceY	The size of the non-responsive area in-between switches on the y-axis.

Returns:

none

void GTCP_touchChannel (uint8_t *channel*)

Selects the currently-active touch panel control channel.

- 0x00 <= channel <= 0x03

Parameters:

channel	The desired touch control channel.

Returns:

none

void GTCP_touchDataEnable (uint8_t *enable*)

Enable or disable touch data transmission.

- enable = 0x00: Transmit OFF
- enable = 0x01: Transmit ON
- Default: enable = 0x00

Parameters:

enable	Turn touch data ON or OFF

Returns:

none

void GTCP_touchModeSelect (uint8_t *mode*)

Set the desired touch mode.

- mode = 0x00: Single touch mode
- 0x01 <= mode <= 0x0a: Multi-touch mode (up to 10 touches)
- Default mode: 0x00

Parameters:

mode	The desired touch mode.

Returns:

none

void GTCP_touchSettingPackageDataStore (uint8_t *packageNum,* uint8_t **data*)

Store touch setting package data.

This function is not intended for common use.

Please contact a Noritake sales consultant for further information.

- packageNum: 0x01 <= packageNum <= 0x04

Parameters:

packageNum	The desired package number.
Data	Touch setting package data.

Returns:

none

void GTCP_touchSettingPackageSelection (uint8_t *packageNum*)

Select the desired touch setting data package to use.

After this command is executed, touch control will start with the selected touch setting package number.

- packageNum: 0x00 <= packageNum <= 0x04
- Default: packageNum = 0x00

Parameters:

packageNum	The desired package number.

Returns:

none

void GTCP_usCommand ()

Returns:

none

void GTCP_useCustomChars (uint8_t *enable*)

Enables or disables the use of custom characters.

Parameters:

enable	Enable of disable custom characters.

Returns:

none

void GTCP_useMultiByteChars (uint8_t *enable*)

Enables the use of multiple-byte characters.

Parameters:

enable	Enable or disable multiple byte characters.

Returns:

none

void GTCP_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 GTCP_writeMixtureDisplayMode (uint8_t *mode*)

Select how background pixels are written to the display memory.

- mode = 0x10: Normal display write (Background pixels are written to display memory)
- mode = 0x11: Thru write (Background pixels are not written to display memory)
- Default: mode = 0x10

Parameters:

mode	The desired mixture display mode

Returns:

none

void GTCP_writeScreenMode (uint8_t *mode*)

Select how display actions will affect the module.

Display screen mode: Display action is valid within the Display or hidden area.

All screen mode: Display action is valid over the entire display memory.

- mode = 0x00: Display screen mode

- mode = 0x01: All screen mode
- Default: mode = 0x00 or memory SW setting

Parameters:

mode	The desired write screen mode.

Returns:

none

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

Sends an x and y coordinate to the GTCP display.

Parameters:

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

Returns:

none

void GTCP_XY1 (unsigned *x*, unsigned *y*)

Sends an x and y coordinate to the GTCP display.

Parameters:

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

Returns:

none

Variable Documentation

unsigned HEIGHT

int interface

unsigned LINES

unsigned WIDTH

GTCP_I2C.c File Reference

```
#include "GTCP_I2C.h"
#include <stddef.h>
```

Functions

- void **GTCP_I2C_Init** (uint8_t GTCPaddress)
- void **GTCP_I2C_Start** (uint8_t direction)
- uint8_t **GTCP_I2C_SingleReadAck** ()
- uint8_t **GTCP_I2C_ReadAck** ()
- uint8_t **GTCP_I2C_SingleReadNack** ()
- uint8_t **GTCP_I2C_ReadNack** ()
- void **GTCP_I2C_MultiRead** (uint8_t *data, int count)
- char * **GTCP_I2C_MultiReadTRDY** ()
- void **GTCP_I2C_Write** (uint8_t data)
- void **GTCP_I2C_MultiWrite** (uint8_t *data)
- void **GTCP_I2C_Stop** ()

Variables

- uint8_t **address**

Function Documentation

void GTCP_I2C_Init (uint8_t *GTCPaddress*)

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

Parameters:

GTCPaddress	The I2C address for the GTCP module.

Returns:

void

void GTCP_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

char* GTCP_I2C_MultiReadTRDY ()

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 GTCP_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 GTCP_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 GTCP_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 GTCP_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 GTCP_I2C_SingleReadNack ()

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

Returns:

read data

void GTCP_I2C_Start (uint8_t *direction*)

Starts the I2C communication on STM32F4xx.

Parameters:

direction	The master's communication direction.

Returns:

void

void GTCP_I2C_Stop ()

Stops the I2C communication.

Returns:

none

void GTCP_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

GTCP_SPI.c File Reference

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

Functions

- void **Delay** (__IO uint32_t time)
- void **GTCP_SPI_Init** ()
- void **startSPI** ()
- void **endSPI** ()
- void **initGTCP** ()
- 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 GTCP_SPI_Init ()

Initialize SPI communication.

Returns:

none

void initGTCP ()

Initialize the GTCP 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 chech 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

GTCP_USART.c File Reference

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

Functions

- void **GTCP_UART_Init** (uint32_t baudrate)
- 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 GTCP_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