# MSP430 GT-CP Code Library

# Reference Manual

**Noritake Co., Inc.**
**Version 2.3**
**Date of Issue: 8/15/2016 (V1.0)**
**Revision: 8/7/2018 (V1.1)**
**1/18/2019 (V1.2)**
**4/23/2021 (V2.0)**
**7/21/2021 (V2.2)**
**10/8/2021 (V2.3)**

# Table of Contents

# Data Structure Documentation

## TFT Struct Reference

`#include <GTCP.h>`

### Data Fields

- unsigned int **WIDTH**
- unsigned int **HEIGHT**
- unsigned int **LINES**
- unsigned int **tspID**
- uint8_t **gain**
- uint8_t **threshold**
- float **bootVer**
- float **firmwareVer**
- char **productType** [16]
- char **tspName** [16]

---

### Field Documentation

**float bootVer**

**float firmwareVer**

**uint8_t gain**

**unsigned int HEIGHT**

**unsigned int LINES**

**char productType[16]**

**uint8_t threshold**

**unsigned int tspID**

**char tspName[16]**

**unsigned int WIDTH**

---

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

- GTCP.h

# File Documentation

## GTCP.c File Reference

```
#include "GTCP.h"
```

### Macros

- #define **ARRAY_SIZE**(a)  (sizeof(a) / sizeof((a)[0]))

### Functions

- void **GTCP_print** (char data)
- uint8_t **GTCP_read** ()
- void **GTCP_readMulti** (uint8_t length)
- void **GTCP_start** (enum **interfaceType selectedInterface**)
- void **GTCP_readDisplayInfo** ()
- void **GTCP_printString** (unsigned char *string)
- void **GTCP_sendCommand** (uint8_t *data, int size)
- void **GTCP_initialize** ()
- void **GTCP_back** ()
- void **GTCP_forward** ()
- void **GTCP_lineFeed** ()
- void **GTCP_home** ()
- void **GTCP_carriageReturn** ()
- void **GTCP_crlf** ()
- void **GTCP_setCursor** (uint16_t x, uint16_t y)
- void **GTCP_clearScreen** ()
- void **GTCP_lineClear** ()
- void **GTCP_lineEndClear** ()
- void **GTCP_cursorOn** ()
- void **GTCP_cursorOff** ()
- void **GTCP_reset** ()
- void **GTCP_setDisplayScreenMode** ()
- void **GTCP_setAllScreenMode** ()
- void **GTCP_setNormalWriteDisplayMode** ()
- void **GTCP_setThruWriteDisplayMode** ()
- void **GTCP_enableMultiByteChars** ()
- void **GTCP_disableMultiByteChars** ()
- void **GTCP_setMultiByteCharSet** (uint8_t code)
- void **GTCP_enableCustomChars** ()
- void **GTCP_disableCustomChars** ()
- void **GTCP_defineCustomChar** (uint8_t codeStart, uint8_t codeEnd, uint8_t format, uint8_t pixelNumX, 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_setOverwriteMode** ()
- void **GTCP_setVerticalScrollMode** ()
- void **GTCP_setHorizontalScrollMode** ()
- void **GTCP_setHorizontalScrollONMode** ()
- void **GTCP_setHorizScrollSpeed** (uint8_t speed)
- void **GTCP_invertOff** ()
- void **GTCP_invertOn** ()
- void **GTCP_wait** (uint8_t wait)
- void **GTCP_shortWait** (uint8_t time)

- void **GTCP_blinkScreenOff** ()
- void **GTCP_blinkScreenOn** (uint8_t pattern, uint8_t onTime, uint8_t offTime, uint8_t cycles)
- void **GTCP_scrollScreen** (uint16_t x, uint16_t y, uint16_t 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, uint16_t x, uint16_t y)
- void **GTCP_randomAction** (uint8_t actionType, uint8_t speed, uint16_t x, uint16_t y)
- void **GTCP_fadeInAction** (uint8_t speed, uint16_t x, uint16_t y)
- void **GTCP_fadeOutAction** (uint8_t speed)
- void **GTCP_displayOff** ()
- void **GTCP_displayOn** ()
- void **GTCP_displayOffWaitTouch** ()
- void **GTCP_setScreenBrightness** (uint8_t level)
- void **GTCP_setFontWidth** (uint8_t widthSetting)
- void **GTCP_setCharacterType** (uint8_t type)
- void **GTCP_setFontSize** (uint8_t size)
- void **GTCP_setOutlineFontSize** (uint16_t charLineHeight, uint16_t nominalCharY, uint16_t nominalCharX, int16_t baseOffsetY)
- void **GTCP_setUserFontAddressSize**(uint32_t address, uint32_t size)
- void **GTCP_setOutlineFontType** (uint8_t fontType)
- void **GTCP_setFontMagnification** (uint8_t x, uint8_t y)
- 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_setShadowBorderingColor** (uint8_t red, uint8_t green, uint8_t blue)
- void **GTCP_turnBackgroundOff** ()
- void **GTCP_turnBackgroundOn** ()
- void **GTCP_selectWindow** (uint8_t window)
- void **GTCP_defineWindow** (uint8_t window, uint16_t x, uint16_t y, uint16_t width, uint16_t height)
- void **GTCP_deleteWindow** (uint8_t window)
- void **GTCP_drawImage** (uint16_t width, uint16_t height, uint8_t format, unsigned char *data)
- void **GTCP_storedBitImageDraw** (uint8_t memory, uint32_t address, uint16_t xDefine, uint16_t xSize, uint16_t ySize, uint8_t format)
- void **GTCP_pixelDrawing** (uint8_t pen, uint16_t x, uint16_t y)
- void **GTCP_drawLineBox** (uint8_t mode, uint8_t pen, uint16_t xStart, uint16_t yStart, uint16_t xEnd, uint16_t yEnd)
- void **GTCP_enableTouchTransmission** ()
- void **GTCP_disableTouchTransmission** ()
- void **GTCP_setTouchChannel** (uint8_t channel)
- void **GTCP_setSingleTouchMode** ()
- void **GTCP_setMultiTouchMode** (uint8_t maxTouches)
- 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_customSwitchModeSingle** (uint8_t channel, uint16_t switchX, uint16_t switchY, uint16_t sizeX, uint16_t sizeY)
- void **GTCP_customSwitchModeMultiple** (uint8_t channel, uint8_t switchNum)
- void **GTCP_customSwitchParam** (uint16_t switchX, uint16_t switchY, uint16_t sizeX, uint16_t sizeY)
- void **GTCP_IOPortSetting** (uint8_t portNumber, uint8_t portSetting)
- void **GTCP_IOPortOutput** (uint8_t portNumber, uint8_t portValue)
- void **GTCP_IOPortInput** (uint8_t portNumber)
- void **GTCP_setSingleMemorySW** (uint8_t memorySW, uint8_t data)
- void **GTCP_setMultipleMemorySwitchHeader** (uint8_t numSettings)
- void **GTCP_setMemorySwitch** (uint8_t switchNumber, uint8_t data)
- void **GTCP_getSingleMemorySWData** (uint8_t switchNum)

- void **GTCP_getMultipleMemorySWData** (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_enterUserSetupMode** ()
- void **GTCP_endUserSetupMode** ()
- void **GTCP_FROMUserFontDefinition** (uint8_t table, uint8_t *data)
- void **GTCP_RAMMacroDefineDelete** (unsigned int length, uint8_t *data)
- void **GTCP_FROMMacroDefineDelete** (uint8_t registrationNum, unsigned int length, 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** (uint32_t size, uint8_t memorySelect, uint32_t address, uint8_t *data)
- void **GTCP_memoryTransfer** (uint32_t size, uint8_t destMemory, uint32_t destAddress, uint8_t srcMemory, uint32_t srcAddress)
- void **GTCP_readMemory** (uint8_t size, uint8_t memorySelect, uint32_t address)
- float **GTCP_getBootVersion** ()
- float **GTCP_getFirmwareVersion** ()
- void **GTCP_getCharacterCodeInfo** ()
- void **GTCP_getLanguageType** ()
- void **GTCP_getMemoryChecksum** (uint8_t address, uint8_t length)
- uint32_t **GTCP_getFROM2MemoryChecksum** (uint32_t address, uint32_t size)
- void **GTCP_getProductType** ()
- void **GTCP_getHorizontalResolution** ()
- void **GTCP_getVerticalResolution** ()
- void **GTCP_getTSPName** ()
- void **GTCP_getTSPID** ()
- void **GTCP_getTouchGain** ()
- void **GTCP_getTouchThreshold** ()
- void **GTCP_touchSettingPackageDataStore** (uint8_t packageNum, uint8_t *data)
- void **GTCP_touchSettingPackageSelection** (uint8_t packageNum)
- void **setPowerSavingMode** (uint8_t wakeup)
- void **setTouchScanPeriod** (uint8_t period)
- void **GTCP_setDisplayOrientation** (int degrees)

---

## Macro Definition Documentation

### #define ARRAY_SIZE( a)  (sizeof(a) / sizeof((a)[0]))

---

## 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_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 *pattern*, uint8_t *onTime*, uint8_t *offTime*, uint8_t *cycles*)

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

- 0x01 <= onTime <= 0xff
- 0x01 <= offTime <= 0xff
- 0x00 <= cycles <= 0xff

**Parameters**

| | |
|---|---|
| *pattern* | Enables or disables screen blinking.<br>• 0x00 - Normal display<br>• 0x01 - Blink display (alternate between normal and blank display)<br>• 0x02 - Reserved. |
| *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_customSwitchModeMultiple (uint8_t *channel*, uint8_t *switchNum*)**

Starts the creation of multiple custom switches. This should be followed by loop of customSwitchParam() that runs **switchNum** times.

**Parameters**

| | |
|---|---|
| *channel* | The desired touch data channel. |
| *switchNum* | Number of switches to define. |

**Returns**

none

**void GTCP_customSwitchModeSingle (uint8_t *channel*, uint16_t *switchX*, uint16_t *switchY*, uint16_t *sizeX*, uint16_t *sizeY*)**

Creates 1 custom switch.

**Parameters**

| | |
|---|---|
| *channel* | The desired touch data channel. |
| *switchX* | The switch's X coordinate. |
| *switchY* | The switch's Y coordinate. |

| | |
|---|---|
| *sizeX* | Switch width. |
| *sizeY* | Switch height. |

**Returns**

none

**void GTCP_customSwitchParam (uint16_t *switchX*, uint16_t *switchY*, uint16_t *sizeX*, uint16_t *sizeY*)**

This defines a custom switch. This should be used after customSwitchModeMultiple()

**Parameters**

| | |
|---|---|
| *switchX* | The switch's X coordinate. |
| *switchY* | The switch's Y coordinate. |
| *sizeX* | Switch width. |
| *sizeY* | Switch height. |

**Returns**

none

**void GTCP_defineCustomChar (uint8_t *codeStart*, uint8_t *codeEnd*, uint8_t *format*, uint8_t *pixelNumX*, 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

- 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**

| | |
|---|---|
| *codeStart* | The code for the character being defined. |
| *codeEnd* | The ending character code value. |
| *format* | The format of the character being defined. |
| **data* | The data for the character being defined. |

**Returns**

none

### void GTCP_defineWindow (uint8_t *window*, uint16_t *x*, uint16_t *y*, uint16_t *width*, uint16_t *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.

- type = 0x01: Select 6x8 pixel character.
- type = 0x02: Select 8x16 pixel character.
- type = 0x03: Select 12x24 pixel character.
- type = 0x04: Select 16x32 pixel character.
- 0x20 <= code <= 0xff

**Parameters**

| | |
|---|---|
| *type* | The custom character type to be deleted. |
| *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_disableCustomChars ()**

Disables the use of custom characters.

**Returns**

none

**void GTCP_disableMultiByteChars ()**

Disables the use of multiple-byte characters.

**Returns**

none

**void GTCP_disableTouchTransmission ()**

Disable touch data transmission.

**Returns**

none

**void GTCP_displayOff ()**

Turns the display OFF.

**Returns**

none

**void GTCP_displayOffWaitTouch ()**

Turn the display OFF and wait for touch input to turn the display back ON.

**Returns**

none

**void GTCP_displayOn ()**

Turns the display ON.

**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 (uint16_t *width*, uint16_t *height*, uint8_t *format*, unsigned char * *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*, uint16_t *xStart*, uint16_t *yStart*, uint16_t *xEnd*, uint16_t *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 start position. |
| *yStart* | Line/Box drawing start position. |
| *xEnd* | Line/Box drawing end position. |
| *yEnd* | Line/Box drawing end position. |

**Returns**

none

### void GTCP_enableCustomChars ()

Enables the use of custom characters.

**Returns**

none

### void GTCP_enableMultiByteChars ()

Enables the use of multiple-byte characters.

**Returns**

none

### void GTCP_enableTouchTransmission ()

Enable touch data transmission.

**Returns**

none

### void GTCP_endUserSetupMode ()

End (exit) user setup mode and initiate a software reset.

**Returns**

none

### void GTCP_enterUserSetupMode ()

Start user setup mode.

The following data is transmitted when entering user setup mode:

1. Header | 0x28 | 1 byte
2. Identifier 1 | 0x65 | 1 byte
3. Identifier 2 | 0x01 | 1 byte
4. NUL | 0x00 | 1 byte

**Returns**

none

### void GTCP_fadeInAction (uint8_t *speed*, uint16_t *x*, uint16_t *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*, unsigned int *length*, 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.

THIS COMMAND IS ONLY VALID IN USER SETUP MODE.

**Parameters**

| | |
|---|---|
| *table* | The desired user table font size. |
| *data* | The user table data. |

**Returns**

none

### float GTCP_getBootVersion ()

Request the connected GT-CP module's boot version. After this command is sent, the module will send 7 response bytes. Example response: "0.00"

**Returns**

none

### void GTCP_getCharacterCodeInfo ()

Request the connected GT-CP module's character code information. After this command is sent, the module will send 18 response bytes. Example response: "JIS X208 L 100"

**Returns**

none

### float GTCP_getFirmwareVersion ()

Request the connected GT-CP module's firmware version. After this command is sent, the module will send 7 response bytes. Example response: "1.62"

**Returns**

none

### uint32_t GTCP_getFROM2MemoryChecksum (uint32_t *address*, uint32_t *size*)

Request the connected GT-CP module's FROM2 memory checksum. After this command is sent, the module will send 7 response bytes.

**Parameters**

| | |
|---|---|
| *address* | FROM2 address for checksum. |
| *size* | Size of checksum calculation. |

**Returns**

none

### void GTCP_getHorizontalResolution ()

Request the connected GT-CP module's horizontal resolution. After this command is sent, the module will send 6 response bytes. Example response: "800"

**Returns**

none

### void GTCP_getLanguageType ()

Request the connected GT-CP module's language type. After this command is sent, the module will send 18 response bytes. Example response: "MULTI LANGUAGE"

**Returns**

none

**void GTCP_getMemoryChecksum (uint8_t *address*, uint8_t *length*)**

Request the connected GT-CP module's memory checksum. After this command is sent, the module will send 7 response bytes.

**Returns**

none

**void GTCP_getMultipleMemorySWData (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_getProductType ()**

Request the connected GT-CP module's product type. After this command is sent, the module will send 18 response bytes. Example response: "GT-C900PA"

**Returns**

none

**void GTCP_getSingleMemorySWData (uint8_t *switchNum*)**

Modify a single memory switch.

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_getTouchGain ()**

Request the connected GT-CP module's current touch gain value. After this command is sent, the module will send 4 response bytes.

**Returns**

none

**void GTCP_getTouchThreshold ()**

Request the connected GT-CP module's current touch threshold value. After this command is sent, the module will send 4 response bytes.

**Returns**

none

**void GTCP_getTSPID ()**

Request the connected GT-CP module's active touch setting package ID. After this command is sent, the module will send 7 response bytes. Example response: "0003"

**Returns**

none

**void GTCP_getTSPName ()**

Request the connected GT-CP module's active touch setting package name. After this command is sent, the module will send 18 response bytes. Example response: "Noritake"

**Returns**

none

**void GTCP_getVerticalResolution ()**

Request the connected GT-CP module's vertical resolution. After this command is sent, the module will send 6 response bytes. Example response: "480"

**Returns**

none

**void GTCP_home ()**

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

**Returns**

none

**void GTCP_initialize ()**

Send the initialize command to the GTCP display.

**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 (uint8_t *portNumber*)**

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

- portNumber = 0x00: Port 0 (GPIO 0-7)
- portNumber = 0x01: Port 1 (GPIO 8-15)
- portNumber = 0x02: Port 2 (GPIO 16-32)
- portNumber = 0x03: Port 3 (GPIO 24, 25)

**Parameters**

| | |
|---|---|
| *portNumber* | The port number to read. |

**Returns**

none

### void GTCP_IOPortOutput (uint8_t *portNumber*, uint8_t *portValue*)

Function to set the output value on the GPIO ports. Settings:

- portNumber = 0x00: Port 0 (GPIO 0-7)
- portNumber = 0x01: Port 1 (GPIO 8-15)
- portNumber = 0x02: Port 2 (GPIO 16-32)
- portNumber = 0x03: Port 3 (GPIO 24, 25)
- 0x00 <= portValue <= 0xff

**Parameters**

| | |
|---|---|
| *portNumber* | The desired port number to set. |
| *portValue* | The value to put on the port. |

**Returns**

none

### void GTCP_IOPortSetting (uint8_t *portNumber*, uint8_t *portSetting*)

Function to set the GPIO ports on the GTCP unit to be inputs or outputs. Settings:

- portNumber = 0x00: Port 0 (GPIO 0-7)
- portNumber = 0x01: Port 1 (GPIO 8-15)
- portNumber = 0x02: Port 2 (GPIO 16-32)
- portNumber = 0x03: Port 3 (GPIO 24, 25)
- portSetting = 0x00 (All input) to 0x0f (all output)
- 0x00 <= portSetting <= 0xff
- The portSetting parameter is bit-wise to the ports present on the module.

**Parameters**

| | |
|---|---|
| *portNumber* | The desired port number to set. |
| *portSetting* | The passed in port setting. |

**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_memoryStore (uint32_t *size*, uint8_t *memorySelect*, uint32_t *address*, 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 * 0x00010000) <= 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):
- 0x000001 <= (sizeLower + sizeUpper * 0x0100 + sizeExtension * 0x00010000) <= 0x00001000
- 0x000000 <= (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_memoryTransfer (uint32_t *size*, uint8_t *destMemory*, uint32_t *destAddress*, uint8_t *srcMemory*, uint32_t *srcAddress*)

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 * 0x100 + destAddressExtension * 0x10000) <= 0x00FFFF
- 0x000000 <= (srcAddressLower + srcAddressUpper * 0x100 + 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*, uint16_t *x*, uint16_t *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*)

Helper function to reduce code and use the correct interface. This function will send the passed-in data through the selected interface.

**Parameters**

| | |
|---|---|
| *data* | The data byte to be sent. |

**Returns**

none

### void GTCP_printString (unsigned char * *string*)

General print string function for all interfaces.

**Parameters**

| | |
|---|---|
| *string* | The string to print to the display. |

**Returns**

none

### void GTCP_RAMMacroDefineDelete (unsigned int *length*, uint8_t * *data*)

Define or delete RAM Macro or RAM Program Macro.

- 0x0000 <= length <= 0x0400
- length > 0x0000: Supplied data is stored as Macro
- length = 0x0000: Macro is deleted

**Parameters**

| | |
|---|---|
| *length* | RAM Macro data length |

| | |
|---|---|
| *data* | RAM Macro data |

**Returns**

none

### void GTCP_randomAction (uint8_t *actionType*, uint8_t *speed*, uint16_t *x*, uint16_t *y*)

Start a checkerboard display action. Command name on software datasheet is "Random Display Action XY".

- 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

### uint8_t GTCP_read ()

Read data from the selected interface.

**Returns**

none

### void GTCP_readDisplayInfo ()

Reads all applicable display information and stores info in the global **TFT** struct "display".

**Returns**

none

### void GTCP_readMemory (uint8_t *size*, uint8_t *memorySelect*, uint32_t *address*)

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 <= address <= 0xFFFFFF
- 0x000000 <= size <= 0xFF
- memorySelect = 0x30 (General-purpose RAM):
- 0x000000 <= address <= 0xFF
- 0x000001 <= size <= 0x000400
- memorySelect = 0x31 (General-purpose FROM):
- 0x000000 <= address <= 0x00FFFF
- 0x000001 <= size <= 0xFF

The following data is transmitted:

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

**Parameters**

| | |
|---|---|
| *size* | Data size, restricted to 255 bytes |
| *memorySelect* | Desired memory select |
| *address* | Memory address, upper byte |

**Returns**

none

### void GTCP_readMulti (uint8_t *length*)

Read multiple bytes of data from the display module.

**Parameters**

| | |
|---|---|
| *length* | Number of bytes to read. |

**Returns**

none

### void GTCP_reset ()

Perform a hardware reset

**Returns**

none

### void GTCP_scrollScreen (uint16_t *x*, uint16_t *y*, uint16_t *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_sendCommand (uint8_t * *data*, int *size*)**

Send a packet of data to the connected display.

**Parameters**

| | |
|---|---|
| *data* | Data to send. |
| *size* | Size of data. |

**Returns**

none

**void GTCP_setAllScreenMode ()**

Set all screen mode. During all screen mode, display actions are valid over the entire display memory.

**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.

- 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_setCharacterType (uint8_t *type*)

Sets the 2-byte character type for the GTCP series module.

- type = 0x00 | Japanese (JIS X0208 (Shift-JIS))
- type = 0x01 | Korean (KSC5601-87)
- type = 0x02 | Simplified Chinese (GB2312-80)
- type = 0x03 | Traditional Chinese (Big-5)

**Parameters**

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

**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)
- 0xFE - UTF-8 input
- 0xFF - User table

**Parameters**

| | |
|---|---|
| *code* | The code for the desired character set. |

**Returns**

none

### void GTCP_setCursor (uint16_t *x*, uint16_t *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_setDisplayOrientation (int *degrees*)

Set the module's display orientation. This function will change the module's memory switch orientation value and then reset the module to use the new orientation. User setup mode is used while changing memory switch.

**Parameters**

| | |
|---|---|
| *degrees* | The desired display orientation. |

**Returns**

none

### void GTCP_setDisplayScreenMode ()

Set display screen mode. During display screen mode, display actions are valid within the display or hidden area.

**Returns**

none

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

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_setFontWidth (uint8_t *widthSetting*)**

Set the desired character width.

- widthSetting = 0x00 | Fixed-width
- widthSetting = 0x02 | Proportional 1
- widthSetting = 0x03 | Proportional 2
- widthSetting = 0x04 | Proportional 3

**Parameters**

| | |
|---|---|
| *widthSetting* | Desired character width setting. |

**Returns**

none

**void GTCP_setHorizontalScrollMode ()**

Set the display mode to horizontal scroll mode.

**Returns**

none

**void GTCP_setHorizontalScrollONMode ()**

Set the display mode to horizontal scroll mode with the scroll ON state.

**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 pixels
- speed = 0x02 - 0x1f : (n-1) * T ms / pixel

**Parameters**

| | |
|---|---|
| *speed* | The desired horizontal scroll speed. |

**Returns**

none

**void GTCP_setMemorySwitch (uint8_t *switchNumber*, uint8_t *data*)**

Modify a memory switch. This function is to be used **numSettings** times after **GTCP_setMultipleMemorySwitchHeader()**.

**Parameters**

| | |
|---|---|
| *switchNumber* | The memory switch to modify. |
| *Data* | The data to write to memory switch. |

**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_setMultipleMemorySwitchHeader (uint8_t *numSettings*)

Function to start modifying multiple memory switch values in a row. This should be followed by **GTCP_setMemorySwitch()** used **numSettings** times.

This command can only be executed while in "user setup mode". So, use this function after **GTCP_enterUserSetupMode()**.

**Parameters**

| | |
|---|---|
| *numSettings* | Number of memory switch settings to write. |

**Returns**

none

### void GTCP_setMultiTouchMode (uint8_t *maxTouches*)

Set multi-touch mode and configure maximum number of simultaneous touches.

**Parameters**

| | |
|---|---|
| *maxTouches* | Maximum number of simultaneous touches. |

**Returns**

none

### void GTCP_setNormalWriteDisplayMode ()

Set display to normal write mode. Background pixels are written to display memory.

This is the default write mode.

**Returns**

none

### void GTCP_setOutlineFontSize (uint16_t *charLineHeight*, uint16_t *nominalCharY*, uint16_t *nominalCharX*, int16_t *baseOffsetY*)

Set the desired outline font size.

- 0x0001 <= charLineHeight <= Ydots
- 0x0001 <= nominalCharY <= 0x03ff
- 0x0001 <= nominalCharX <= 0x03ff
- 0x0001 <= baseOffsetY <= 0x03ff | if 0x8000: zero offset

**Parameters**

| | |
|---|---|
| *charLineHeight* | Y-size (height) of character line (line spacing), in pixels. |
| *nominalCharY* | Nominal character Y-size (height), in pixels. |
| *nominalCharX* | Nominal character X-size (width), in pixels. |
| *baseOffsetY* | Baseline Y-offset from cursor position, in pixels. |

**Returns**

none

### void GTCP_setOutlineFontSize (uint32_t address, uint32_t size)

Set the address and size for a user-supplied font file in FROM2.

- 0x0000 <= address <= 0x0FAFFFFF
- 0x0001 <= size <= 0x0FB00000

**Parameters**

| | |
|---|---|
| *address* | Font file start address in general-purpose FROM2. |
| *size* | Font file size in general-purpose FROM2. |

**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_setOverwriteMode ()

Set the display mode to overwrite mode.

**Returns**

none

### void GTCP_setScreenBrightness (uint8_t *level*)

Sets the screen brightness of the GTCP module.

Brightness level = (level / 255) * 100%

**Parameters**

| | |
|---|---|
| *level* | The desired brightness level. |

**Returns**

none

### void GTCP_setShadowBorderingColor (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Set the character shadow and bordering color.

- 0x00 = No brightness
- 0x7F = 50% brightness
- 0xFF = Maximum brightness

**Parameters**

| | |
|---|---|
| *red* | Red brightness |
| *green* | Green brightness |
| *blue* | Blue brightness |

**Returns**

none

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

Sets a memory switch to the passed in value.

This command can only be executed while in "user setup mode". So, use this function after **GTCP_enterUserSetupMode()**.

**Parameters**

| | |
|---|---|
| *memorySW* | The memory SW to be changed. |
| *data* | The data to be written to the memory SW. |

**Returns**

none

**void GTCP_setSingleTouchMode ()**

Set single touch mode.

**Returns**

none

**void GTCP_setThruWriteDisplayMode ()**

Set display to thru write mode. Background pixels are not written to display memory.

**Returns**

none

**void GTCP_setTouchChannel (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_setVerticalScrollMode ()**

Set the display mode to vertical 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*, uint16_t *x*, uint16_t *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_start (enum interfaceType *selectedInterface*)**

New GTCP Initialize function

- Establish interface
- Initialize module
- Collect module information
- Module information is stored in display struct.

**Parameters**

| | |
|---|---|
| *interface* | Selected serial interface. (UART, I2C, or SPI) |

**Returns**

none

**void GTCP_storedBitImageDraw (uint8_t *memory*, uint32_t *address*, uint16_t *xDefine*, uint16_t *xSize*, uint16_t *ySize*, uint8_t *format*)**

Software datasheet function: 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. |
| *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_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_turnBackgroundOff ()

Turns the background OFF. When the background is OFF, its color is always black.

#### Returns

none

### void GTCP_turnBackgroundOn ()

Turns the background ON. When the background is ON, its color is determined by GTCP_setBackgroundColor.

#### Parameters

| | |
|---|---|
| *none* | |

### void GTCP_wait (uint8_t *wait*)

Suspends the program and waits for a user defined amount of time. Wait time = t * approximately 0.47s

#### Parameters

| | |
|---|---|
| *wait* | The amount of time for the MCU to wait. |

#### Returns

none

### void setPowerSavingMode (uint8_t *wakeup*)

Set the desired power saving mode.

The wakeup parameter determines many sleep mode options related to different bits.

**Definable area:**

m = 01h: Sleep mode

w = b7(upper bit), b6, b5, b4, b3, b2, b1, b0(lower bit)

| bit | 0 | 1 |
|---|---|---|
| b7 | Reserved | |
| b6 | Refer to the following list | |
| b5 | | |
| b4 | | |
| b3 | Reserved | |
| b2 | Reserved | |
| b1 | Reserved | |
| b0 | No wakeup on touch | Wakeup on touch |

* If w = 00h (no wakeup methods are set), return to normal operation is only possible by external reset or VCC power cycling.

| b4 | b5 | b6 | GPIO15 wakeup condition |
|---|---|---|---|
| 0 | - | - | No wakeup on GPIO15 input |
| 1 | 0 | 0 | Wakeup on GPIO15 LOW level |
| 1 | 0 | 1 | Wakeup on GPIO15 falling edge |
| 1 | 1 | 0 | Wakeup on GPIO15 HIGH level |
| 1 | 1 | 1 | Wakeup on GPIO15 rising edge |

#### Parameters

| | |
|---|---|
| *mode* | Desired power saving mode. |
| *wakeup* | Desired wake up method. |

#### Returns

none

### void setTouchScanPeriod (uint8_t *period*)

Set the amount of time in-between each touch scan. This parameter is used if the "wakeup on touch" power saving mode is selected.

0x05 (5ms) <= period <= 0xFE (254ms)

Default: memory switch 61

### Parameters

| | |
|---|---|
| *period* | The amount of time between each touch scan. |

### Returns

none

# GTCP.h File Reference

```
#include <msp430f5529.h>
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "GTCP_SPI.h"
#include "GTCP_I2C.h"
#include "GTCP_UART.h"
```

## Data Structures

- struct **TFT**

## Macros

- #define **MAX_MATRIX_SWITCHES_X** 0x10
- #define **MAX_MATRIX_SWITCHES_Y** 0x10
- #define **TOUCH_HEADER** 0x10
- #define **TOUCH_COORDINATE_RELEASED** 0x10
- #define **TOUCH_COORDINATE_TOUCHED** 0x11
- #define **TOUCH_MATRIXSWITCH_RELEASED** 0x20
- #define **TOUCH_MATRIXSWITCH_TOUCHED** 0x21
- #define **TOUCH_CUSTOMSWITCH_RELEASED** 0x30
- #define **TOUCH_CUSTOMSWITCH_TOUCHED** 0x31
- #define **LINE_DRAW** 0
- #define **BOX_DRAW** 1
- #define **BOXFILL_DRAW** 2
- #define **PEN_BACKGROUND_COLOR** 0
- #define **PEN_CHARACTER_COLOR** 1
- #define **IMAGE_FORMAT_MONOCHROME** 0x81
- #define **IMAGE_FORMAT_6BIT** 0x86
- #define **IMAGE_FORMAT_12BIT** 0x8C
- #define **IMAGE_FORMAT_16BIT** 0x90
- #define **IMAGE_FORMAT_16BIT_HS** 0x91
- #define **IMAGE_FORMAT_24BIT** 0x98
- #define **IMAGE_FORMAT_BMP** 0xF0
- #define **GTCP_PORT0** 0
- #define **GTCP_PORT1** 1
- #define **GTCP_PORT2** 2
- #define **GTCP_PORT3** 3
- #define **GPIO0** 0b00000001
- #define **GPIO1** 0b00000010
- #define **GPIO2** 0b00000100
- #define **GPIO3** 0b00001000
- #define **GPIO4** 0b00010000
- #define **GPIO5** 0b00100000
- #define **GPIO6** 0b01000000
- #define **GPIO7** 0b10000000
- #define **GPIO8** 0b00000001
- #define **GPIO9** 0b00000010
- #define **GPIO10** 0b00000100
- #define **GPIO11** 0b00001000
- #define **GPIO12** 0b00010000
- #define **GPIO13** 0b00100000
- #define **GPIO14** 0b01000000
- #define **GPIO15** 0b10000000

- #define **GPIO16** 0b00000001
- #define **GPIO17** 0b00000010
- #define **GPIO18** 0b00000100
- #define **GPIO19** 0b00001000
- #define **GPIO20** 0b00010000
- #define **GPIO21** 0b00100000
- #define **GPIO22** 0b01000000
- #define **GPIO23** 0b10000000
- #define **GPIO24** 0b00000001
- #define **GPIO25** 0b00000010
- #define **INTERNATIONAL_FONT_SET_MSW** 0
- #define **CHARACTER_TABLE_TYPE_MSW** 1
- #define **HORIZONTAL_SCROLL_SPEED_MSW** 2
- #define **REVERSE_DISPLAY_MSW** 3
- #define **WRITE_MIXTURE_DISPLAY_MODE_MSW** 4
- #define **BRIGHTNESS_LEVEL_MSW** 5
- #define **DISPLAY_ORIENTATION_MSW** 6
- #define **WRITE_SCREEN_MODE_MSW** 7
- #define **FONT_SIZE_MSW** 8
- #define **TWO_BYTE_CHARACTER_MSW** 9
- #define **FONT_MAGNIFICATION_X_MSW** 10
- #define **FONT_MAGNIFICATION_Y_MSW** 11
- #define **CHARACTER_STYLE_MSW** 12
- #define **TWO_BYTE_CHARACTER_TYPE_MSW** 13
- #define **DOWNLOAD_6X8_CHARACTER_RESTORE_MSW** 16
- #define **DOWNLOAD_8X16_CHARACTER_RESTORE_MSW** 17
- #define **DOWNLOAD_16X16_CHARACTER_RESTORE_MSW** 18
- #define **FROM_MACRO_EXECUTE_AT_POWER_ON_MSW** 19
- #define **UART_BAUDRATE_MSW** 48
- #define **UART_PARITY_MSW** 49
- #define **MACRO_END_CODE_ENABLE_DISABLE_MSW** 52
- #define **MACRO_END_CODE_MSW** 53
- #define **MACRO_END_CLEAR_SCREEN_MSW** 54
- #define **TOUCH_GAIN_MSW** 58
- #define **TOUCH_THRESHOLD_MSW** 59
- #define **TOUCH_TRANSMIT_MODE_MSW** 60
- #define **TOUCH_SCAN_PERIOD_MSW** 61
- #define **TOUCH_SENSITIVITY_SELECTION_MSW** 62
- #define **TOUCH_PACKAGE_SELECTION_MSW** 63
- #define **CHARACTER_TABLE_6X8** 1
- #define **CHARACTER_TABLE_8X16** 2
- #define **CHARACTER_TABLE_12X24** 3
- #define **CHARACTER_TABLE_16X32** 4
- #define **CHARACTER_TABLE_6X8_SIZE** 768
- #define **CHARACTER_TABLE_8X16_SIZE** 2048
- #define **CHARACTER_TABLE_12X24_SIZE** 4608
- #define **CHARACTER_TABLE_16X32_SIZE** 8192

## Enumerations

- enum **interfaceType** { **UART**, **SPI**, **I2C** }

## Functions

- void **GTCP_print** (char data)
- uint8_t **GTCP_read** ()
- void **GTCP_readMulti** (uint8_t length)
- void **GTCP_start** (enum **interfaceType selectedInterface**)

- void **GTCP_readDisplayInfo** ()
- void **GTCP_printString** (unsigned char *string)
- void **GTCP_sendCommand** (uint8_t *data, int size)
- void **GTCP_initialize** ()
- void **GTCP_back** ()
- void **GTCP_forward** ()
- void **GTCP_lineFeed** ()
- void **GTCP_home** ()
- void **GTCP_carriageReturn** ()
- void **GTCP_crlf** ()
- void **GTCP_setCursor** (uint16_t x, uint16_t y)
- void **GTCP_clearScreen** ()
- void **GTCP_lineClear** ()
- void **GTCP_lineEndClear** ()
- void **GTCP_cursorOn** ()
- void **GTCP_cursorOff** ()
- void **GTCP_reset** ()
- void **GTCP_setDisplayScreenMode** ()
- void **GTCP_setAllScreenMode** ()
- void **GTCP_setNormalWriteDisplayMode** ()
- void **GTCP_setThruWriteDisplayMode** ()
- void **GTCP_enableMultiByteChars** ()
- void **GTCP_disableMultiByteChars** ()
- void **GTCP_setMultiByteCharSet** (uint8_t code)
- void **GTCP_enableCustomChars** ()
- void **GTCP_disableCustomChars** ()
- 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_setOverwriteMode** ()
- void **GTCP_setVerticalScrollMode** ()
- void **GTCP_setHorizontalScrollMode** ()
- void **GTCP_setHorizontalScrollONMode** ()
- void **GTCP_setHorizScrollSpeed** (uint8_t speed)
- void **GTCP_invertOff** ()
- void **GTCP_invertOn** ()
- void **GTCP_wait** (uint8_t wait)
- void **GTCP_shortWait** (uint8_t time)
- void **GTCP_blinkScreenOff** ()
- void **GTCP_blinkScreenOn** (uint8_t pattern, uint8_t onTime, uint8_t offTime, uint8_t cycles)
- void **GTCP_scrollScreen** (uint16_t x, uint16_t y, uint16_t 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, uint16_t x, uint16_t y)
- void **GTCP_randomAction** (uint8_t actionType, uint8_t speed, uint16_t x, uint16_t y)
- void **GTCP_fadeInAction** (uint8_t speed, uint16_t x, uint16_t y)
- void **GTCP_fadeOutAction** (uint8_t speed)
- void **GTCP_displayOff** ()
- void **GTCP_displayOn** ()
- void **GTCP_displayOffWaitTouch** ()
- void **GTCP_setScreenBrightness** (uint8_t level)
- void **GTCP_setFontWidth** (uint8_t widthSetting)
- void **GTCP_setFontSize** (uint8_t size)
- void **GTCP_setOutlineFontSize** (uint16_t charLineHeight, uint16_t nominalCharY, uint16_t nominalCharX, int16_t baseOffsetY)

- void **GTCP_setOutlineFontType** (uint8_t fontType)
- void **GTCP_setFontMagnification** (uint8_t x, uint8_t y)
- 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_setShadowBorderingColor** (uint8_t red, uint8_t green, uint8_t blue)
- void **GTCP_turnBackgroundOff** ()
- void **GTCP_turnBackgroundOn** ()
- void **GTCP_selectWindow** (uint8_t window)
- void **GTCP_defineWindow** (uint8_t window, uint16_t x, uint16_t y, uint16_t width, uint16_t height)
- void **GTCP_deleteWindow** (uint8_t window)
- void **GTCP_drawImage** (uint16_t width, uint16_t height, uint8_t format, unsigned char *data)
- void **GTCP_storedBitImageDraw** (uint8_t memory, uint32_t address, uint16_t xDefine, uint16_t xSize, uint16_t ySize, uint8_t format)
- void **GTCP_pixelDrawing** (uint8_t pen, uint16_t x, uint16_t y)
- void **GTCP_drawLineBox** (uint8_t mode, uint8_t pen, uint16_t xStart, uint16_t yStart, uint16_t xEnd, uint16_t yEnd)
- void **GTCP_enableTouchTransmission** ()
- void **GTCP_disableTouchTransmission** ()
- void **GTCP_setTouchChannel** (uint8_t channel)
- void **GTCP_setSingleTouchMode** ()
- void **GTCP_setMultiTouchMode** (uint8_t maxTouches)
- 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_customSwitchModeSingle** (uint8_t channel, uint16_t switchX, uint16_t switchY, uint16_t sizeX, uint16_t sizeY)
- void **GTCP_customSwitchModeMultiple** (uint8_t channel, uint8_t switchNum)
- void **GTCP_customSwitchParam** (uint16_t switchX, uint16_t switchY, uint16_t sizeX, uint16_t sizeY)
- void **GTCP_IOPortSetting** (uint8_t portNumber, uint8_t portSetting)
- void **GTCP_IOPortOutput** (uint8_t portNumber, uint8_t portValue)
- void **GTCP_IOPortInput** (uint8_t portNumber)
- void **GTCP_setSingleMemorySW** (uint8_t memorySW, uint8_t data)
- void **GTCP_setMultipleMemorySwitchHeader** (uint8_t numSettings)
- void **GTCP_setMemorySwitch** (uint8_t switchNumber, uint8_t data)
- void **GTCP_getSingleMemorySWData** (uint8_t switchNum)
- void **GTCP_getMultipleMemorySWData** (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_enterUserSetupMode** ()
- void **GTCP_endUserSetupMode** ()
- void **GTCP_FROMUserFontDefinition** (uint8_t table, uint8_t *data)
- void **GTCP_RAMMacroDefineDelete** (unsigned int length, uint8_t *data)
- void **GTCP_FROMMacroDefineDelete** (uint8_t registrationNum, unsigned int length, 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** (uint32_t size, uint8_t memorySelect, uint32_t address, uint8_t *data)
- void **GTCP_memoryTransfer** (uint32_t size, uint8_t destMemory, uint32_t destAddress, uint8_t srcMemory, uint32_t srcAddress)

- void **GTCP_readMemory** (uint8_t size, uint8_t memorySelect, uint32_t address)
- float **GTCP_getBootVersion** ()
- float **GTCP_getFirmwareVersion** ()
- void **GTCP_getCharacterCodeInfo** ()
- void **GTCP_getLanguageType** ()
- void **GTCP_getMemoryChecksum** (uint8_t address, uint8_t length)
- uint32_t **GTCP_getFROM2MemoryChecksum** (uint32_t address, uint32_t size)
- void **GTCP_getProductType** ()
- void **GTCP_getHorizontalResolution** ()
- void **GTCP_getVerticalResolution** ()
- void **GTCP_getTSPName** ()
- void **GTCP_getTSPID** ()
- void **GTCP_getTouchGain** ()
- void **GTCP_getTouchThreshold** ()
- void **GTCP_touchSettingPackageDataStore** (uint8_t packageNum, uint8_t *data)
- void **GTCP_touchSettingPackageSelection** (uint8_t packageNum)
- void **setPowerSavingMode** (uint8_t wakeup)
- void **setTouchScanPeriod** (uint8_t period)
- void **GTCP_setDisplayOrientation** (int degrees)

## Variables

- **TFT display**
- enum **interfaceType interface**
- uint16_t **Xdots**
- uint16_t **Ydots**
- uint16_t **Max_Xdot**
- uint16_t **Max_Ydot**
- uint16_t **Max_Xdot_CurtWin**
- uint16_t **Max_Ydot_CurtWin**
- uint32_t **DispMemSize**
- uint32_t **Max_DispMemAddr**

---

## Macro Definition Documentation

**#define BOX_DRAW 1**

**#define BOXFILL_DRAW 2**

**#define BRIGHTNESS_LEVEL_MSW 5**

**#define CHARACTER_STYLE_MSW 12**

**#define CHARACTER_TABLE_12X24 3**

**#define CHARACTER_TABLE_12X24_SIZE 4608**

**#define CHARACTER_TABLE_16X32 4**

**#define CHARACTER_TABLE_16X32_SIZE 8192**

**#define CHARACTER_TABLE_6X8 1**

**#define CHARACTER_TABLE_6X8_SIZE 768**

**#define CHARACTER_TABLE_8X16 2**

**#define CHARACTER_TABLE_8X16_SIZE 2048**

**#define CHARACTER_TABLE_TYPE_MSW 1**

**#define DISPLAY_ORIENTATION_MSW 6**

**#define DOWNLOAD_16X16_CHARACTER_RESTORE_MSW 18**

**#define DOWNLOAD_6X8_CHARACTER_RESTORE_MSW 16**

**#define DOWNLOAD_8X16_CHARACTER_RESTORE_MSW 17**

**#define FONT_MAGNIFICATION_X_MSW 10**

**#define FONT_MAGNIFICATION_Y_MSW 11**

**#define FONT_SIZE_MSW 8**

**#define FROM_MACRO_EXECUTE_AT_POWER_ON_MSW 19**

**#define GPIO0 0b00000001**

**#define GPIO1 0b00000010**

**#define GPIO10 0b00000100**

**#define GPIO11 0b00001000**

**#define GPIO12 0b00010000**

**#define GPIO13 0b00100000**

**#define GPIO14 0b01000000**

**#define GPIO15 0b10000000**

**#define GPIO16 0b00000001**

**#define GPIO17 0b00000010**

**#define GPIO18 0b00000100**

**#define GPIO19 0b00001000**

**#define GPIO2 0b00000100**

**#define GPIO20 0b00010000**

**#define GPIO21 0b00100000**

**#define GPIO22 0b01000000**

**#define GPIO23 0b10000000**

```
#define GPIO24   0b00000001
#define GPIO25   0b00000010
#define GPIO3   0b00001000
#define GPIO4   0b00010000
#define GPIO5   0b00100000
#define GPIO6   0b01000000
#define GPIO7   0b10000000
#define GPIO8   0b00000001
#define GPIO9   0b00000010
#define GTCP_PORT0   0
#define GTCP_PORT1   1
#define GTCP_PORT2   2
#define GTCP_PORT3   3
#define HORIZONTAL_SCROLL_SPEED_MSW   2
#define IMAGE_FORMAT_12BIT   0x8C
#define IMAGE_FORMAT_16BIT   0x90
#define IMAGE_FORMAT_16BIT_HS   0x91
#define IMAGE_FORMAT_24BIT   0x98
#define IMAGE_FORMAT_6BIT   0x86
#define IMAGE_FORMAT_BMP   0xF0
#define IMAGE_FORMAT_MONOCHROME   0x81
#define INTERNATIONAL_FONT_SET_MSW   0
#define LINE_DRAW   0
#define MACRO_END_CLEAR_SCREEN_MSW   54
#define MACRO_END_CODE_ENABLE_DISABLE_MSW   52
#define MACRO_END_CODE_MSW   53
#define MAX_MATRIX_SWITCHES_X   0x10
#define MAX_MATRIX_SWITCHES_Y   0x10
#define PEN_BACKGROUND_COLOR   0
#define PEN_CHARACTER_COLOR   1
#define REVERSE_DISPLAY_MSW   3
#define TOUCH_COORDINATE_RELEASED   0x10
#define TOUCH_COORDINATE_TOUCHED   0x11
#define TOUCH_CUSTOMSWITCH_RELEASED   0x30
#define TOUCH_CUSTOMSWITCH_TOUCHED   0x31
#define TOUCH_GAIN_MSW   58
#define TOUCH_HEADER   0x10
#define TOUCH_MATRIXSWITCH_RELEASED   0x20
#define TOUCH_MATRIXSWITCH_TOUCHED   0x21
#define TOUCH_PACKAGE_SELECTION_MSW   63
```

**#define TOUCH_SCAN_PERIOD_MSW 61**

**#define TOUCH_SENSITIVITY_SELECTION_MSW 62**

**#define TOUCH_THRESHOLD_MSW 59**

**#define TOUCH_TRANSMIT_MODE_MSW 60**

**#define TWO_BYTE_CHARACTER_MSW 9**

**#define TWO_BYTE_CHARACTER_TYPE_MSW 13**

**#define UART_BAUDRATE_MSW 48**

**#define UART_PARITY_MSW 49**

**#define WRITE_MIXTURE_DISPLAY_MODE_MSW 4**

**#define WRITE_SCREEN_MODE_MSW 7**

---

## Enumeration Type Documentation

**enum interfaceType**

**Enumerator:**

| | |
|---|---|
| UART | |
| SPI | |
| I2C | |

---

## Variable Documentation

**TFT display**

**uint32_t DispMemSize**

**enum interfaceType interface**

**uint32_t Max_DispMemAddr**

**uint16_t Max_Xdot**

**uint16_t Max_Xdot_CurtWin**

**uint16_t Max_Ydot**

**uint16_t Max_Ydot_CurtWin**

**uint16_t Xdots**

**uint16_t Ydots**

# GTCP_I2C.c File Reference

```
#include "GTCP_I2C.h"
```

## Functions

- void **initI2C** ()
- void **resetUSCIStateMachine_I2C** ()
- void **initUSCIStateMachine_I2C** ()
- void **configureI2C** ()
- void **configureI2CGPIO** ()
- void **checkStop** ()
- void **checkStart** ()
- void **sendWriteStart** ()
- void **sendReadStart** ()
- void **sendStop** ()
- void **RX_Int** (char enable)
- void **TX_Int** (char enable)
- void **readI2C** (uint8_t length)
- void **readMultipleI2CNI** ()
- void **writeI2C** (unsigned char data)
- void **writeMultipleI2C** (unsigned char *data)
- void **writeArrayI2C** (unsigned char *data, int size)

---

## Function Documentation

### void checkStart ()

Check if the start condition was sent to the slave device. Blocking function if the start condition has not been sent.

**Returns**

none

### void checkStop ()

Check if the stop condition was sent to the slave device. Blocking function if a stop condition has not been sent.

**Returns**

none

### void configureI2C ()

Configure registers for I2C interface.

**Returns**

none

### void configureI2CGPIO ()

Configure GPIO pins for the I2C interface.

**Returns**

none

### void initI2C ()

Initialize I2C communication on the MSP430.

Wiring Reference | MSP430F5529:

- SBUSY(MBUSY)(4) -> P1.6
- TRDY(7) -> P2.7
- SCL(2) -> P4.2
- SDA(3) -> P4.1
- RESET(1) -> P1.5

Wiring Reference | MSP430FR6989:

- SBUSY(MBUSY)(4) -> P1.4
- TRDY(7) -> P1.5
- SCL(2) -> P4.1
- SDA(3) -> P4.0
- RESET(1) -> P2.2

#### Returns

none

### void initUSCIStateMachine_I2C ()

Release USCI reset state. Can also be seen as USCI initialize.

#### Returns

none

### void readI2C (uint8_t *length*)

Read data from the module using the RX interrupt.

#### Parameters

| | |
|---|---|
| *length* | Number of bytes to read. |

#### Returns

none

### void readMultipleI2CNI ()

Read data from the module without interrupts. Read data in to RXDataArray global array. Only works with MSP430F5529.

#### Returns

The touch data read from the module.

### void resetUSCIStateMachine_I2C ()

Put USCI in reset state.

#### Returns

none

### void RX_Int (char *enable*)

Enable or disable the receive interrupt.

**Parameters**

| | |
|---|---|
| *enable* | Enable/disable receive interrupt. |

**Returns**

none

### void sendReadStart ()

Make sure the receiver mode is set and send a start condition.

**Returns**

none

### void sendStop ()

Send a stop condition to the slave device.

**Returns**

none

### void sendWriteStart ()

Set the host as a transmitter and send a start condition.

**Returns**

none

### void TX_Int (char *enable*)

Enable or disable the transmit interrupt.

Currently not used.

**Parameters**

| | |
|---|---|
| *enable* | Enable/disable transmit interrupt. |

**Returns**

none

### void writeArrayI2C (unsigned char * *data*, int *size*)

Send an array of data to the slave device.

**Parameters**

| | |
|---|---|
| *data* | Data to send. |
| *size* | Size of data array. |

**Returns**

none

### void writeI2C (unsigned char *data*)

Send a byte of data to the slave device.

**Parameters**

| | |
|---|---|
| *data* | Data byte to write. |

**Returns**

none

### void writeMultipleI2C (unsigned char * *data*)

Send multiple bytes of data to the slave device. Strings work well with this function. Does not work well with commands.

### Parameters

| | |
|---|---|
| *data* | String data to send. |

### Returns

none

# GTCP_I2C.h File Reference

```
#include <msp430f5529.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include "Interface.h"
```

## Functions

- void **initI2C** ()
- void **resetUSCIStateMachine_I2C** ()
- void **initUSCIStateMachine_I2C** ()
- void **configureI2C** ()
- void **configureI2CGPIO** ()
- void **checkStop** ()
- void **checkStart** ()
- void **sendWriteStart** ()
- void **sendReadStart** ()
- void **sendStop** ()
- void **RX_Int** (char enable)
- void **TX_Int** (char enable)
- void **readI2C** (uint8_t length)
- void **readMultipleI2CNI** ()
- void **writeI2C** (unsigned char data)
- void **writeMultipleI2C** (unsigned char *data)
- void **writeArrayI2C** (unsigned char *data, int size)
- __interrupt void **USCI_B1_ISR** (void)

## Variables

- unsigned char * **PTxData**
- unsigned char * **PRxData**
- int **TXByteCtr**
- char **TXData**
- int **RXByteCtr**
- char **RXData**
- unsigned char **RXDataArray** [**RX_BUFFER_SIZE**]
- int **TRDYFlag**

---

## Variable Documentation

**unsigned char* PRxData**

**unsigned char* PTxData**

**int RXByteCtr**

**char RXData**

**unsigned char RXDataArray[RX_BUFFER_SIZE]**

**int TRDYFlag**

**int TXByteCtr**

**char TXData**

# GTCP_SPI.c File Reference

```
#include "GTCP_SPI.h"
```

## Functions

- void **initSPI** ()
- void **resetUSCIStateMachine_SPI** ()
- void **initUSCIStateMachine_SPI** ()
- void **configureSPIGPIO** ()
- void **configureSPI** ()
- void **setCS** ()
- void **clearCS** ()
- void **checkMBUSY** ()
- void **checkTXBufEmpty** ()
- void **checkRXBufEmpty** ()
- void **beginWriteSPI** ()
- void **putTXBUF** (uint8_t data)
- uint8_t **readRXBUF** ()
- void **endSPI** ()
- void **writeSPI** (char data)
- void **writeStringSPI** (unsigned char *data)
- void **writeArraySPI** (uint8_t *data, int size)
- void **startSPIRead** ()
- void **readSPI** (unsigned char *data)
- int **readSPIStatus** ()

---

## Function Documentation

### void beginWriteSPI ()

Starts SPI write communication.

**Returns**

none

### void checkMBUSY ()

Wait until MBUSY is cleared.

**Returns**

none

### void checkRXBufEmpty ()

Wait until SPI receive buffer is empty.

**Returns**

none

### void checkTXBufEmpty ()

Wait until SPI transmit buffer is empty.

**Returns**

none

### void clearCS ()

Clear chip select (CS).

**Returns**

none

### void configureSPI ()

### void configureSPIGPIO ()

### void endSPI ()

Ends SPI communication

**Returns**

none

### void initSPI ()

Initializes the SPI interface for MSP430 to communicate with a Noritake GTCP series module.

Wiring Reference | MSP430F5529:

- CS(5) -> P2.6
- MBUSY(4) -> P4.0
- MOSI(3) -> P3.0
- MISO(6) -> P3.1
- SCK(2) -> P3.2
- RESET(1) -> P1.5

Wiring Reference | MSP430FR6989:

- CS(5) -> P2.5
- MBUSY(4) -> P2.7
- MOSI(3) -> P1.6
- MISO(6) -> P1.7
- SCK(2) -> P1.4
- RESET(1) -> P1.5

**Returns**

none

### void initUSCIStateMachine_SPI ()

Release USCI reset state. Can also be seen as USCI initialize.

**Returns**

none

### void putTXBUF (uint8_t *data*)

Put one byte of data into the transmit buffer.

**Parameters**

| | |
|---|---|
| *data* | Data to put in transmit buffer. |

**Returns**

none

### uint8_t readRXBUF ()

Read the current receive buffer data byte.

**Returns**

Receive buffer data byte.

### void readSPI (unsigned char * *data*)

Read bytesToRead number of bytes and store them in the passed-in pointer location.

**Parameters**

| | |
|---|---|
| *data* | Receive data array pointer. |

**Returns**

none

### int readSPIStatus ()

Reads the status byte from the device.

**Returns**

The number of bytes to be read from the device.

### void resetUSCIStateMachine_SPI ()

Put USCI in reset state.

**Returns**

none

### void setCS ()

Set chip select (CS) HIGH.

**Returns**

none

### void startSPIRead ()

Start the read sequence for SPI.

**Returns**

none

### void writeArraySPI (uint8_t * *data*, int *size*)

Writes an array of data to the destination device. This function must be used after the SPI interface has started.

**Parameters**

| | |
|---|---|
| *data* | Data to write. |
| *size* | Number of bytes to write. |

**Returns**

none

**void writeSPI (char *data*)**

Writes a character to the destination device. This method must be used after the SPI interface has started.

**Parameters**

| | |
|---|---|
| *data* | The character to be transmitted. |

**Returns**

none

**void writeStringSPI (unsigned char * *data*)**

Writes a string of characters to the destination device. This function must be used after the SPI interface has started.

**Parameters**

| | |
|---|---|
| *data* | The data to be transmitted. |

**Returns**

none

# GTCP_SPI.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <msp430f5529.h>
#include "Interface.h"
```

## Macros

- #define **READ_DELAY** 22
- #define **CS_DELAY** 15
- #define **WRITE_IDENTIFIER** 0x44
- #define **DATAREAD_IDENTIFIER** 0x54
- #define **STATUSREAD_IDENTIFIER** 0x58
- #define **MAX_BYTES_TO_READ** 63

## Functions

- void **initSPI** ()
- void **resetUSCIStateMachine_SPI** ()
- void **initUSCIStateMachine_SPI** ()
- void **configureSPIGPIO** ()
- void **configureSPI** ()
- void **setCS** ()
- void **clearCS** ()
- void **checkMBUSY** ()
- void **checkTXBufEmpty** ()
- void **checkRXBufEmpty** ()
- void **beginWriteSPI** ()
- void **putTXBUF** (uint8_t data)
- uint8_t **readRXBUF** ()
- void **endSPI** ()
- void **writeSPI** (char data)
- void **writeStringSPI** (unsigned char *data)
- void **writeArraySPI** (uint8_t *data, int size)
- void **startSPIRead** ()
- void **readSPI** (unsigned char *data)
- int **readSPIStatus** ()

## Variables

- int **dataRead**
- int **statusRead**
- int **bytesToRead**

---

## Macro Definition Documentation

**#define CS_DELAY 15**

**#define DATAREAD_IDENTIFIER 0x54**

**#define MAX_BYTES_TO_READ 63**

**#define READ_DELAY 22**

**#define STATUSREAD_IDENTIFIER 0x58**

**#define WRITE_IDENTIFIER 0x44**

---

## Variable Documentation

**int bytesToRead**

**int dataRead**

**int statusRead**

# GTCP_UART.c File Reference

```
#include "GTCP_UART.h"
```

## Functions

- void **initUART** ()
- void **resetUSCIStateMachine_UART** ()
- void **initUSCIStateMachine_UART** ()
- void **configureUART** ()
- void **configureUARTGPIO** ()
- void **setHBUSY** ()
- void **clearHBUSY** ()
- void **waitMBUSYClear** ()
- void **waitTXBUFClear** ()
- void **waitRXBUFByte** ()
- void **writeUART** (uint8_t data)
- void **writeStringUART** (unsigned char *data)
- void **writeArrayUART** (unsigned char *data, int size)
- uint8_t **readUART** ()
- void **readUARTMultiple** (int length)

---

## Function Documentation

### void clearHBUSY ()

Clear the HBUSY signal.

**Returns**

none

### void configureUART ()

Configure UART register settings.

**Returns**

none

### void configureUARTGPIO ()

Configure GPIO settings for UART.

**Returns**

none

### void initUART ()

Initializes the UART interface for MSP430 to communicate with a Noritake GT-CP series modules.

Wiring Reference | MSP430F5529:

- RESET(Pin 1) -> P1.5
- RxD(Pin 3) -> P3.3 (TX)
- MBUSY/DTR(Pin 4) -> P4.0
- HBUSY/DSR(Pin 5) -> P2.6
- TxD(Pin 6) -> P3.4 (RX)

Wiring Reference | MSP430FR6989:

- RESET(Pin 1) -> P1.5
- RxD(Pin 3) -> P2.0 (TX)
- MBUSY/DTR(Pin 4) -> P4.0
- HBUSY/DSR(Pin 5) -> P2.6
- TxD(Pin 6) -> P2.1 (RX)

**Returns**

none

### void initUSCIStateMachine_UART ()

Release USCI reset state. Can also be seen as USCI initialize.

**Returns**

none

### uint8_t readUART ()

Reads in a byte of data.

**Returns**

The byte of data read from the device.

### void readUARTMultiple (int *length*)

Read multiple bytes in.

**Parameters**

| | |
|---|---|
| *length* | Number of bytes to read. |

**Returns**

none

### void resetUSCIStateMachine_UART ()

Put USCI in reset state.

**Returns**

none

### void setHBUSY ()

Set the HBUSY signal HIGH.

**Returns**

none

### void waitMBUSYClear ()

Wait until MBUSY is clear(0) or LOW.

**Returns**

none

### void waitRXBUFByte ()

Wait until a complete character is in the receive buffer.

**Returns**

none

**void waitTXBUFClear ()**

Wait until the transmit buffer is clear/empty.

**Returns**

none

**void writeArrayUART (unsigned char * *data*, int *size*)**

Write an array of data to the display module.

**Parameters**

| | |
|---|---|
| *data* | The data array to write. |
| *size* | The size of the data array. |

**Returns**

none

**void writeStringUART (unsigned char * *data*)**

Sends an array of bytes.

**Parameters**

| | |
|---|---|
| *data* | The text string to be transmitted. |

**Returns**

none

**void writeUART (uint8_t *data*)**

Sends a byte of data.

**Returns**

none

# GTCP_UART.h File Reference

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

## Functions

- void **initUART** ()
- void **resetUSCIStateMachine_UART** ()
- void **initUSCIStateMachine_UART** ()
- void **configureUART** ()
- void **configureUARTGPIO** ()
- void **setHBUSY** ()
- void **clearHBUSY** ()
- void **waitMBUSYClear** ()
- void **waitTXBUFClear** ()
- void **waitRXBUFByte** ()
- void **writeUART** (uint8_t data)
- void **writeStringUART** (unsigned char *data)
- void **writeArrayUART** (unsigned char *data, int size)
- uint8_t **readUART** ()
- void **readUARTMultiple** (int length)

---

# Interface.c File Reference

```
#include "Interface.h"
#include <msp430f5529.h>
```

## Functions

- void **initRESET** ()
- void **RESET** ()
- int **arraySize** (unsigned char *data)

---

## Function Documentation

### void initRESET ()

Initialize P1.5 for RESET output.

**Returns**

none

### void RESET ()

Perform a hardware RESET.

**Returns**

none

### int arraySize (unsigned char * *data*)

Find the size of the given array.

**Parameters**

| | |
|---|---|
| *data* | Array to inspect. |

**Returns**

Size of array.

# Interface.h File Reference

## Macros

- #define **RX_BUFFER_SIZE** 258
- #define **RESET_DELAY** 1000
- #define **DISPLAY_POWER_DELAY** 1600000

## Functions

- void **initRESET** ()
- void **RESET** ()
- int **arraySize** (unsigned char *data)

## Variables

- unsigned char **RXDataArray** [**RX_BUFFER_SIZE**]

---

## Macro Definition Documentation

**#define DISPLAY_POWER_DELAY 1600000**

**#define RESET_DELAY 1000**

**#define RX_BUFFER_SIZE 258**

---

## Variable Documentation

**unsigned char RXDataArray[RX_BUFFER_SIZE]** `[extern]`

# Index

INDEX