Graphiti_API_Library

1.0.0

Generated by Doxygen 1.14.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

DisplayStatusEvent_C . . . . ??
Draw . . . . ??
Graphiti_API::DrawEvent . . . . ??
DrawEvent_C . . . . ??
Graphiti_API . . . . ??
    GraphitiExtension . . . . ??
Graphiti_API_HID . . . . ??
Graphiti_API_VCP . . . . ??
GraphitiConnection . . . . ??
    GraphitiConnectionVCP . . . . ??
GraphitiHandle . . . . ??
KeyEvent_C . . . . ??
Pin . . . . ??
Graphiti_API::PinInfo . . . . ??
PinInfo_C . . . . ??
ThreadSafeQueue< T > . . . . ??

Chapter 2

Class Index

2.1 Class List

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

DisplayStatusEvent_C . . . . . ??
Draw
    Holds draw Event informaiton . . . . . ??
Graphiti_API::DrawEvent . . . . . ??
DrawEvent_C . . . . . ??
Graphiti_API . . . . . ??
Graphiti_API_HID . . . . . ??
Graphiti_API_VCP . . . . . ??
GraphitiConnection . . . . . ??
GraphitiConnectionVCP . . . . . ??
GraphitiExtension
    Class to improve the ease of use of the Graphiti Graphiti_API can be used with or without GraphitExtension GraphitiExtension can also be used as a reference to how the Graphiti_API can be used . . . . . ??
GraphitiHandle . . . . . ??
KeyEvent_C . . . . . ??
Pin
    Holds pin information . . . . . ??
Graphiti_API::PinInfo . . . . . ??
PinInfo_C . . . . . ??
ThreadSafeQueue< T > . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

lib/build/CMakeFiles/4.0.2/CompilerIdC/CMakeCCompilerId.c . . . . ??
lib/build/CMakeFiles/4.0.2/CompilerIdCXX/CMakeCXXCompilerId.cpp . . . . ??
lib/include/Graphiti/API.hpp . . . . ??
lib/include/Graphiti/API_HID.hpp . . . . ??
lib/include/Graphiti/API_VCP.hpp . . . . ??
lib/include/Graphiti/Extension.hpp . . . . ??
lib/include/Graphiti/ThreadSafeQueue.hpp . . . . ??
lib/include/Graphiti/Connection/Connection.hpp . . . . ??
lib/include/Graphiti/Connection/Connection_HID.hpp . . . . ??
lib/include/Graphiti/Connection/Connection_VCP.hpp . . . . ??
lib/include/Graphiti/CWrapper/capi.hpp . . . . ??
lib/src/API.cpp . . . . ??
lib/src/API_HID.cpp . . . . ??
lib/src/API_VCP.cpp . . . . ??
lib/src/Extension.cpp . . . . ??
lib/src/Connection/Connection_HID.cpp . . . . ??
lib/src/Connection/Connection_VCP.cpp . . . . ??
lib/src/CWrapper/capi.cpp . . . . ??

Chapter 4

Class Documentation

4.1 DisplayStatusEvent_C Struct Reference

```
#include <capi.hpp>
```

Public Attributes

- uint8_t * data
- int length
- bool has_value

4.1.1 Member Data Documentation

4.1.1.1 data

```
uint8_t* DisplayStatusEvent_C::data
```

4.1.1.2 has_value

```
bool DisplayStatusEvent_C::has_value
```

4.1.1.3 length

```
int DisplayStatusEvent_C::length
```

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

- lib/include/Graphiti/CWrapper/capi.hpp

4.2 Draw Struct Reference

Holds draw Event informaiton.

4.2.1 Detailed Description

Holds draw Event informaiton.

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

- lib/include/Graphiti/API.hpp

4.3 Graphiti_API::DrawEvent Struct Reference

```
#include <API.hpp>
```

Public Attributes

- int length
  length - amount of pins
- std::vector< PinInfo > pins
  pins - pins of PinInfo

4.3.1 Member Data Documentation

4.3.1.1 length

```
int Graphiti_API::DrawEvent::length
```

length - amount of pins

4.3.1.2 pins

```
std::vector<PinInfo> Graphiti_API::DrawEvent::pins
```

pins - pins of PinInfo

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

- lib/include/Graphiti/API.hpp

4.4 DrawEvent_C Struct Reference

```
#include <capi.hpp>
```

Public Attributes

- PinInfo_C * pins
- int length
- bool has_value

4.4.1 Member Data Documentation

4.4.1.1 has_value

```
bool DrawEvent_C::has_value
```

4.4.1.2 length

```
int DrawEvent_C::length
```

4.4.1.3 pins

```
PinInfo_C* DrawEvent_C::pins
```

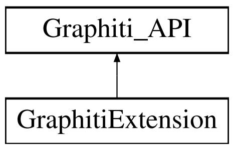
The documentation for this struct was generated from the following file:

- lib/include/Graphiti/CWrapper/capi.hpp

4.5 Graphiti_API Class Reference

```
#include <API.hpp>
```

Inheritance diagram for Graphiti_API:



Classes

- struct DrawEvent
- struct PinInfo

Public Member Functions

- Graphiti_API ()

    Construct a new Graphiti_API object.
- Graphiti_API (GraphitiConnection ∗connection)

    Construct a new Graphiti_API object with connection.
- ∼Graphiti_API ()

    Destroy the Graphiti_API object.
- void setConnection (GraphitiConnection ∗connection)

    Set Connection.
- void startResponseThread ()

    Start Response Thread.
- void stopResponseThread ()

    Stop Response Thread.
- std::optional< std::string > getNextOutputEvent ()

    Get the Next Output Event object.
- std::optional< std::vector< uint8_t > > getNextDisplayStatusEvent ()

    Get the Next Display Status Event object.
- std::optional< std::set< std::string > > getNextKeyEvent ()

    Get the Next Key Event object.
- std::optional< std::string > getNextGestureEvent ()

    Get the Next Gesture Event object.
- std::optional< DrawEvent > getNextDrawEvent ()

    Get the Next Draw Event object.
- void sendACK ()

    4.1 ACK request
- void sendNACK ()

    4.1 NACK request
- void getSoftwareVersion ()

    4.2.1 Get Software Version
- void getHardWareVersion ()

    4.2.2 Get Hardware Version
- void getSerialNumber ()

    4.2.3 Get Unit Serial Number
- void getBatteryStatus ()

    4.2.4 Get Battery Status
- void getResolutionInformation ()

    4.2.5 Get Resolution Information
- void getDeviceOrientation ()

    4.2.6 Get Device Orientation
- void getHeightInformation ()

    4.2.7 Get Height Information
- void getDeviceName ()

    4.2.8 Get Device Name
- void updateDisplay (std::vector< uint8_t > screen)

    4.3 Display Access Commands
- void setDisplay ()

    4.3.2 Set Display
- void clearDisplay ()

    4.3.2 Clear Display
- void updateSinglePixel (uint8_t row, uint8_t column, uint8_t height, uint8_t blinkRate)

4.3.3 Update Single pixel on Display
- void updateSingleRow (uint8_t row, std::vector< uint8_t > rowData)

  4.3.4 Update Single Row on Display
- void updateSingleColumn (uint8_t column, std::vector< uint8_t > columnData)

  4.3.5 Update Single Column on Display
- void getAllPixelsPositionStatus ()

  4.3.6 Get All Pixels' Position Status
- void getSinglePixelPositionStatus (uint8_t row, uint8_t column)

  4.3.7 Get Single Pixel Position Status
- void getSingleRowPixelPositionStatus (uint8_t row)

  4.3.8 Get Single Row Pixels Position Status
- void getSingleColumnPixelPositionStatus (uint8_t column)

  4.3.9 Get Single Column Pixels Position Status
- void showMessage (std::string message, std::vector< uint8_t > selectFlags, uint8_t blinkRate)

  4.3.10 Show Message on the Device
- void setCursor (uint8_t row, uint8_t column, uint8_t height, uint8_t length, uint8_t blinkRate)

  4.3.11 Set Cursor
- void sendImageInterruptible (std::string name, const std::string &filepath)

  4.3.12 Send Image (Interruptible)
- void sendImageBlocking (std::string name, const std::string &filepath)

  4.3.13 Send Image (Blocking)
- void setKeyEvent (bool enabled)

  4.4.1 Set Key Event
- void setTouchEvent (bool enabled)

  4.5.1 Set Touch Event
- void getLastTouchPointStatus ()

  4.5.2 Get Last Touch Point Status
- void vibratorControlCommand (uint8_t frequencyRange, uint8_t dutyCycle, uint16_t duration)

  4.6 Vibrator Control Command
- void getDateAndTime ()

  4.7.1 Get Date and Time
- void setDateAndTime (uint8_t day, uint8_t month, uint16_t year, uint8_t hour, uint8_t minute, uint8_t second)

  4.7.2 Set Date and Time

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Graphiti_API() [1/2]

```
Graphiti_API::Graphiti_API ()
```

Construct a new Graphiti_API object.

Construct a new Graphiti_API::Graphiti_API object Without connection.

4.5.1.2 Graphiti_API() [2/2]

```
Graphiti_API::Graphiti_API (
            GraphitiConnection * connection)
```

Construct a new Graphiti_API object with connection.

Construct a new Graphiti_API::Graphiti_API object.

Parameters

	connection	connection object for reading device output

Sets up maps for use in Graphiti_API

Parameters

	connection	connection for GraphitiConnection read calls

4.5.1.3 ∼Graphiti_API()

```
Graphiti_API::∼Graphiti_API ()
```

Destroy the Graphiti_API object.

Destroy the Graphiti_API::Graphiti_API object.

Stops thread responses

4.5.2 Member Function Documentation

4.5.2.1 clearDisplay()

```
void Graphiti_API::clearDisplay ()
```

4.3.2 Clear Display

The 'Clear Display' command is used to set or reset all the pins of the display to the lowest (in line with the surface) position.

4.5.2.2 getAllPixelsPositionStatus()

```
void Graphiti_API::getAllPixelsPositionStatus ()
```

4.3.6 Get All Pixels' Position Status

The 'Get All Pixels Position Status' command is used to retrieve the present status (position) of each pixel of the entire display (40x60) in a single response.

4.5.2.3 getBatteryStatus()

```
void Graphiti_API::getBatteryStatus ()
```

4.2.4 Get Battery Status

The 'Get Battery Status' command retrieves the current battery state of the device including the charging status.

4.5.2.4 getDateAndTime()

```
void Graphiti_API::getDateAndTime ()
```

4.7.1 Get Date and Time

The 'Get Date and Time' command is used to get the date and time information of the device. The device provides clock information in the 24-hour format.

4.5.2.5 getDeviceName()

```
void Graphiti_API::getDeviceName ()
```

4.2.8 Get Device Name

The 'Get Device Name' command is used to retrieve the name of the device.

4.5.2.6 getDeviceOrientation()

```
void Graphiti_API::getDeviceOrientation ()
```

4.2.6 Get Device Orientation

The 'Get Device Orientation' command retrieves the current orientation of the device. Presently the device supports landscape orientation only.

4.5.2.7 getHardWareVersion()

```
void Graphiti_API::getHardWareVersion ()
```

4.2.2 Get Hardware Version

The 'Get Hardware Version' command retrieves the current version of the device hardware.

4.5.2.8 getHeightInformation()

```
void Graphiti_API::getHeightInformation ()
```

4.2.7 Get Height Information

The 'Get Height Information' command retrieves information about the number of height levels supported by the device for all the pins.

4.5.2.9 getLastTouchPointStatus()

```
void Graphiti_API::getLastTouchPointStatus ()
```

4.5.2 Get Last Touch Point Status

The 'Get Last Touch Point Status' is used to get the height of the pin which was last touched. To get last touch point status, you must first touch the pin and then transmit command from the host.

4.5.2.10 getNextDisplayStatusEvent()

```
std::optional< std::vector< uint8_t > > Graphiti_API::getNextDisplayStatusEvent ()
```

Get the Next Display Status Event object.

Returns

std::optional<std::vector<uint8_t>> optional event (nullopt when empty)

4.5.2.11 getNextDrawEvent()

```
std::optional< Graphiti_API::DrawEvent > Graphiti_API::getNextDrawEvent ()
```

Get the Next Draw Event object.

Returns

std::optional<DrawEvent> draw event option

null output when no event

4.5.2.12 getNextGestureEvent()

```
std::optional< std::string > Graphiti_API::getNextGestureEvent ()
```

Get the Next Gesture Event object.

Returns

std::string gesture

4.5.2.13 getNextKeyEvent()

```
std::optional< std::set< std::string > > Graphiti_API::getNextKeyEvent ()
```

Get the Next Key Event object.

Returns

std::optional<std::set<std::string>> string option of key event

4.5.2.14 getNextOutputEvent()

```
std::optional< std::string > Graphiti_API::getNextOutputEvent ()
```

Get the Next Output Event object.

Returns

std::optional<std::string> optional string of event

4.5.2.15 getResolutionInformation()

```
void Graphiti_API::getResolutionInformation ()
```

4.2.5 Get Resolution Information

The 'Get Resolution Information' command retrieves the details of horizontal and vertical resolution supported by the device.

4.5.2.16 getSerialNumber()

```
void Graphiti_API::getSerialNumber ()
```

4.2.3 Get Unit Serial Number

The 'Get Unit Serial Number' command retrieves the serial number of the connected device.

4.5.2.17 getSingleColumnPixelPositionStatus()

```
void Graphiti_API::getSingleColumnPixelPositionStatus (
            uint8_t column)
```

4.3.9 Get Single Column Pixels Position Status

The 'Get Single Column Pixels Position Status' command is used to retrieve the present status of each pixel in the requested column.

Parameters

	column	column ID 1 to 60

4.5.2.18 getSinglePixelPositionStatus()

```
void Graphiti_API::getSinglePixelPositionStatus (
            uint8_t row,
            uint8_t column)
```

4.3.7 Get Single Pixel Position Status

The 'Get Single Pixel Position Status' command is used to retrieve the present status of the requested pixel.

Parameters

	row	row ID 1 to 40
	column	column ID 1 to 60

4.5.2.19 getSingleRowPixelPositionStatus()

```
void Graphiti_API::getSingleRowPixelPositionStatus (
            uint8_t row)
```

4.3.8 Get Single Row Pixels Position Status

The 'Get Single Row Pixels Position Status' command is used to retrieve the present status of each pixel in the requested row.

Parameters

row	row ID 1 to 40

4.5.2.20 getSoftwareVersion()

```
void Graphiti_API::getSoftwareVersion ()
```

4.2.1 Get Software Version

The 'Get Software Version' command retrieves the version of the current firmware in the device.

4.5.2.21 sendACK()

```
void Graphiti_API::sendACK ()
```

4.1 ACK request

You can send this command to the device to inform that your received data is correct.

4.5.2.22 sendImageBlocking()

```
void Graphiti_API::sendImageBlocking (
            std::string name,
            const std::string & filepath)
```

4.3.13 Send Image (Blocking)

This command is used to display the actual image such as a BMP, PNG, or JPG file on the device. (Read Graphiti API Spec for more details).

Parameters

name	name of image including file extension
filepath	filepath to image(including name and file extension)

This command is used to display the actual image such as a BMP, PNG, or JPG file on the device. (Read Graphiti API Spec for more details)

Parameters

name	name of image with extension (.jpg)
size	size of image
filepath	exact or relative path to image

4.5.2.23 sendImageInterruptible()

```
void Graphiti_API::sendImageInterruptible (
            std::string name,
            const std::string & filepath)
```

4.3.12 Send Image (Interruptible)

This command is used to send and display an image file, such as a BMP, PNG, or JPG file on the device. It is possible to terminate the data transfer of an image in this API and send a new image. (Read Graphiti API Spec for more details)

Parameters

name	name of image including file extension
filepath	filepath to image(including name and file extension)

4.5.2.24 sendNACK()

```
void Graphiti_API::sendNACK ()
```

4.1 NACK request

Whenever the checksum error is detected in the received data at the host side, you can send NACK to the device so that the device will send the same data again. The host application must send NACK command within 300ms of the previous response, otherwise the device will not serve it.

4.5.2.25 setConnection()

```
void Graphiti_API::setConnection (
            GraphitiConnection * connection)
```

Set Connection.

Set the Connection object for the Graphiti

Used in conjunction with constructor that does not set the connection object or otherwise

Parameters

connection	

4.5.2.26 setCursor()

```
void Graphiti_API::setCursor (
            uint8_t row,
            uint8_t column,
            uint8_t height,
            uint8_t length,
            uint8_t blinkRate)
```

4.3.11 Set Cursor

The 'Set Cursor' command is used to set the cursor on the display. Here, you need to give the cursor position and size of the cursor.

Parameters

row	row ID 1 to 40
column	column ID 1 to 60
height	height 1 to 4
length	length 1 to 60
blinkRate	blinkRate 0 to 50 (5 seconds)

4.5.2.27 setDateAndTime()

```
void Graphiti_API::setDateAndTime (
             uint8_t day,
             uint8_t month,
             uint16_t year,
             uint8_t hour,
             uint8_t minute,
             uint8_t second)
```

4.7.2 Set Date and Time

The 'Set Date and Time' command is used to set the device date and time in 24 hour format. The device will accept data in 24hr format.

Parameters

	day	day byte
	month	month byte
	year	year byte
	hour	hour byte
	minute	minute byte
	second	second byte

4.5.2.28 setDisplay()

```
void Graphiti_API::setDisplay ()
```

4.3.2 Set Display

The 'Set Display' command is used to set or reset all the pins of the display to the highest height position.

4.5.2.29 setKeyEvent()

```
void Graphiti_API::setKeyEvent (
             bool enabled)
```

4.4.1 Set Key Event

The 'Set Key Event' command is used to enable or disable the key press event. When enabled, you will be able to get the information of each key press of the device keypad. This information will include the key value and its event type. Note: Key cobinaitons that are reserved are: (5 + 6 + 8), (8 + Down), (7 + 8)

Parameters

	enabled	true when enabled, false when disabled

4.5.2.30 setTouchEvent()

```
void Graphiti_API::setTouchEvent (
             bool enabled)
```

4.5.1 Set Touch Event

The 'Set Touch Event' command is used to enable or disable the touch event. When this API is enabled, you will be able to get information of each touch event (after enabling the respective mode: either gesture mode or draw mode) on the touch panel. (See more info on Graphiti API Specification) API for touch coordinates is supported in Draw mode only as of Version 0.22.

Parameters

enabled	true when enabled, false when disabled

4.5.2.31 showMessage()

```
void Graphiti_API::showMessage (
            std::string message,
            std::vector< uint8_t > selectFlags,
            uint8_t blinkRate)
```

4.3.10 Show Message on the Device

Parameters

message	string message up to 20 characters
selectFlags	int array of flags of 0, 1, 2 for nothing, underlining, and display cursor (up to 20 flags)
blinkRate	blink rate 0 to 50 (5 seconds)

4.5.2.32 startResponseThread()

```
void Graphiti_API::startResponseThread ()
```

Start Response Thread.

Starts response loop to read from Graphiti output

4.5.2.33 stopResponseThread()

```
void Graphiti_API::stopResponseThread ()
```

Stop Response Thread.

Stops response loop that reads from Graphiti output

4.5.2.34 updateDisplay()

```
void Graphiti_API::updateDisplay (
            std::vector< uint8_t > screen)
```

4.3 Display Access Commands

The response to all the commands in this category is received only after the display has been updated. Heights vary from 0x00 to 0x04. Blinkrates vary from 0 - No blinking, 1 - 100 ms to 50 - 5 seconds

4.3.1 Update Display

The 'Update Display' command is used to configure the pins to the required height.

Parameters

screen	screen data of pixel heights and blinking rates

4.5.2.35 updateSingleColumn()

```
void Graphiti_API::updateSingleColumn (
            uint8_t column,
            std::vector< uint8_t > columnData)
```

4.3.5 Update Single Column on Display

The 'Update Single Column' command is used to configure a single column to a desired height position. To update the entire column, the pixel value and blinking rate for each pixel in a column (40 pixels) needs to be provided.

Parameters

column	column ID 1 to 60
columnData	column data with pixel value 0 to 4 height and blinking rate 0 to 50

4.5.2.36 updateSinglePixel()

```
void Graphiti_API::updateSinglePixel (
            uint8_t row,
            uint8_t column,
            uint8_t height,
            uint8_t blinkRate)
```

4.3.3 Update Single pixel on Display

The 'Update Single Pixel' command is used to configure a single pin to the desired height position and blink rate interval.

Parameters

row	row ID 1 to 40
column	column ID 1 to 60
height	height 0 to 4
blinkRate	blink rate 0 to 50 (5 seconds)

4.5.2.37 updateSingleRow()

```
void Graphiti_API::updateSingleRow (
            uint8_t row,
            std::vector< uint8_t > rowData)
```

4.3.4 Update Single Row on Display

The 'Update Single Row' command is used to configure a single row to a desired height position. To update the entire row, the pixel value and blinking rate for each pixel in a row (60 pixels) needs to be provided.

Parameters

row	row ID 1 to 40
rowData	

4.5.2.38 vibratorControlCommand()

```
void Graphiti_API::vibratorControlCommand (
            uint8_t frequencyRange,
            uint8_t dutyCycle,
            uint16_t duration)
```

4.6 Vibrator Control Command

This command is used to drive vibrators available in the device with different frequency, duty cycle and duration.

Parameters

frequencyRange	frequency 10 - 100 kHz
dutyCycle	duty cycle 40 - 100 percent
duration	duration 100 - 1000 ms

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

- lib/include/Graphiti/API.hpp
- lib/src/API.cpp

4.6 Graphiti_API_HID Class Reference

```
#include <API_HID.hpp>
```

Public Member Functions

- Graphiti_API_HID (unsigned short vid, unsigned short pid)
- ∼Graphiti_API_HID ()
- bool open ()
- void close ()
- bool write (const std::vector< unsigned char > &data)
- std::vector< unsigned char > read (size_t size)

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Graphiti_API_HID()

```
Graphiti_API_HID::Graphiti_API_HID (
            unsigned short vid,
            unsigned short pid)
```

4.6.1.2 ∼Graphiti_API_HID()

```
Graphiti_API_HID::∼Graphiti_API_HID ()
```

4.6.2 Member Function Documentation

4.6.2.1 close()

```
void Graphiti_API_HID::close ()
```

4.6.2.2 open()

```
bool Graphiti_API_HID::open ()
```

4.6.2.3 read()

```
std::vector< unsigned char > Graphiti_API_HID::read (
             size_t size)
```

4.6.2.4 write()

```
bool Graphiti_API_HID::write (
             const std::vector< unsigned char > & data)
```

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

- lib/include/Graphiti/API_HID.hpp
- lib/src/API_HID.cpp

4.7 Graphiti_API_VCP Class Reference

```
#include <API_VCP.hpp>
```

Public Member Functions

- Graphiti_API_VCP (const std::string &port)
- ∼Graphiti_API_VCP ()
- bool open ()
- void close ()
- bool write (const std::vector< unsigned char > &data)
- std::vector< unsigned char > read (size_t size)

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Graphiti_API_VCP()

```
Graphiti_API_VCP::Graphiti_API_VCP (
             const std::string & port)  [explicit]
```

4.7.1.2 ∼Graphiti_API_VCP()

```
Graphiti_API_VCP::∼Graphiti_API_VCP ()
```

4.7.2 Member Function Documentation

4.7.2.1 close()

```
void Graphiti_API_VCP::close ()
```

4.7.2.2 open()

```
bool Graphiti_API_VCP::open ()
```

4.7.2.3 read()

```
std::vector< unsigned char > Graphiti_API_VCP::read (
             size_t size)
```

4.7.2.4 write()

```
bool Graphiti_API_VCP::write (
             const std::vector< unsigned char > & data)
```

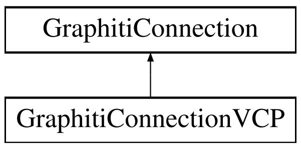
The documentation for this class was generated from the following files:

- lib/include/Graphiti/API_VCP.hpp
- lib/src/API_VCP.cpp

4.8 GraphitiConnection Class Reference

```
#include <Connection.hpp>
```

Inheritance diagram for GraphitiConnection:



Public Member Functions

- virtual ∼GraphitiConnection ()
- virtual bool open ()=0
- virtual void close ()=0
- virtual bool write (const std::vector< unsigned char > &data)=0
- virtual std::vector< unsigned char > read (size_t size)=0

4.8.1 Constructor & Destructor Documentation

4.8.1.1 ∼GraphitiConnection()

```
virtual GraphitiConnection::~GraphitiConnection ()  [inline], [virtual]
```

4.8.2 Member Function Documentation

4.8.2.1 close()

```
virtual void GraphitiConnection::close ()  [pure virtual]
```

Implemented in GraphitiConnectionVCP.

4.8.2.2 open()

```
virtual bool GraphitiConnection::open ()  [pure virtual]
```

Implemented in GraphitiConnectionVCP.

4.8.2.3 read()

```
virtual std::vector< unsigned char > GraphitiConnection::read (
             size_t size)  [pure virtual]
```

Implemented in GraphitiConnectionVCP.

4.8.2.4 write()

```
virtual bool GraphitiConnection::write (
             const std::vector< unsigned char > & data)  [pure virtual]
```

Implemented in GraphitiConnectionVCP.

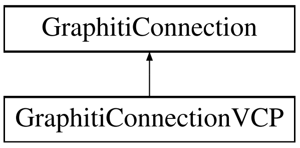
The documentation for this class was generated from the following file:

- lib/include/Graphiti/Connection/Connection.hpp

4.9 GraphitiConnectionVCP Class Reference

```
#include <Connection_VCP.hpp>
```

Inheritance diagram for GraphitiConnectionVCP:



Public Member Functions

- GraphitiConnectionVCP (const std::string &port)
- bool open ()
- void close ()
- bool write (const std::vector< unsigned char > &data)
- std::vector< unsigned char > read (size_t size)

Public Member Functions inherited from GraphitiConnection

- virtual ∼GraphitiConnection ()

4.9.1 Constructor & Destructor Documentation

4.9.1.1 GraphitiConnectionVCP()

```
GraphitiConnectionVCP::GraphitiConnectionVCP (
             const std::string & port)  [explicit]
```

4.9.2 Member Function Documentation

4.9.2.1 close()

```
void GraphitiConnectionVCP::close ()  [virtual]
```

Implements GraphitiConnection.

4.9.2.2 open()

```
bool GraphitiConnectionVCP::open ()  [virtual]
```

Implements GraphitiConnection.

4.9.2.3 read()

```
std::vector< unsigned char > GraphitiConnectionVCP::read (
            size_t size)  [virtual]
```

Implements GraphitiConnection.

4.9.2.4 write()

```
bool GraphitiConnectionVCP::write (
            const std::vector< unsigned char > & data)  [virtual]
```

Implements GraphitiConnection.

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

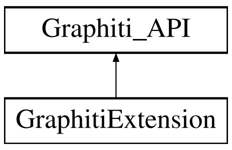
- lib/include/Graphiti/Connection/Connection_VCP.hpp
- lib/src/Connection/Connection_VCP.cpp

4.10 GraphitiExtension Class Reference

Class to improve the ease of use of the Graphiti Graphiti_API can be used with or without GraphitExtension GraphitiExtension can also be used as a reference to how the Graphiti_API can be used.

`#include <Extension.hpp>`

Inheritance diagram for GraphitiExtension:



Public Member Functions

- GraphitiExtension ()

  Construct a new Graphiti Extension object.
- GraphitiExtension (GraphitiConnection ∗connection)

  Construct a new Graphiti Extension object.
- ∼GraphitiExtension ()

  Destroy the Graphiti Extension:: Graphiti Extension object.
- bool startUpVCP (std::string port, bool keyEventsBool, bool touchEventsBool)

  Starts up VCP connection and turns on events.
- void shutDownVCP (bool keyEventsBool, bool touchEventsBool)

  Shuts down VCP connection and turns of events.
- void keyLoop (std::function< void(const std::set< std::string > &, void ∗)> key_func, bool ∗loop_condition, void ∗parameters=nullptr)

  Key Loop for concurrent key responses.

- void drawLoop (std::function< void(const Graphiti_API::DrawEvent &, void *)> draw_func, bool *loop_↩ condition, void *parameters=nullptr)

    Draw Loop for concurrent draw events.
- void setPin (int row, int col, int height)

    Set the Pin on screen.
- int index (int row, int col)

    Gets index of pin on Graphiti.
- void clearScreen ()

    Clears Screen display data in GraphitiExtension.
- void sleep (int time)

    Shorted sleep call Can copy it and make it locally so you can call sleep(2); rather than "graphitiExtension->sleep(2);" for simlpicity.

Public Member Functions inherited from Graphiti_API

- Graphiti_API ()

    Construct a new Graphiti_API object.
- Graphiti_API (GraphitiConnection *connection)

    Construct a new Graphiti_API object with connection.
- ~Graphiti_API ()

    Destroy the Graphiti_API object.
- void setConnection (GraphitiConnection *connection)

    Set Connection.
- void startResponseThread ()

    Start Response Thread.
- void stopResponseThread ()

    Stop Response Thread.
- std::optional< std::string > getNextOutputEvent ()

    Get the Next Output Event object.
- std::optional< std::vector< uint8_t > > getNextDisplayStatusEvent ()

    Get the Next Display Status Event object.
- std::optional< std::set< std::string > > getNextKeyEvent ()

    Get the Next Key Event object.
- std::optional< std::string > getNextGestureEvent ()

    Get the Next Gesture Event object.
- std::optional< DrawEvent > getNextDrawEvent ()

    Get the Next Draw Event object.
- void sendACK ()

    4.1 ACK request
- void sendNACK ()

    4.1 NACK request
- void getSoftwareVersion ()

    4.2.1 Get Software Version
- void getHardWareVersion ()

    4.2.2 Get Hardware Version
- void getSerialNumber ()

    4.2.3 Get Unit Serial Number
- void getBatteryStatus ()

    4.2.4 Get Battery Status
- void getResolutionInformation ()

4.2.5 Get Resolution Information
- void getDeviceOrientation ()

  4.2.6 Get Device Orientation
- void getHeightInformation ()

  4.2.7 Get Height Information
- void getDeviceName ()

  4.2.8 Get Device Name
- void updateDisplay (std::vector< uint8_t > screen)

  4.3 Display Access Commands
- void setDisplay ()

  4.3.2 Set Display
- void clearDisplay ()

  4.3.2 Clear Display
- void updateSinglePixel (uint8_t row, uint8_t column, uint8_t height, uint8_t blinkRate)

  4.3.3 Update Single pixel on Display
- void updateSingleRow (uint8_t row, std::vector< uint8_t > rowData)

  4.3.4 Update Single Row on Display
- void updateSingleColumn (uint8_t column, std::vector< uint8_t > columnData)

  4.3.5 Update Single Column on Display
- void getAllPixelsPositionStatus ()

  4.3.6 Get All Pixels' Position Status
- void getSinglePixelPositionStatus (uint8_t row, uint8_t column)

  4.3.7 Get Single Pixel Position Status
- void getSingleRowPixelPositionStatus (uint8_t row)

  4.3.8 Get Single Row Pixels Position Status
- void getSingleColumnPixelPositionStatus (uint8_t column)

  4.3.9 Get Single Column Pixels Position Status
- void showMessage (std::string message, std::vector< uint8_t > selectFlags, uint8_t blinkRate)

  4.3.10 Show Message on the Device
- void setCursor (uint8_t row, uint8_t column, uint8_t height, uint8_t length, uint8_t blinkRate)

  4.3.11 Set Cursor
- void sendImageInterruptible (std::string name, const std::string &filepath)

  4.3.12 Send Image (Interruptible)
- void sendImageBlocking (std::string name, const std::string &filepath)

  4.3.13 Send Image (Blocking)
- void setKeyEvent (bool enabled)

  4.4.1 Set Key Event
- void setTouchEvent (bool enabled)

  4.5.1 Set Touch Event
- void getLastTouchPointStatus ()

  4.5.2 Get Last Touch Point Status
- void vibratorControlCommand (uint8_t frequencyRange, uint8_t dutyCycle, uint16_t duration)

  4.6 Vibrator Control Command
- void getDateAndTime ()

  4.7.1 Get Date and Time
- void setDateAndTime (uint8_t day, uint8_t month, uint16_t year, uint8_t hour, uint8_t minute, uint8_t second)

  4.7.2 Set Date and Time

Public Attributes

- std::vector< uint8_t > screen

  Screen indicies of pin heights.

4.10.1 Detailed Description

Class to improve the ease of use of the Graphiti Graphiti_API can be used with or without GraphitExtension GraphitiExtension can also be used as a reference to how the Graphiti_API can be used.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 GraphitiExtension() [1/2]

```
GraphitiExtension::GraphitiExtension ()
```

Construct a new Graphiti Extension object.

4.10.2.2 GraphitiExtension() [2/2]

```
GraphitiExtension::GraphitiExtension (
            GraphitiConnection * connection)
```

Construct a new Graphiti Extension object.

Parameters

	connection	Connection to device

4.10.2.3 ∼GraphitiExtension()

```
GraphitiExtension::∼GraphitiExtension ()
```

Destroy the Graphiti Extension:: Graphiti Extension object.

4.10.3 Member Function Documentation

4.10.3.1 clearScreen()

```
void GraphitiExtension::clearScreen ()
```

Clears Screen display data in GraphitiExtension.

4.10.3.2 drawLoop()

```
void GraphitiExtension::drawLoop (
            std::function< void(const Graphiti_API::DrawEvent &, void *)> draw_func,
            bool * loop_condition,
            void * parameters = nullptr)
```

Draw Loop for concurrent draw events.

Parameters

draw_func	function using draw input
loop_condition	condition for looping
parameters	pointer to other variables used by function

4.10.3.3 index()

```
int GraphitiExtension::index (
            int row,
            int col)
```

Gets index of pin on Graphiti.

Parameters

row	row
col	column

Returns

int index on Graphiti screen

4.10.3.4 keyLoop()

```
void GraphitiExtension::keyLoop (
            std::function< void(const std::set< std::string > &, void *)> key_func,
            bool * loop_condition,
            void * parameters = nullptr)
```

Key Loop for concurrent key responses.

Can be used or be used as a reference

Note: Key cobinaitons that are reserved are: (5 + 6 + 8), (8 + Down), (7 + 8)

Parameters

key_func	function to use keys
loop_condition	condition for looping
parameters	pointer to other variables used by function

4.10.3.5 setPin()

```
void GraphitiExtension::setPin (
            int row,
            int col,
            int height)
```

Set the Pin on screen.

Parameters

row	row
col	column
height	pin heights 0-4

4.10.3.6 shutDownVCP()

```
void GraphitiExtension::shutDownVCP (
             bool keyEventsBool,
             bool touchEventsBool)
```

Shuts down VCP connection and turns of events.

Parameters

port	port name
keyEventsBool	boolean for key events

4.10.3.7 sleep()

```
void GraphitiExtension::sleep (
             int time)
```

Shorted sleep call Can copy it and make it locally so you can call sleep(2); rather than "graphitiExtension->sleep(2);" for simlpicity.

Parameters

time	time seconds

4.10.3.8 startUpVCP()

```
bool GraphitiExtension::startUpVCP (
             std::string port,
             bool keyEventsBool,
             bool touchEventsBool)
```

Starts up VCP connection and turns on events.

Parameters

port	port name
keyEventsBool	boolean for key events

Returns

true

false

4.10.4 Member Data Documentation

4.10.4.1 screen

```
std::vector<uint8_t> GraphitiExtension::screen
```

Screen indicies of pin heights.

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

- lib/include/Graphiti/Extension.hpp
- lib/src/Extension.cpp

4.11 GraphitiHandle Struct Reference

Public Attributes

- GraphitiExtension api

4.11.1 Member Data Documentation

4.11.1.1 api

```
GraphitiExtension GraphitiHandle::api
```

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

- lib/src/CWrapper/capi.cpp

4.12 KeyEvent_C Struct Reference

```
#include <capi.hpp>
```

Public Attributes

- char ** keys
- int count
- bool has_value

4.12.1 Member Data Documentation

4.12.1.1 count

```
int KeyEvent_C::count
```

4.12.1.2 has_value

```
bool KeyEvent_C::has_value
```

4.12.1.3 keys

```
char** KeyEvent_C::keys
```

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

- lib/include/Graphiti/CWrapper/capi.hpp

4.13 Pin Struct Reference

Holds pin information.

4.13.1 Detailed Description

Holds pin information.

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

- lib/include/Graphiti/API.hpp

4.14 Graphiti_API::PinInfo Struct Reference

```
#include <API.hpp>
```

Public Attributes

- int rowID

  rowID - row ID 1 to 40
- int columnID

  columnID - column ID 1 to 60
- int height

  height - pin height 0 to 4
- int blinkRate

  blinkRate - blink rate 0 to 50 (5 seconds)

4.14.1 Member Data Documentation

4.14.1.1 blinkRate

```
int Graphiti_API::PinInfo::blinkRate
```

blinkRate - blink rate 0 to 50 (5 seconds)

4.14.1.2 columnID

```
int Graphiti_API::PinInfo::columnID
```

columnID - column ID 1 to 60

4.14.1.3 height

```
int Graphiti_API::PinInfo::height
```

height - pin height 0 to 4

4.14.1.4 rowID

```
int Graphiti_API::PinInfo::rowID
```

rowID - row ID 1 to 40

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

- lib/include/Graphiti/API.hpp

4.15 PinInfo_C Struct Reference

```
#include <capi.hpp>
```

Public Attributes

- int rowID
- int columnID
- int height
- int blinkRate

4.15.1 Member Data Documentation

4.15.1.1 blinkRate

```
int PinInfo_C::blinkRate
```

4.15.1.2 columnID

```
int PinInfo_C::columnID
```

4.15.1.3 height

```
int PinInfo_C::height
```

4.15.1.4 rowID

```
int PinInfo_C::rowID
```

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

- lib/include/Graphiti/CWrapper/capi.hpp

4.16 ThreadSafeQueue< T > Class Template Reference

```
#include <ThreadSafeQueue.hpp>
```

Public Member Functions

- ThreadSafeQueue ()=default
- ~ThreadSafeQueue ()=default
- ThreadSafeQueue (const ThreadSafeQueue &)=delete
- ThreadSafeQueue & operator= (const ThreadSafeQueue &)=delete
- void push (const T &value)
- bool pop (T &value)
- bool empty () const
- size_t size () const

4.16.1 Constructor & Destructor Documentation

4.16.1.1 ThreadSafeQueue() [1/2]

```
template<typename T>
ThreadSafeQueue< T >::ThreadSafeQueue ()  [default]
```

4.16.1.2 ~ThreadSafeQueue()

```
template<typename T>
ThreadSafeQueue< T >::~ThreadSafeQueue ()  [default]
```

4.16.1.3 ThreadSafeQueue() [2/2]

```
template<typename T>
ThreadSafeQueue< T >::ThreadSafeQueue (
             const ThreadSafeQueue< T > & )  [delete]
```

4.16.2 Member Function Documentation

4.16.2.1 empty()

```
template<typename T>
bool ThreadSafeQueue< T >::empty () const  [inline]
```

4.16.2.2 operator=()

```
template<typename T>
ThreadSafeQueue & ThreadSafeQueue< T >::operator= (
             const ThreadSafeQueue< T > & )  [delete]
```

4.16.2.3 pop()

```
template<typename T>
bool ThreadSafeQueue< T >::pop (
             T & value)  [inline]
```

4.16.2.4 push()

```
template<typename T>
void ThreadSafeQueue< T >::push (
             const T & value)  [inline]
```

4.16.2.5 size()

```
template<typename T>
size_t ThreadSafeQueue< T >::size () const  [inline]
```

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

- lib/include/Graphiti/ThreadSafeQueue.hpp

Chapter 5

File Documentation

5.1 lib/build/CMakeFiles/4.0.2/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_STD_99 199901L
- #define C_STD_11 201112L
- #define C_STD_17 201710L
- #define C_STD_23 202311L
- #define C_VERSION

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

5.1.1 Macro Definition Documentation

5.1.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

5.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

5.1.1.3 C_STD_11

```
#define C_STD_11 201112L
```

5.1.1.4 C_STD_17

```
#define C_STD_17 201710L
```

5.1.1.5 C_STD_23

```
#define C_STD_23 202311L
```

5.1.1.6 C_STD_99

```
#define C_STD_99 199901L
```

5.1.1.7 C_VERSION

```
#define C_VERSION
```

5.1.1.8 COMPILER_ID

```
#define COMPILER_ID ""
```

5.1.1.9 DEC

```
#define DEC(
                n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

5.1.1.10 HEX

```
#define HEX(
                n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

5.1.1.11 PLATFORM_ID

```
#define PLATFORM_ID
```

5.1.1.12 STRINGIFY

```
#define STRINGIFY(
                X)
```

Value:

```
STRINGIFY_HELPER(X)
```

5.1.1.13 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
                X)
```

Value:

```
#X
```

5.1.2 Function Documentation

5.1.2.1 main()

```
int main (
              int argc,
              char * argv[])
```

5.1.3 Variable Documentation

5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

5.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["




  "OFF"

"]"
```

5.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

5.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

5.2 lib/build/CMakeFiles/4.0.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD_98 199711L
- #define CXX_STD_11 201103L
- #define CXX_STD_14 201402L
- #define CXX_STD_17 201703L
- #define CXX_STD_20 202002L
- #define CXX_STD_23 202302L
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

5.2.1 Macro Definition Documentation

5.2.1.1 __has_include

```
#define __has_include(
              x)
```

Value:

```
0
```

5.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

5.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

5.2.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

5.2.1.5 CXX_STD_11

```
#define CXX_STD_11 201103L
```

5.2.1.6 CXX_STD_14

```
#define CXX_STD_14 201402L
```

5.2.1.7 CXX_STD_17

```
#define CXX_STD_17 201703L
```

5.2.1.8 CXX_STD_20

```
#define CXX_STD_20 202002L
```

5.2.1.9 CXX_STD_23

```
#define CXX_STD_23 202302L
```

5.2.1.10 CXX_STD_98

```
#define CXX_STD_98 199711L
```

5.2.1.11 DEC

```
#define DEC(
              n)
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

5.2.1.12 HEX

```
#define HEX(
              n)
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

5.2.1.13 PLATFORM_ID

```
#define PLATFORM_ID
```

5.2.1.14 STRINGIFY

```
#define STRINGIFY(
              X)
```

Value:

```
STRINGIFY_HELPER(X)
```

5.2.1.15 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

Value:

```
#X
```

5.2.2 Function Documentation

5.2.2.1 main()

```
int main (
             int argc,
             char * argv[])
```

5.2.3 Variable Documentation

5.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

5.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

5.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["




  "OFF"

"]"
```

5.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["







  "98"

"]"
```

5.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

5.3 lib/include/Graphiti/API.hpp File Reference

```
#include <Graphiti/Connection/Connection.hpp>
#include <thread>
#include <atomic>
#include <string>
#include <vector>
#include <cstdint>
#include <iostream>
#include <functional>
#include <iomanip>
#include <optional>
#include <map>
#include <set>
#include <sstream>
#include <fstream>
#include "ThreadSafeQueue.hpp"
```

Classes

- class Graphiti_API
- struct Graphiti_API::PinInfo
- struct Graphiti_API::DrawEvent

Macros

- #define MAX_ROW_DATA 60
- #define MAX_COLUMN_DATA 40
- #define ROW_COUNT 40
- #define COLUMN_COUNT 60

5.3.1 Macro Definition Documentation

5.3.1.1 COLUMN_COUNT

```
#define COLUMN_COUNT 60
```

5.3.1.2 MAX_COLUMN_DATA

```
#define MAX_COLUMN_DATA 40
```

5.3.1.3 MAX_ROW_DATA

```
#define MAX_ROW_DATA 60
```

5.3.1.4 ROW_COUNT

```
#define ROW_COUNT 40
```

5.4 API.hpp

Go to the documentation of this file.

```
#pragma once
#include <Graphiti/Connection/Connection.hpp>
#include <thread>
#include <atomic>
#include <string>
#include <vector>
#include <cstdint>
#include <iostream>
#include <functional>
#include <iomanip>
#include <optional>
#include <map>
#include <set>
#include <sstream>
#include <fstream>

#include "ThreadSafeQueue.hpp"

#define MAX_ROW_DATA 60
#define MAX_COLUMN_DATA 40

#define ROW_COUNT 40
#define COLUMN_COUNT 60

class Graphiti_API {
public:
    Graphiti_API();

    Graphiti_API(GraphitiConnection* connection);

    ~Graphiti_API();

    void setConnection(GraphitiConnection* connection);

    void startResponseThread();

    void stopResponseThread();

    std::optional<std::string> getNextOutputEvent();

    std::optional<std::vector<uint8_t» getNextDisplayStatusEvent();

    std::optional<std::set<std::string» getNextKeyEvent();

    std::optional<std::string> getNextGestureEvent();

    struct PinInfo {
        int rowID;

        int columnID;

        int height;

        int blinkRate;
    };

    struct DrawEvent {
        int length;

        std::vector<PinInfo> pins;
    };

    std::optional<DrawEvent> getNextDrawEvent();

    void sendACK();

    void sendNACK();

    void getSoftwareVersion();

    void getHardWareVersion();

    void getSerialNumber();

    void getBatteryStatus();

    void getResolutionInformation();

    void getDeviceOrientation();

    void getHeightInformation();

    void getDeviceName();


    void updateDisplay(std::vector<uint8_t> screen);

    void setDisplay();

    void clearDisplay();

    void updateSinglePixel(uint8_t row, uint8_t column, uint8_t height, uint8_t blinkRate);

    void updateSingleRow(uint8_t row, std::vector<uint8_t> rowData);

    void updateSingleColumn(uint8_t column, std::vector<uint8_t> columnData);

    void getAllPixelsPositionStatus();

    void getSinglePixelPositionStatus(uint8_t row, uint8_t column);

    void getSingleRowPixelPositionStatus(uint8_t row);

    void getSingleColumnPixelPositionStatus(uint8_t column);

    void showMessage(std::string message, std::vector<uint8_t> selectFlags, uint8_t blinkRate);

    void setCursor(uint8_t row, uint8_t column, uint8_t height, uint8_t length, uint8_t blinkRate);

    void sendImageInterruptible(std::string name, const std::string& filepath);

    void sendImageBlocking(std::string name, const std::string& filepath);

    /*4.4 User Keys Access Commands*/

    void setKeyEvent(bool enabled);

    void setTouchEvent(bool enabled);

    void getLastTouchPointStatus();

    void vibratorControlCommand(uint8_t frequencyRange, uint8_t dutyCycle, uint16_t duration);

    void getDateAndTime();

    void setDateAndTime(uint8_t day, uint8_t month, uint16_t year, uint8_t hour, uint8_t minute,
uint8_t second);

private:

    GraphitiConnection* conn_;

    std::thread responseThread;

    std::atomic<bool> running{ false };

    void runResponseLoop();

    void getResponse();

    void initMaps();

    std::unordered_map<uint8_t, std::function<void()» responseSelectFunc;

    std::map<std::pair<int, int>, std::string> keyMap;

    struct ResponseInfo {
        std::string funtion_name;
        std::string event;
        int length_bytes;
    };

    std::unordered_map<uint8_t, ResponseInfo> responseInfoMap;

    std::unordered_map<uint8_t, std::string> commonResponseMap;

    std::unordered_map<uint8_t, std::string> gestureMap;

    ThreadSafeQueue<std::set<std::string» keyEventQueue;

    ThreadSafeQueue<std::string> outputEventQueue;

    ThreadSafeQueue<std::string> gestureEventQueue;

    ThreadSafeQueue<DrawEvent> drawEventQueue;

    ThreadSafeQueue<std::vector<uint8_t» displayStatusQueue;

    bool readBytes(std::vector<uint8_t> &data, int length);

    void readStatus(uint8_t startByte, std::string returnName, int length);

    void setUpResponseMap();

    void setUpKeyMap();

    void setUpGestureMap();

    void setUpCommonResponseMap();

    void readKeyEventStatus();

    void readKeyEvent();

    void readTouchEventStatus();

    void readDrawEvent();

    void readGestureEvent();

    void readCommonResponse();

    std::optional<std::string> bytesToString(std::vector<uint8_t> &data, int length);

    void readCommand(uint8_t startByte, std::string name,
     std::string event, int lengthBytes);

    bool checkEndByte(std::vector<uint8_t> data, uint8_t startByte);

    std::vector<uint8_t> calcChecksum(const std::vector<uint8_t>& data);

    void Graphiti_IO_Write(std::vector<uint8_t> bytes);

    std::vector<uint8_t> startOfFrameCheck(std::vector<uint8_t> bytes, uint8_t startByte);

    void sendImage(uint8_t commandByte, std::string name, const std::string& filepath);

    void printKeyEvent(std::set<std::string> keyEvent);

    void debugByte(uint8_t byte);

    void printVectorHex(const std::vector<uint8_t>& data);
};
```

5.5 lib/include/Graphiti/API_HID.hpp File Reference

```
#include <vector>
#include <string>
```

Classes

- class Graphiti_API_HID

5.6 API_HID.hpp

Go to the documentation of this file.

```
#pragma once
//#include <hidapi/hidapi.hpp>
#include <vector>
#include <string>

class Graphiti_API_HID {
public:
    Graphiti_API_HID(unsigned short vid, unsigned short pid);
    ~Graphiti_API_HID();

    bool open();
    void close();
    bool write(const std::vector<unsigned char>& data);
    std::vector<unsigned char> read(size_t size);

private:
//    hid_device* device_;
    unsigned short vid_;
    unsigned short pid_;
};
```

5.7 lib/include/Graphiti/API_VCP.hpp File Reference

```
#include <asio.hpp>
#include <vector>
#include <string>
```

Classes

- class Graphiti_API_VCP

Macros

- #define ASIO_STANDALONE

5.7.1 Macro Definition Documentation

5.7.1.1 ASIO_STANDALONE

```
#define ASIO_STANDALONE
```

5.8 API_VCP.hpp

Go to the documentation of this file.

```
#pragma once

#define ASIO_STANDALONE
#include <asio.hpp>
using namespace asio;
#include <vector>
#include <string>

class Graphiti_API_VCP {
public:
    explicit Graphiti_API_VCP(const std::string& port);
    ~Graphiti_API_VCP();

    bool open();
    void close();
    bool write(const std::vector<unsigned char>& data);
    std::vector<unsigned char> read(size_t size);

private:
    std::string port_;
    asio::io_context io_;
    asio::serial_port serial_;
};
```

5.9 lib/include/Graphiti/Connection/Connection.hpp File Reference

```
#include <vector>
```

Classes

- class GraphitiConnection

5.10 Connection.hpp

Go to the documentation of this file.

```
#pragma once
#include <vector>

class GraphitiConnection {
public:
    virtual ~GraphitiConnection() {}
    virtual bool open() = 0;
    virtual void close() = 0;
    virtual bool write(const std::vector<unsigned char>& data) = 0;
    virtual std::vector<unsigned char> read(size_t size) = 0;
};

```

5.11 lib/include/Graphiti/Connection/Connection_HID.hpp File Reference

5.12 Connection_HID.hpp

Go to the documentation of this file.

```

```

5.13 lib/include/Graphiti/Connection/Connection_VCP.hpp File Reference

```
#include <string>
#include <vector>
#include <asio.hpp>
#include "Connection.hpp"
```

Classes

- class GraphitiConnectionVCP

Macros

- #define ASIO_STANDALONE

5.13.1 Macro Definition Documentation

5.13.1.1 ASIO_STANDALONE

```
#define ASIO_STANDALONE
```

5.14 Connection_VCP.hpp

Go to the documentation of this file.

```
#ifndef GRAPHITI_CONNECTION_VCP_H
#define GRAPHITI_CONNECTION_VCP_H

#include <string>
#include <vector>
#define ASIO_STANDALONE
#include <asio.hpp>
using namespace asio;
#include "Connection.hpp"

class GraphitiConnectionVCP : public GraphitiConnection {
public:
    explicit GraphitiConnectionVCP(const std::string& port);

    bool open();
    void close();
    bool write(const std::vector<unsigned char>& data);
    std::vector<unsigned char> read(size_t size);

private:
    asio::io_context io_;
    asio::serial_port serial_;
    std::string port_;
};

#endif // GRAPHITI_CONNECTION_VCP_H
```

5.15 lib/include/Graphiti/CWrapper/capi.hpp File Reference

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

Classes

- struct DisplayStatusEvent_C
- struct KeyEvent_C
- struct PinInfo_C
- struct DrawEvent_C

Typedefs

- typedef struct GraphitiHandle GraphitiHandle

Functions

- GraphitiHandle ∗ graphiti_create ()
- GraphitiHandle ∗ graphiti_createWithConnection (void ∗connection)
- void graphiti_destroy (GraphitiHandle ∗handle)
- void graphiti_setConnection (GraphitiHandle ∗handle, void ∗connection)
- bool startUpVCP (GraphitiHandle ∗handle, const char ∗portName, bool keyEventsBool, bool touchEvents←↩ Bool)
- void shutDownVCP (GraphitiHandle ∗handle, bool keyEventsBool, bool touchEventsBool)
- int graphiti_index (GraphitiHandle ∗handle, int row, int col)
- void setPin (GraphitiHandle ∗handle, int row, int col, int height)
- void clearScreen (GraphitiHandle ∗handle)
- void sleep (GraphitiHandle ∗handle, int time)
- void graphiti_startResponseThread (GraphitiHandle ∗handle)
- void graphiti_stopResponseThread (GraphitiHandle ∗handle)
- char ∗ graphiti_getNextOutputEvent (GraphitiHandle ∗handle)
- void graphiti_freeString (char ∗str)
- DisplayStatusEvent_C graphiti_getNextDisplayStatusEvent (GraphitiHandle ∗handle)
- void graphiti_freeDisplayStatusEvent (DisplayStatusEvent_C ∗event)
- KeyEvent_C graphiti_getNextKeyEvent (GraphitiHandle ∗handle)
- void graphiti_freeKeyEvent (KeyEvent_C ∗event)
- char ∗ graphiti_getNextGestureEvent (GraphitiHandle ∗handle)
- DrawEvent_C graphiti_getNextDrawEvent (GraphitiHandle ∗handle)
- void graphiti_freeDrawEvent (DrawEvent_C ∗event)
- void graphiti_sendACK (GraphitiHandle ∗handle)
- void graphiti_sendNACK (GraphitiHandle ∗handle)
- void graphiti_getSoftwareVersion (GraphitiHandle ∗handle)
- void graphiti_getHardWareVersion (GraphitiHandle ∗handle)
- void graphiti_getSerialNumber (GraphitiHandle ∗handle)
- void graphiti_getBatteryStatus (GraphitiHandle ∗handle)
- void graphiti_getResolutionInformation (GraphitiHandle ∗handle)
- void graphiti_getDeviceOrientation (GraphitiHandle ∗handle)
- void graphiti_getHeightInformation (GraphitiHandle ∗handle)
- void graphiti_getDeviceName (GraphitiHandle ∗handle)
- void graphiti_updateDisplay (GraphitiHandle ∗handle, const uint8_t ∗screen_data, int length)
- void graphiti_setDisplay (GraphitiHandle ∗handle)
- void graphiti_clearDisplay (GraphitiHandle ∗handle)
- void graphiti_updateSinglePixel (GraphitiHandle ∗handle, int row, int column, int height, int blinkRate)
- void graphiti_updateSingleRow (GraphitiHandle ∗handle, int row, const uint8_t ∗rowData, int length)
- void graphiti_updateSingleColumn (GraphitiHandle ∗handle, int column, const uint8_t ∗columnData, int length)
- void graphiti_getAllPixelsPositionStatus (GraphitiHandle ∗handle)

- void graphiti_getSinglePixelPositionStatus (GraphitiHandle ∗handle, int row, int column)
- void graphiti_getSingleRowPixelPositionStatus (GraphitiHandle ∗handle, uint8_t row)
- void graphiti_getSingleColumnPixelPositionStatus (GraphitiHandle ∗handle, uint8_t column)
- void graphiti_showMessage (GraphitiHandle ∗handle, const char ∗message, const uint8_t ∗selectFlags, int flagLength, int blinkRate)
- void graphiti_setCursor (GraphitiHandle ∗handle, int row, int column, int height, int length, int blinkRate)
- void graphiti_sendImageInterruptible (GraphitiHandle ∗handle, const char ∗name, const char ∗filepath)
- void graphiti_sendImageBlocking (GraphitiHandle ∗handle, const char ∗name, const char ∗filepath)
- void graphiti_setKeyEvent (GraphitiHandle ∗handle, bool enabled)
- void graphiti_setTouchEvent (GraphitiHandle ∗handle, bool enabled)
- void graphiti_getLastTouchPointStatus (GraphitiHandle ∗handle)
- void graphiti_vibratorControlCommand (GraphitiHandle ∗handle, int frequencyRange, int dutyCycle, int duration)
- void graphiti_getDateAndTime (GraphitiHandle ∗handle)
- void graphiti_setDateAndTime (GraphitiHandle ∗handle, int day, int month, int year, int hour, int minute, int second)

5.15.1 Typedef Documentation

5.15.1.1 GraphitiHandle

```
typedef struct GraphitiHandle GraphitiHandle
```

5.15.2 Function Documentation

5.15.2.1 clearScreen()

```
void clearScreen (
             GraphitiHandle * handle)
```

5.15.2.2 graphiti_clearDisplay()

```
void graphiti_clearDisplay (
             GraphitiHandle * handle)
```

5.15.2.3 graphiti_create()

```
GraphitiHandle * graphiti_create ()
```

5.15.2.4 graphiti_createWithConnection()

```
GraphitiHandle * graphiti_createWithConnection (
             void * connection)
```

5.15.2.5 graphiti_destroy()

```
void graphiti_destroy (
            GraphitiHandle * handle)
```

5.15.2.6 graphiti_freeDisplayStatusEvent()

```
void graphiti_freeDisplayStatusEvent (
            DisplayStatusEvent_C * event)
```

5.15.2.7 graphiti_freeDrawEvent()

```
void graphiti_freeDrawEvent (
            DrawEvent_C * event)
```

5.15.2.8 graphiti_freeKeyEvent()

```
void graphiti_freeKeyEvent (
            KeyEvent_C * event)
```

5.15.2.9 graphiti_freeString()

```
void graphiti_freeString (
            char * str)
```

5.15.2.10 graphiti_getAllPixelsPositionStatus()

```
void graphiti_getAllPixelsPositionStatus (
            GraphitiHandle * handle)
```

5.15.2.11 graphiti_getBatteryStatus()

```
void graphiti_getBatteryStatus (
            GraphitiHandle * handle)
```

5.15.2.12 graphiti_getDateAndTime()

```
void graphiti_getDateAndTime (
            GraphitiHandle * handle)
```

5.15.2.13 graphiti_getDeviceName()

```
void graphiti_getDeviceName (
            GraphitiHandle * handle)
```

5.15.2.14 graphiti_getDeviceOrientation()

```
void graphiti_getDeviceOrientation (
             GraphitiHandle * handle)
```

5.15.2.15 graphiti_getHardWareVersion()

```
void graphiti_getHardWareVersion (
             GraphitiHandle * handle)
```

5.15.2.16 graphiti_getHeightInformation()

```
void graphiti_getHeightInformation (
             GraphitiHandle * handle)
```

5.15.2.17 graphiti_getLastTouchPointStatus()

```
void graphiti_getLastTouchPointStatus (
             GraphitiHandle * handle)
```

5.15.2.18 graphiti_getNextDisplayStatusEvent()

```
DisplayStatusEvent_C graphiti_getNextDisplayStatusEvent (
             GraphitiHandle * handle)
```

5.15.2.19 graphiti_getNextDrawEvent()

```
DrawEvent_C graphiti_getNextDrawEvent (
             GraphitiHandle * handle)
```

5.15.2.20 graphiti_getNextGestureEvent()

```
char * graphiti_getNextGestureEvent (
             GraphitiHandle * handle)
```

5.15.2.21 graphiti_getNextKeyEvent()

```
KeyEvent_C graphiti_getNextKeyEvent (
             GraphitiHandle * handle)
```

5.15.2.22 graphiti_getNextOutputEvent()

```
char * graphiti_getNextOutputEvent (
             GraphitiHandle * handle)
```

5.15.2.23 graphiti_getResolutionInformation()

```
void graphiti_getResolutionInformation (
            GraphitiHandle * handle)
```

5.15.2.24 graphiti_getSerialNumber()

```
void graphiti_getSerialNumber (
            GraphitiHandle * handle)
```

5.15.2.25 graphiti_getSingleColumnPixelPositionStatus()

```
void graphiti_getSingleColumnPixelPositionStatus (
            GraphitiHandle * handle,
            uint8_t column)
```

5.15.2.26 graphiti_getSinglePixelPositionStatus()

```
void graphiti_getSinglePixelPositionStatus (
            GraphitiHandle * handle,
            int row,
            int column)
```

5.15.2.27 graphiti_getSingleRowPixelPositionStatus()

```
void graphiti_getSingleRowPixelPositionStatus (
            GraphitiHandle * handle,
            uint8_t row)
```

5.15.2.28 graphiti_getSoftwareVersion()

```
void graphiti_getSoftwareVersion (
            GraphitiHandle * handle)
```

5.15.2.29 graphiti_index()

```
int graphiti_index (
            GraphitiHandle * handle,
            int row,
            int col)
```

5.15.2.30 graphiti_sendACK()

```
void graphiti_sendACK (
            GraphitiHandle * handle)
```

5.15.2.31 graphiti_sendImageBlocking()

```
void graphiti_sendImageBlocking (
            GraphitiHandle * handle,
            const char * name,
            const char * filepath)
```

5.15.2.32 graphiti_sendImageInterruptible()

```
void graphiti_sendImageInterruptible (
            GraphitiHandle * handle,
            const char * name,
            const char * filepath)
```

5.15.2.33 graphiti_sendNACK()

```
void graphiti_sendNACK (
            GraphitiHandle * handle)
```

5.15.2.34 graphiti_setConnection()

```
void graphiti_setConnection (
            GraphitiHandle * handle,
            void * connection)
```

5.15.2.35 graphiti_setCursor()

```
void graphiti_setCursor (
            GraphitiHandle * handle,
            int row,
            int column,
            int height,
            int length,
            int blinkRate)
```

5.15.2.36 graphiti_setDateAndTime()

```
void graphiti_setDateAndTime (
            GraphitiHandle * handle,
            int day,
            int month,
            int year,
            int hour,
            int minute,
            int second)
```

5.15.2.37 graphiti_setDisplay()

```
void graphiti_setDisplay (
             GraphitiHandle * handle)
```

5.15.2.38 graphiti_setKeyEvent()

```
void graphiti_setKeyEvent (
             GraphitiHandle * handle,
             bool enabled)
```

5.15.2.39 graphiti_setTouchEvent()

```
void graphiti_setTouchEvent (
             GraphitiHandle * handle,
             bool enabled)
```

5.15.2.40 graphiti_showMessage()

```
void graphiti_showMessage (
             GraphitiHandle * handle,
             const char * message,
             const uint8_t * selectFlags,
             int flagLength,
             int blinkRate)
```

5.15.2.41 graphiti_startResponseThread()

```
void graphiti_startResponseThread (
             GraphitiHandle * handle)
```

5.15.2.42 graphiti_stopResponseThread()

```
void graphiti_stopResponseThread (
             GraphitiHandle * handle)
```

5.15.2.43 graphiti_updateDisplay()

```
void graphiti_updateDisplay (
             GraphitiHandle * handle,
             const uint8_t * screen_data,
             int length)
```

5.15.2.44 graphiti_updateSingleColumn()

```
void graphiti_updateSingleColumn (
            GraphitiHandle * handle,
            int column,
            const uint8_t * columnData,
            int length)
```

5.15.2.45 graphiti_updateSinglePixel()

```
void graphiti_updateSinglePixel (
            GraphitiHandle * handle,
            int row,
            int column,
            int height,
            int blinkRate)
```

5.15.2.46 graphiti_updateSingleRow()

```
void graphiti_updateSingleRow (
            GraphitiHandle * handle,
            int row,
            const uint8_t * rowData,
            int length)
```

5.15.2.47 graphiti_vibratorControlCommand()

```
void graphiti_vibratorControlCommand (
            GraphitiHandle * handle,
            int frequencyRange,
            int dutyCycle,
            int duration)
```

5.15.2.48 setPin()

```
void setPin (
            GraphitiHandle * handle,
            int row,
            int col,
            int height)
```

5.15.2.49 shutDownVCP()

```
void shutDownVCP (
            GraphitiHandle * handle,
            bool keyEventsBool,
            bool touchEventsBool)
```

5.15.2.50 sleep()

```
void sleep (
            GraphitiHandle * handle,
            int time)
```

5.15.2.51 startUpVCP()

```
bool startUpVCP (
            GraphitiHandle * handle,
            const char * portName,
            bool keyEventsBool,
            bool touchEventsBool)
```

5.16 capi.hpp

Go to the documentation of this file.

```
#pragma once

#include <stdint.h>

#ifdef __cplusplus
extern "C" {
#endif

typedef struct GraphitiHandle GraphitiHandle;

GraphitiHandle* graphiti_create();
GraphitiHandle* graphiti_createWithConnection(void* connection);
void graphiti_destroy(GraphitiHandle* handle);

void graphiti_setConnection(GraphitiHandle* handle, void* connection);

bool startUpVCP(GraphitiHandle* handle, const char* portName, bool keyEventsBool, bool
      touchEventsBool);

void shutDownVCP(GraphitiHandle* handle, bool keyEventsBool, bool touchEventsBool);

int graphiti_index(GraphitiHandle* handle, int row, int col);

void setPin(GraphitiHandle* handle, int row, int col, int height);

void clearScreen(GraphitiHandle* handle);

void sleep(GraphitiHandle* handle, int time);

void graphiti_startResponseThread(GraphitiHandle* handle);
void graphiti_stopResponseThread(GraphitiHandle* handle);

char* graphiti_getNextOutputEvent(GraphitiHandle* handle);
void graphiti_freeString(char* str);

typedef struct {
    uint8_t* data;     // Pointer to event data
    int length;        // Length of data
    bool has_value;    // Whether optional has value
} DisplayStatusEvent_C;

DisplayStatusEvent_C graphiti_getNextDisplayStatusEvent(GraphitiHandle* handle);
void graphiti_freeDisplayStatusEvent(DisplayStatusEvent_C* event);

// ---- Key Event ----
typedef struct {
    char** keys;       // Array of key strings
    int count;         // Number of keys
    bool has_value;    // Whether optional has value
} KeyEvent_C;

KeyEvent_C graphiti_getNextKeyEvent(GraphitiHandle* handle);
void graphiti_freeKeyEvent(KeyEvent_C* event);

char* graphiti_getNextGestureEvent(GraphitiHandle* handle);

// ---- Draw Event ----
typedef struct {
    int rowID;
    int columnID;
    int height;
    int blinkRate;
} PinInfo_C;

typedef struct {
    PinInfo_C* pins;  // Array of pins
    int length;       // Number of pins
    bool has_value;   // Whether optional has value
} DrawEvent_C;

DrawEvent_C graphiti_getNextDrawEvent(GraphitiHandle* handle);
void graphiti_freeDrawEvent(DrawEvent_C* event);

//ACK & NACK
void graphiti_sendACK(GraphitiHandle* handle);
void graphiti_sendNACK(GraphitiHandle* handle);

void graphiti_getSoftwareVersion(GraphitiHandle* handle);
void graphiti_getHardWareVersion(GraphitiHandle* handle);
void graphiti_getSerialNumber(GraphitiHandle* handle);
void graphiti_getBatteryStatus(GraphitiHandle* handle);
void graphiti_getResolutionInformation(GraphitiHandle* handle);
void graphiti_getDeviceOrientation(GraphitiHandle* handle);
void graphiti_getHeightInformation(GraphitiHandle* handle);
void graphiti_getDeviceName(GraphitiHandle* handle);

void graphiti_updateDisplay(GraphitiHandle* handle, const uint8_t* screen_data, int length);
void graphiti_setDisplay(GraphitiHandle* handle);
void graphiti_clearDisplay(GraphitiHandle* handle);
void graphiti_updateSinglePixel(GraphitiHandle* handle, int row, int column, int height, int
blinkRate);
void graphiti_updateSingleRow(GraphitiHandle* handle, int row, const uint8_t* rowData, int length);
void graphiti_updateSingleColumn(GraphitiHandle* handle, int column, const uint8_t* columnData, int
length);

void graphiti_getAllPixelsPositionStatus(GraphitiHandle* handle);
void graphiti_getSinglePixelPositionStatus(GraphitiHandle* handle, int row, int column);
void graphiti_getSingleRowPixelPositionStatus(GraphitiHandle* handle, uint8_t row);
void graphiti_getSingleColumnPixelPositionStatus(GraphitiHandle* handle, uint8_t column);

void graphiti_showMessage(GraphitiHandle* handle, const char* message, const uint8_t* selectFlags, int
flagLength, int blinkRate);
void graphiti_setCursor(GraphitiHandle* handle, int row, int column, int height, int length, int
blinkRate);
void graphiti_sendImageInterruptible(GraphitiHandle* handle, const char* name, const char* filepath);
void graphiti_sendImageBlocking(GraphitiHandle* handle, const char* name, const char* filepath);

void graphiti_setKeyEvent(GraphitiHandle* handle, bool enabled);
void graphiti_setTouchEvent(GraphitiHandle* handle, bool enabled);
void graphiti_getLastTouchPointStatus(GraphitiHandle* handle);
void graphiti_vibratorControlCommand(GraphitiHandle* handle, int frequencyRange, int dutyCycle, int
duration);
void graphiti_getDateAndTime(GraphitiHandle* handle);
void graphiti_setDateAndTime(GraphitiHandle* handle, int day, int month, int year, int hour, int
minute, int second);

#ifdef __cplusplus
}
#endif
```

5.17 lib/include/Graphiti/Extension.hpp File Reference

```
#include <cstdio>
#include <iostream>
#include <chrono>
#include "API.hpp"
#include <Graphiti/Connection/Connection_VCP.hpp>
```

Classes

- class GraphitiExtension

Class to improve the ease of use of the Graphiti Graphiti_API can be used with or without GraphitExtension GraphitiExtension can also be used as a reference to how the Graphiti_API can be used.

5.18 Extension.hpp

Go to the documentation of this file.

```
#pragma once

#include <cstdio>
#include <iostream>
#include <chrono>

#include "API.hpp"

#include <Graphiti/Connection/Connection_VCP.hpp>

class GraphitiExtension : public Graphiti_API {
public:
    GraphitiExtension();

    GraphitiExtension(GraphitiConnection* connection);


    ~GraphitiExtension();

    bool startUpVCP(std::string port, bool keyEventsBool, bool touchEventsBool);

    void shutDownVCP(bool keyEventsBool, bool touchEventsBool);

    std::vector<uint8_t> screen;

    void keyLoop(
        std::function<void(const std::set<std::string>&, void*)> key_func,
        bool *loop_condition,
        void* parameters = nullptr
    );

    void drawLoop(
        std::function<void(const Graphiti_API::DrawEvent&, void*)> draw_func,
        bool *loop_condition,
        void* parameters = nullptr
    );

    void setPin(int row, int col, int height);

    int index(int row, int col);

    void clearScreen();

    void sleep(int time);

private:

    GraphitiConnectionVCP* vcpConn = nullptr;
};
```

5.19 lib/include/Graphiti/ThreadSafeQueue.hpp File Reference

```
#include <queue>
#include <mutex>
```

Classes

- class ThreadSafeQueue< T >

5.20 ThreadSafeQueue.hpp

Go to the documentation of this file.

```
#ifndef THREAD_SAFE_QUEUE_H
#define THREAD_SAFE_QUEUE_H

#include <queue>
#include <mutex>

//Library for ThreadSafeQueue's in C++

template<typename T>
class ThreadSafeQueue {
private:
    std::queue<T> queue_;
    std::mutex mutex_;

public:
    ThreadSafeQueue() = default;
    ~ThreadSafeQueue() = default;

    // Disable copy and assignment
    ThreadSafeQueue(const ThreadSafeQueue&) = delete;
    ThreadSafeQueue& operator=(const ThreadSafeQueue&) = delete;

    void push(const T& value) {
        std::lock_guard<std::mutex> lock(mutex_);
        queue_.push(value);
    }

    bool pop(T& value) {
        std::lock_guard<std::mutex> lock(mutex_);
        if (queue_.empty()) return false;
        value = queue_.front();
        queue_.pop();
        return true;
    }

    bool empty() const {
        std::lock_guard<std::mutex> lock(mutex_);
        return queue_.empty();
    }

    size_t size() const {
        std::lock_guard<std::mutex> lock(mutex_);
        return queue_.size();
    }
};

#endif // THREAD_SAFE_QUEUE_H
```

5.21 lib/src/API.cpp File Reference

```
#include <cstdio>
#include <Graphiti/API.hpp>
```

5.22 lib/src/API_HID.cpp File Reference

```
#include <Graphiti/API_HID.hpp>
#include <iostream>
```

5.23 lib/src/API_VCP.cpp File Reference

```
#include <Graphiti/API_VCP.hpp>
#include <iostream>
```

5.24 lib/src/Connection/Connection_HID.cpp File Reference

5.25 lib/src/Connection/Connection_VCP.cpp File Reference

```
#include <Graphiti/Connection/Connection_VCP.hpp>
#include <iostream>
```

5.26 lib/src/CWrapper/capi.cpp File Reference

```
#include <vector>
#include <set>
#include <string>
#include <cstring>
#include <Graphiti/CWrapper/capi.hpp>
#include <Graphiti/Extension.hpp>
```

Classes

- struct GraphitiHandle

Functions

- GraphitiHandle ∗ graphiti_create ()
- GraphitiHandle ∗ graphiti_create_with_connection (void ∗connection)
- void graphiti_destroy (GraphitiHandle ∗handle)
- void graphiti_setConnection (GraphitiHandle ∗handle, void ∗connection)
- void graphiti_startUpVCP (GraphitiHandle ∗handle, const char ∗portName, bool keyEventsBool, bool touch←EventsBool)
- void graphiti_shutDownVCP (GraphitiHandle ∗handle, const char ∗portName, bool keyEventsBool, bool touchEventsBool)
- int graphiti_index (GraphitiHandle ∗handle, int row, int col)
- void graphiti_setPin (GraphitiHandle ∗handle, int row, int col, int height)
- void graphiti_clearScreen (GraphitiHandle ∗handle)
- void graphiti_sleep (GraphitiHandle ∗handle, int time)
- void graphiti_startResponseThread (GraphitiHandle ∗handle)
- void graphiti_stopResponseThread (GraphitiHandle ∗handle)
- char ∗ graphiti_getNextOutputEvent (GraphitiHandle ∗handle)
- void graphiti_freeString (char ∗str)
- DisplayStatusEvent_C graphiti_getNextDisplayStatusEvent (GraphitiHandle ∗handle)
- void graphiti_freeDisplayStatusEvent (DisplayStatusEvent_C ∗event)
- KeyEvent_C graphiti_getNextKeyEvent (GraphitiHandle ∗handle)
- void graphiti_freeKeyEvent (KeyEvent_C ∗event)
- char ∗ graphiti_getNextGestureEvent (GraphitiHandle ∗handle)
- DrawEvent_C graphiti_getNextDrawEvent (GraphitiHandle ∗handle)
- void graphiti_freeDrawEvent (DrawEvent_C ∗event)
- void graphiti_sendACK (GraphitiHandle ∗handle)
- void graphiti_sendNACK (GraphitiHandle ∗handle)

- void graphiti_getSoftwareVersion (GraphitiHandle ∗handle)
- void graphiti_updateDisplay (GraphitiHandle ∗handle, const uint8_t ∗screen_data, int length)
- void graphiti_setDisplay (GraphitiHandle ∗handle)
- void graphiti_clearDisplay (GraphitiHandle ∗handle)
- void graphiti_updateSinglePixel (GraphitiHandle ∗handle, int row, int column, int height, int blinkRate)
- void graphiti_updateSingleRow (GraphitiHandle ∗handle, int row, const uint8_t ∗rowData, int length)
- void graphiti_updateSingleColumn (GraphitiHandle ∗handle, int column, const uint8_t ∗columnData, int length)
- void graphiti_getAllPixelsPositionStatus (GraphitiHandle ∗handle)
- void graphiti_getSinglePixelPositionStatus (GraphitiHandle ∗handle, int row, int column)
- void graphiti_getSingleRowPixelPositionStatus (GraphitiHandle ∗handle, uint8_t row)
- void graphiti_getSingleColumnPixelPositionStatus (GraphitiHandle ∗handle, uint8_t column)
- void graphiti_showMessage (GraphitiHandle ∗handle, const char ∗message, const uint8_t ∗selectFlags, int flagLength, int blinkRate)
- void graphiti_setCursor (GraphitiHandle ∗handle, int row, int column, int height, int length, int blinkRate)
- void graphiti_sendImageInterruptible (GraphitiHandle ∗handle, const char ∗name, const char ∗filepath)
- void graphiti_sendImageBlocking (GraphitiHandle ∗handle, const char ∗name, const char ∗filepath)
- void graphiti_setKeyEvent (GraphitiHandle ∗handle, bool enabled)
- void graphiti_setTouchEvent (GraphitiHandle ∗handle, bool enabled)
- void graphiti_getLastTouchPointStatus (GraphitiHandle ∗handle)
- void graphiti_vibratorControlCommand (GraphitiHandle ∗handle, int frequencyRange, int dutyCycle, int duration)
- void graphiti_getDateAndTime (GraphitiHandle ∗handle)
- void graphiti_setDateAndTime (GraphitiHandle ∗handle, int day, int month, int year, int hour, int minute, int second)

5.26.1 Function Documentation

5.26.1.1 graphiti_clearDisplay()

```
void graphiti_clearDisplay (
            GraphitiHandle ∗ handle)
```

5.26.1.2 graphiti_clearScreen()

```
void graphiti_clearScreen (
            GraphitiHandle ∗ handle)
```

5.26.1.3 graphiti_create()

```
GraphitiHandle ∗ graphiti_create ()
```

5.26.1.4 graphiti_create_with_connection()

```
GraphitiHandle ∗ graphiti_create_with_connection (
            void ∗ connection)
```

5.26.1.5 graphiti_destroy()

```
void graphiti_destroy (
            GraphitiHandle * handle)
```

5.26.1.6 graphiti_freeDisplayStatusEvent()

```
void graphiti_freeDisplayStatusEvent (
            DisplayStatusEvent_C * event)
```

5.26.1.7 graphiti_freeDrawEvent()

```
void graphiti_freeDrawEvent (
            DrawEvent_C * event)
```

5.26.1.8 graphiti_freeKeyEvent()

```
void graphiti_freeKeyEvent (
            KeyEvent_C * event)
```

5.26.1.9 graphiti_freeString()

```
void graphiti_freeString (
            char * str)
```

5.26.1.10 graphiti_getAllPixelsPositionStatus()

```
void graphiti_getAllPixelsPositionStatus (
            GraphitiHandle * handle)
```

5.26.1.11 graphiti_getDateAndTime()

```
void graphiti_getDateAndTime (
            GraphitiHandle * handle)
```

5.26.1.12 graphiti_getLastTouchPointStatus()

```
void graphiti_getLastTouchPointStatus (
            GraphitiHandle * handle)
```

5.26.1.13 graphiti_getNextDisplayStatusEvent()

```
DisplayStatusEvent_C graphiti_getNextDisplayStatusEvent (
            GraphitiHandle * handle)
```

5.26.1.14 graphiti_getNextDrawEvent()

```
DrawEvent_C graphiti_getNextDrawEvent (
             GraphitiHandle * handle)
```

5.26.1.15 graphiti_getNextGestureEvent()

```
char * graphiti_getNextGestureEvent (
             GraphitiHandle * handle)
```

5.26.1.16 graphiti_getNextKeyEvent()

```
KeyEvent_C graphiti_getNextKeyEvent (
             GraphitiHandle * handle)
```

5.26.1.17 graphiti_getNextOutputEvent()

```
char * graphiti_getNextOutputEvent (
             GraphitiHandle * handle)
```

5.26.1.18 graphiti_getSingleColumnPixelPositionStatus()

```
void graphiti_getSingleColumnPixelPositionStatus (
             GraphitiHandle * handle,
             uint8_t column)
```

5.26.1.19 graphiti_getSinglePixelPositionStatus()

```
void graphiti_getSinglePixelPositionStatus (
             GraphitiHandle * handle,
             int row,
             int column)
```

5.26.1.20 graphiti_getSingleRowPixelPositionStatus()

```
void graphiti_getSingleRowPixelPositionStatus (
             GraphitiHandle * handle,
             uint8_t row)
```

5.26.1.21 graphiti_getSoftwareVersion()

```
void graphiti_getSoftwareVersion (
             GraphitiHandle * handle)
```

5.26.1.22 graphiti_index()

```
int graphiti_index (
             GraphitiHandle * handle,
             int row,
             int col)
```

5.26.1.23 graphiti_sendACK()

```
void graphiti_sendACK (
             GraphitiHandle * handle)
```

5.26.1.24 graphiti_sendImageBlocking()

```
void graphiti_sendImageBlocking (
             GraphitiHandle * handle,
             const char * name,
             const char * filepath)
```

5.26.1.25 graphiti_sendImageInterruptible()

```
void graphiti_sendImageInterruptible (
             GraphitiHandle * handle,
             const char * name,
             const char * filepath)
```

5.26.1.26 graphiti_sendNACK()

```
void graphiti_sendNACK (
             GraphitiHandle * handle)
```

5.26.1.27 graphiti_setConnection()

```
void graphiti_setConnection (
             GraphitiHandle * handle,
             void * connection)
```

5.26.1.28 graphiti_setCursor()

```
void graphiti_setCursor (
             GraphitiHandle * handle,
             int row,
             int column,
             int height,
             int length,
             int blinkRate)
```

5.26.1.29 graphiti_setDateAndTime()

```
void graphiti_setDateAndTime (
            GraphitiHandle * handle,
            int day,
            int month,
            int year,
            int hour,
            int minute,
            int second)
```

5.26.1.30 graphiti_setDisplay()

```
void graphiti_setDisplay (
            GraphitiHandle * handle)
```

5.26.1.31 graphiti_setKeyEvent()

```
void graphiti_setKeyEvent (
            GraphitiHandle * handle,
            bool enabled)
```

5.26.1.32 graphiti_setPin()

```
void graphiti_setPin (
            GraphitiHandle * handle,
            int row,
            int col,
            int height)
```

5.26.1.33 graphiti_setTouchEvent()

```
void graphiti_setTouchEvent (
            GraphitiHandle * handle,
            bool enabled)
```

5.26.1.34 graphiti_showMessage()

```
void graphiti_showMessage (
            GraphitiHandle * handle,
            const char * message,
            const uint8_t * selectFlags,
            int flagLength,
            int blinkRate)
```

5.26.1.35 graphiti_shutDownVCP()

```
void graphiti_shutDownVCP (
            GraphitiHandle * handle,
            const char * portName,
            bool keyEventsBool,
            bool touchEventsBool)
```

5.26.1.36 graphiti_sleep()

```
void graphiti_sleep (
            GraphitiHandle * handle,
            int time)
```

5.26.1.37 graphiti_startResponseThread()

```
void graphiti_startResponseThread (
            GraphitiHandle * handle)
```

5.26.1.38 graphiti_startUpVCP()

```
void graphiti_startUpVCP (
            GraphitiHandle * handle,
            const char * portName,
            bool keyEventsBool,
            bool touchEventsBool)
```

5.26.1.39 graphiti_stopResponseThread()

```
void graphiti_stopResponseThread (
            GraphitiHandle * handle)
```

5.26.1.40 graphiti_updateDisplay()

```
void graphiti_updateDisplay (
            GraphitiHandle * handle,
            const uint8_t * screen_data,
            int length)
```

5.26.1.41 graphiti_updateSingleColumn()

```
void graphiti_updateSingleColumn (
            GraphitiHandle * handle,
            int column,
            const uint8_t * columnData,
            int length)
```

5.26.1.42 graphiti_updateSinglePixel()

```
void graphiti_updateSinglePixel (
            GraphitiHandle * handle,
            int row,
            int column,
            int height,
            int blinkRate)
```

5.26.1.43 graphiti_updateSingleRow()

```
void graphiti_updateSingleRow (
            GraphitiHandle * handle,
            int row,
            const uint8_t * rowData,
            int length)
```

5.26.1.44 graphiti_vibratorControlCommand()

```
void graphiti_vibratorControlCommand (
            GraphitiHandle * handle,
            int frequencyRange,
            int dutyCycle,
            int duration)
```

5.27 lib/src/Extension.cpp File Reference

```
#include <cstdio>
#include <Graphiti/Extension.hpp>
```