GTKTerm

Generated by Doxygen 1.9.4

Chapter 1

GTKTerm: The source code architecture

This file describes the architecture of GTKTerm. GtkTerm has several objects and uses signals to communicate between these objects.

One of the subgoals is not to use any global variables but exchange data by the use of signals. For that only the array of signals is a global variable.

Use of GTKTerm/GtkTerm/gtkterm naming schema: In this document several ways of Upper/Lowercase combinations of GTKTerm is used:

- GTKTerm: The name of the application
- GtkTerm: The first part of the name of the object in the source code. For example: GtkTermWindow.
- gtk_term: The first part of the function of an object in the source code. For example: gtkterm_window_init

1.1 General description

GTKTerm is build with the GTK4 framework. It uses Gobjects and communicates (mostly) through signals.

GTKTerm is the main application object. It is a holder for the keyfile. The commandline interfaces uses the application object framework to handle all commandline options. The options are connected to the relevant GObjects by signals. Almost all objects have a 'public' and 'private' part. However the 'public' part is not globally known (except for GtkTerm application object).

The core of the application is the terminal. This is a VTE object and handles all communication to and from the serial port. The terminal window holds the configuration of the terminal window and the serial ports. The configuraton is copied from the GtkTerm application which holds the keyfile. It is copied back to the keyfile if it is saved. For now the GtkTerm application has just one terminal window. The architecture of GTKTerm is able to support multiple terminal windows in future releases.

1.2 Objects

This part lists an overview of all objects used in GTKTerm. For details about implementation please use the GTKTERM.pdf which is a Doxygen generated overview of the GTKTerm source code.

1.2.1 GtkTerm

GtkTerm is the main GtkApplication object for GTKTerm. It starts the gtkterm_window and handles the cmdline interface (CLI). Options given at the CLI are directly stored into the in memory keyfile. This in memory keyfile is the base for the configuration of the terminal windows. Getting configuration for the terminal window is done by signals for the [section] needed.

1.2.1.1 Members

1.2.1.2 Signals

1.2.1.3 Main functions

1.2.2 GtkTermWindow

GtkTermWindow is the main application window for GtkTerm. It creates all widgets and does the handling for the statusbar.

1.2.2.1 Members

1.2.2.2 Signals

1.2.2.3 Main functions

1.2.3 GtkTermTerminal

The terminal window in which all serial communication is shown. It is an VTE object and hold the configuration for the terminal and serial port. Each terminal window (just 1 for now) has only one serial interface which it connects to. It has 2 USR signals for communication from scripts. The terminal communicates (transmit/receive) with the serial port through the GtkTermBuffer.

1.2.3.1 Members

1.2.3.2 Signals

1.2.3.3 Main functions

1.2.4 GtkTermConfiguration

GtkTerm does all operation on the keyfile. It loads, saves the file and removes, checks sections. It also copies the section configuration info the configuration for the terminal.

1.2.4.1 Members

1.2.4.2 Signals

1.2.4.3 Main functions

1.2.5 GtkTermSerialPort

The Serial port object which does all communication to the serial port. It configures the port based on the port_conf from terminal. It has an in- and output stream to communicate with the serial device.

1.2.5.1 Members

1.2.5.2 Signals

1.2.5.3 Main functions

1.2.6 GtkTermBuffer

The buffer is the interface between the serial port and the terminal. The buffer receives the data from the serial port and 'translates' it to ASCII or HEX. It gets notified from the serial port when nieuw data is recieved. After conversion and markup (CR/LF Timestamps) it notifies the terminal which can feed it to the vte.

1.2.6.1 Members

1.2.6.2 Signals

1.2.6.3 Main functions

1.3 Resources

For the migration to gtk4 several links were used:

- https://docs.gtk.org/gobject/tutorial.html
- https://docs.gtk.org/gobject/concepts.html
- https://docs.gtk.org/glib/
- https://toshiocp.github.io/Gtk4-tutorial/index.html
- https://blogs.gnome.org/mcatanzaro/2022/07/27/common-glib-programming-errors/

The gtk room on IRC was a big support when asking questions.

Also special thanks to Jens Georg. Sellerie (an earlier fork of GTKTerm) was used as inspiration to solve some problems.

Chapter 2

Todo List

Member create_window (GApplication *, GtkTermWindow *)

remove and set it with properties.

Member gtkterm_activate (GApplication *app)

embed create_window in the gtkapplication window

Member gtkterm_buffer_add_data (GObject *object, gpointer data, gpointer user_data)

Make buffer also work with greater datastrings to make tbe above work.

Member gtkterm_config_options []

Update gtkterm.1.

Member gtkterm_serial_port_serial_data_received (GObject *source, GAsyncResult *res, gpointer user_data)

send to terminal window and show in infobar

Member gtkterm_terminal_constructed (GObject *object)

: convert to notify on message

: make configurable from the config file

Member gtkterm_terminal_port_status_changed (GObject *object, GParamSpec *pspec, gpointer user_data)

convert to notify signal on message...

Member gtkterm_window_init (GtkTermWindow *window)

: Rename it

Member GtkTermConfigurationItems [][CONF_ITEM_LENGTH]

Add the short option.

Member GtkTermTerminalPrivate::macros

convert macros -> object

Member on_gtkterm_quit (GSimpleAction *action, GVariant *parameter, gpointer user_data)

: Should be part of the Gtkterm application struct

Member on_gtkterm_send_raw (GSimpleAction *action, GVariant *parameter, gpointer user_data)

rewrite for GTKTerm

Member on_gtkterm_toggle_radio (GSimpleAction *action, GVariant *parameter, gpointer user_data)

rewrite for GTKTerm

Member on_gtkterm_toggle_state (GSimpleAction *, GVariant *, gpointer)

rewrite for GTKTerm

Member open_response_cb (GtkNativeDialog ∗dialog, int response_id, gpointer user_data)

rewrite for gtkterm

Member term_config_t::char_queue

This is not possible, so remove?

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

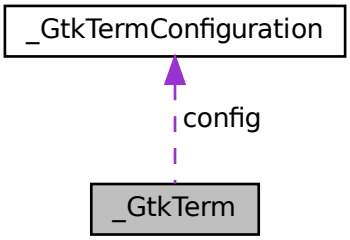
Class Documentation

5.1 _GtkTerm Struct Reference

The main GtkTerm application class.

`#include <gtkterm.h>`

Collaboration diagram for _GtkTerm:



Public Attributes

- GtkApplication parent_instance
- GOptionGroup * g_term_group
- GOptionGroup * g_port_group
- GOptionGroup * g_config_group
- GActionGroup * action_group

  App action group.
- GtkTermConfiguration * config

  The Key file with the configurations.
- char * section

  The section provided from the cli.

5.1.1 Detailed Description

The main GtkTerm application class.

All application specific variables are defined here.

5.1.2 Member Data Documentation

5.1.2.1 action_group

```
GActionGroup* _GtkTerm::action_group
```

App action group.

Referenced by gtkterm_init().

5.1.2.2 config

```
GtkTermConfiguration* _GtkTerm::config
```

The Key file with the configurations.

Referenced by gtkterm_init(), and gtkterm_terminal_constructed().

5.1.2.3 g_config_group

```
GOptionGroup* _GtkTerm::g_config_group
```

Referenced by gtkterm_add_cmdline_options(), and on_gtkterm_quit().

5.1.2.4 g_port_group

```
GOptionGroup* _GtkTerm::g_port_group
```

Referenced by gtkterm_add_cmdline_options(), and on_gtkterm_quit().

5.1.2.5 g_term_group

```
GOptionGroup* _GtkTerm::g_term_group
```

Referenced by gtkterm_add_cmdline_options(), and on_gtkterm_quit().

5.1.2.6 parent_instance

```
GtkApplication _GtkTerm::parent_instance
```

5.1.2.7 section

```
char* _GtkTerm::section
```

The section provided from the cli.

Referenced by gtkterm_init(), and on_gtkterm_quit().

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

- gtkterm.h

5.2 _GtkTermBuffer Struct Reference

Public Attributes

- GObject parent_instance

5.2.1 Member Data Documentation

5.2.1.1 parent_instance

```
GObject _GtkTermBuffer::parent_instance
```

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

- gtkterm_buffer.c

5.3 _GtkTermBufferClass Struct Reference

Public Attributes

- GObjectClass parent_class

5.3.1 Member Data Documentation

5.3.1.1 parent_class

```
GObjectClass _GtkTermBufferClass::parent_class
```

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

- gtkterm_buffer.c

5.4 _GtkTermConfiguration Struct Reference

Public Attributes

- GObject parent_instance

5.4.1 Member Data Documentation

5.4.1.1 parent_instance

```
GObject _GtkTermConfiguration::parent_instance
```

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

- gtkterm_configuration.c

5.5 _GtkTermConfigurationClass Struct Reference

Public Attributes

- GObjectClass parent_class

5.5.1 Member Data Documentation

5.5.1.1 parent_class

```
GObjectClass _GtkTermConfigurationClass::parent_class
```

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

- gtkterm_configuration.c

5.6 _GtkTermSerialPort Struct Reference

Public Attributes

- GObject parent_instance

5.6.1 Member Data Documentation

5.6.1.1 parent_instance

```
GObject _GtkTermSerialPort::parent_instance
```

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

- gtkterm_serial_port.c

5.7 _GtkTermSerialPortClass Struct Reference

Public Attributes

- GObjectClass parent_class

5.7.1 Member Data Documentation

5.7.1.1 parent_class

```
GObjectClass _GtkTermSerialPortClass::parent_class
```

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

- gtkterm_serial_port.c

5.8 _GtkTermTerminal Struct Reference

Public Attributes

- VteTerminal vte_object

5.8.1 Member Data Documentation

5.8.1.1 vte_object

```
VteTerminal _GtkTermTerminal::vte_object
```

The actual terminal object

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

- gtkterm_terminal.c

5.9 _GtkTermTerminalClass Struct Reference

Public Attributes

- VteTerminalClass vte_class

5.9.1 Member Data Documentation

5.9.1.1 vte_class

```
VteTerminalClass _GtkTermTerminalClass::vte_class
```

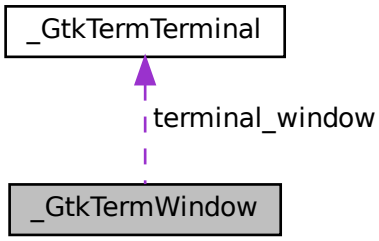
The documentation for this struct was generated from the following file:

- gtkterm_terminal.c

5.10 _GtkTermWindow Struct Reference

MainWindow specific variables here.

Collaboration diagram for _GtkTermWindow:



Public Attributes

- GtkApplicationWindow parent_instance
- GtkWidget * message
- GtkWidget * infobar
- GtkBox * statusbox
- GtkBox * status_config
- GtkWidget * menubutton
- GMenuModel * toolmenu
- GtkScrolledWindow * scrolled_window
- GtkTermTerminal * terminal_window
- GtkWidget * search_bar
- GActionGroup * action_group
- GtkWidget * status_config_message [3]
- GtkWidget * status_serial_signal [SERIAL_SIGNALS]
- GtkWidget * status_message
- int width
- int height
- bool maximized
- bool fullscreen

5.10.1 Detailed Description

MainWindow specific variables here.

5.10.2 Member Data Documentation

5.10.2.1 action_group

```
GActionGroup* _GtkTermWindow::action_group
```

Window action group

Referenced by gtkterm_window_init().

5.10.2.2 fullscreen

```
bool _GtkTermWindow::fullscreen
```

Window maximized?

Referenced by gtkterm_window_constructed(), gtkterm_window_init(), gtkterm_window_load_state(), gtkterm_window_size_allocate(
gtkterm_window_store_state(), and surface_state_changed().

5.10.2.3 height

```
int _GtkTermWindow::height
```

Window height

Referenced by gtkterm_window_constructed(), gtkterm_window_init(), gtkterm_window_load_state(), gtkterm_window_size_allocate(
and gtkterm_window_store_state().

5.10.2.4 infobar

```
GtkWidget* _GtkTermWindow::infobar
```

Infobar

Referenced by clicked_cb(), and gtkterm_show_infobar().

5.10.2.5 maximized

```
bool _GtkTermWindow::maximized
```

Window minimized?

Referenced by gtkterm_window_constructed(), gtkterm_window_init(), gtkterm_window_load_state(), gtkterm_window_size_allocate(
gtkterm_window_store_state(), and surface_state_changed().

5.10.2.6 menubutton

```
GtkWidget* _GtkTermWindow::menubutton
```

Toolbar

Referenced by gtkterm_window_init().

5.10.2.7 message

```
GtkWidget* _GtkTermWindow::message
```

Message for the infobar

Referenced by gtkterm_show_infobar(), and gtkterm_terminal_constructed().

5.10.2.8 parent_instance

```
GtkApplicationWindow _GtkTermWindow::parent_instance
```

5.10.2.9 scrolled_window

```
GtkScrolledWindow* _GtkTermWindow::scrolled_window
```

Make the terminal window scrolled

Referenced by create_window().

5.10.2.10 search_bar

```
GtkWidget* _GtkTermWindow::search_bar
```

Searchbar

5.10.2.11 status_config

```
GtkBox* _GtkTermWindow::status_config
```

Displays the actual used configuration

Referenced by config_status_bar().

5.10.2.12 status_config_message

```
GtkWidget* _GtkTermWindow::status_config_message[3]
```

Referenced by config_status_bar(), and update_statusbar().

5.10.2.13 status_message

```
GtkWidget* _GtkTermWindow::status_message
```

5.10.2.14 status_serial_signal

```
GtkWidget* _GtkTermWindow::status_serial_signal[SERIAL_SIGNALS]
```

Referenced by config_status_bar(), and gtkterm_window_set_signals().

5.10.2.15 statusbox

```
GtkBox* _GtkTermWindow::statusbox
```

Box for statusbar messages

Referenced by config_status_bar().

5.10.2.16 terminal_window

```
GtkTermTerminal* _GtkTermWindow::terminal_window
```

The terminal window

Referenced by create_window().

5.10.2.17 toolmenu

```
GMenuModel* _GtkTermWindow::toolmenu
```

Menu

Referenced by gtkterm_window_init().

5.10.2.18 width

```
int _GtkTermWindow::width
```

Window width

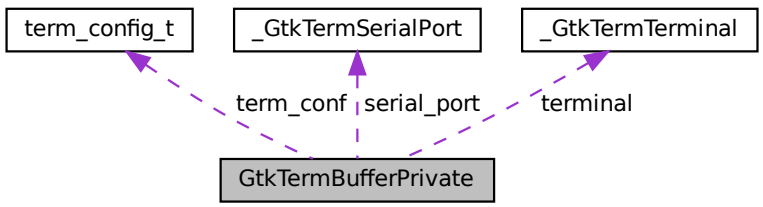
Referenced by gtkterm_window_constructed(), gtkterm_window_init(), gtkterm_window_load_state(), gtkterm_window_size_allocate(and gtkterm_window_store_state().

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

- gtkterm_window.c

5.11 GtkTermBufferPrivate Struct Reference

Collaboration diagram for GtkTermBufferPrivate:



Public Attributes

- char * buffer
- uint32_t tail
- term_config_t * term_conf
- bool lf_received
- bool cr_received
- bool need_to_write_timestamp
- GError * config_error
- GtkTermConfigurationState config_status
- GtkTermSerialPort * serial_port
- GtkTermTerminal * terminal
- GError * error

5.11.1 Member Data Documentation

5.11.1.1 buffer

```
char* GtkTermBufferPrivate::buffer
```

The actual buffer

Referenced by gtkterm_buffer_add_data(), gtkterm_buffer_finalize(), gtkterm_buffer_init(), and gtkterm_buffer_repage().

5.11.1.2 config_error

```
GError* GtkTermBufferPrivate::config_error
```

Error of the last file operation

Referenced by gtkterm_buffer_get_error(), and gtkterm_buffer_set_status().

5.11.1.3 config_status

```
GtkTermConfigurationState GtkTermBufferPrivate::config_status
```

Status when operating the buffer

Referenced by gtkterm_buffer_get_status(), and gtkterm_buffer_set_status().

5.11.1.4 cr_received

```
bool GtkTermBufferPrivate::cr_received
```

Reminder if we have a CR received

Referenced by gtkterm_buffer_add_data(), and gtkterm_buffer_init().

5.11.1.5 error

```
GError* GtkTermBufferPrivate::error
```

5.11.1.6 lf_received

```
bool GtkTermBufferPrivate::lf_received
```

Reminder if we have a LF received

Referenced by gtkterm_buffer_init().

5.11.1.7 need_to_write_timestamp

```
bool GtkTermBufferPrivate::need_to_write_timestamp
```

Reminder we need to write the timestamp (after a CR)

Referenced by gtkterm_buffer_add_data(), and gtkterm_buffer_init().

5.11.1.8 serial_port

```
GtkTermSerialPort* GtkTermBufferPrivate::serial_port
```

For connecting to the serial-data-received signals

Referenced by gtkterm_buffer_constructed(), gtkterm_buffer_dispose(), and gtkterm_buffer_set_property().

5.11.1.9 tail

```
uint32_t GtkTermBufferPrivate::tail
```

The tail of the buffer

Referenced by gtkterm_buffer_add_data(), gtkterm_buffer_init(), and gtkterm_buffer_repage().

5.11.1.10 term_conf

```
term_config_t* GtkTermBufferPrivate::term_conf
```

Referenced by gtkterm_buffer_add_data(), gtkterm_buffer_dispose(), and gtkterm_buffer_set_property().

5.11.1.11 terminal

```
GtkTermTerminal* GtkTermBufferPrivate::terminal
```

For connecting to the vte-data-received signals

Referenced by gtkterm_buffer_constructed(), gtkterm_buffer_dispose(), and gtkterm_buffer_set_property().

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

- gtkterm_buffer.c

5.12 GtkTermConfigurationPrivate Struct Reference

Public Attributes

- GKeyFile * key_file
- GFile * config_file
- GError * config_error
- GtkTermConfigurationState config_status
- bool config_is_dirty

5.12.1 Member Data Documentation

5.12.1.1 config_error

```
GError* GtkTermConfigurationPrivate::config_error
```

Error of the last file operation

Referenced by gtkterm_configuration_finalize(), gtkterm_configuration_get_error(), gtkterm_configuration_init(), and gtkterm_configuration_set_status().

5.12.1.2 config_file

```
GFile* GtkTermConfigurationPrivate::config_file
```

The config file

Referenced by gtkterm_configuration_check_configuration_file(), gtkterm_configuration_init(), gtkterm_configuration_load_keyfile(), gtkterm_configuration_save_keyfile(), and gtkterm_configuration_set_config_file().

5.12.1.3 config_is_dirty

```
bool GtkTermConfigurationPrivate::config_is_dirty
```

If changes are made but not yet saved

Referenced by check_keyfile(), gtkterm_configuration_init(), gtkterm_configuration_save_keyfile(), and on_set_config_options().

5.12.1.4 config_status

```
GtkTermConfigurationState GtkTermConfigurationPrivate::config_status
```

Status when operating configfiles

Referenced by gtkterm_configuration_get_status(), and gtkterm_configuration_set_status().

5.12.1.5 key_file

```
GKeyFile* GtkTermConfigurationPrivate::key_file
```

The memory loaded keyfile

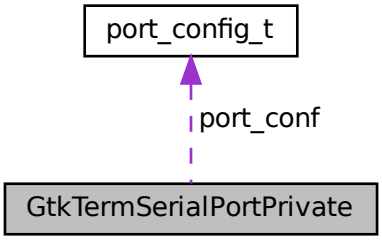
Referenced by check_keyfile(), gtkterm_configuration_check_configuration_file(), gtkterm_configuration_copy_section(), gtkterm_configuration_default_configuration(), gtkterm_configuration_init(), gtkterm_configuration_list_config(), gtkterm_configuration_load_keyfile(), gtkterm_configuration_load_serial_config(), gtkterm_configuration_load_terminal_config(), gtkterm_configuration_print_section(), gtkterm_configuration_remove_section(), gtkterm_configuration_save_keyfile(), gtkterm_configuration_validate(), and on_set_config_options().

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

- gtkterm_configuration.c

5.13 GtkTermSerialPortPrivate Struct Reference

Collaboration diagram for GtkTermSerialPortPrivate:



Public Attributes

- GUdevClient * udev_client
- port_config_t * port_conf
- struct termios port_termios
- GOutputStream * output_stream
- GInputStream * input_stream
- GCancellable * cancellable
- unsigned int control_signal_timeout
- int port_fd
- int control_signals
- GtkTermSerialPortState port_status
- GError * port_error

5.13.1 Member Data Documentation

5.13.1.1 cancellable

```
GCancellable* GtkTermSerialPortPrivate::cancellable
```

Allowing canceling async operation (reading port)

Referenced by gtkterm_serial_port_close(), gtkterm_serial_port_open(), gtkterm_serial_port_serial_data_received(), and gtkterm_serial_port_serial_data_transmit().

5.13.1.2 control_signal_timeout

```
unsigned int GtkTermSerialPortPrivate::control_signal_timeout
```

Interval to check/update serial control signals

Referenced by gtkterm_serial_port_close(), and gtkterm_serial_port_open().

5.13.1.3 control_signals

```
int GtkTermSerialPortPrivate::control_signals
```

The last serial signals (updated from the timeout)

Referenced by gtkterm_serial_port_control_signals_read(), gtkterm_serial_port_get_property(), gtkterm_serial_port_get_signals(), and gtkterm_serial_port_init().

5.13.1.4 input_stream

```
GInputStream* GtkTermSerialPortPrivate::input_stream
```

The incomming stream from the device

Referenced by gtkterm_serial_port_close(), gtkterm_serial_port_open(), and gtkterm_serial_port_serial_data_received().

5.13.1.5 output_stream

```
GOutputStream* GtkTermSerialPortPrivate::output_stream
```

The outgoing stream to the device

Referenced by gtkterm_serial_port_close(), gtkterm_serial_port_open(), and gtkterm_serial_port_serial_data_transmit().

5.13.1.6 port_conf

```
port_config_t* GtkTermSerialPortPrivate::port_conf
```

The configuration for the serial port

Referenced by gtkterm_serial_port_config(), gtkterm_serial_port_device_monitor(), gtkterm_serial_port_event_udev(), gtkterm_serial_port_get_string(), gtkterm_serial_port_lock(), gtkterm_serial_port_open(), gtkterm_serial_port_read_signals(), gtkterm_serial_port_serial_data_transmit(), gtkterm_serial_port_set_property(), and gtkterm_serial_port_unlock().

5.13.1.7 port_error

```
GError* GtkTermSerialPortPrivate::port_error
```

Referenced by gtkterm_serial_port_finalize(), gtkterm_serial_port_get_error(), and gtkterm_serial_port_set_status().

5.13.1.8 port_fd

```
int GtkTermSerialPortPrivate::port_fd
```

The port file descriptor

Referenced by gtkterm_serial_port_close(), gtkterm_serial_port_config(), gtkterm_serial_port_get_string(), gtkterm_serial_port_init(), gtkterm_serial_port_lock(), gtkterm_serial_port_open(), gtkterm_serial_port_read_signals(), gtkterm_serial_port_serial_data_transmit(), gtkterm_serial_port_set_signals(), and gtkterm_serial_port_unlock().

5.13.1.9 port_status

```
GtkTermSerialPortState GtkTermSerialPortPrivate::port_status
```

State of the serial port

Referenced by gtkterm_serial_port_get_property(), gtkterm_serial_port_get_status(), and gtkterm_serial_port_set_status().

5.13.1.10 port_termios

```
struct termios GtkTermSerialPortPrivate::port_termios
```

Saved port termios configuration for this port

Referenced by gtkterm_serial_port_close(), and gtkterm_serial_port_open().

5.13.1.11 udev_client

```
GUdevClient* GtkTermSerialPortPrivate::udev_client
```

The udev client for monitoring the device status

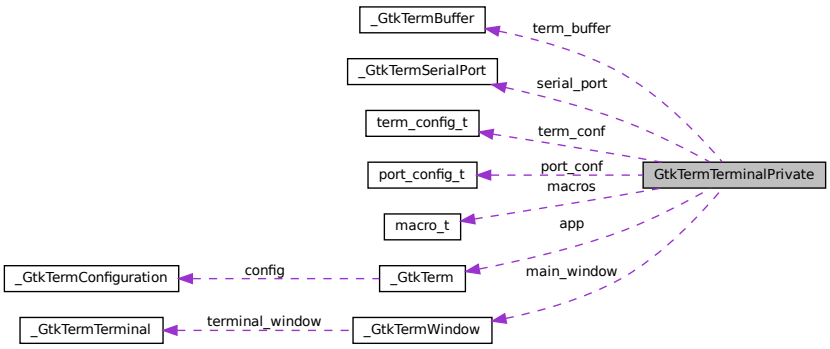
Referenced by gtkterm_serial_port_device_monitor().

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

- gtkterm_serial_port.c

5.14 GtkTermTerminalPrivate Struct Reference

Collaboration diagram for GtkTermTerminalPrivate:



Public Attributes

- GtkTermBuffer * term_buffer
- GtkTermSerialPort * serial_port
- term_config_t * term_conf
- port_config_t * port_conf
- macro_t * macros
- GtkTermTerminalView view_mode
- char * section
- GtkTerm * app
- GtkTermWindow * main_window

5.14.1 Member Data Documentation

5.14.1.1 app

`GtkTerm* GtkTermTerminalPrivate::app`

Pointer to the app for getting [section] and keyfile

Referenced by gtkterm_terminal_constructed(), and gtkterm_terminal_set_property().

5.14.1.2 macros

`macro_t* GtkTermTerminalPrivate::macros`

Todo convert macros -> object

5.14.1.3 main_window

`GtkTermWindow* GtkTermTerminalPrivate::main_window`

Pointer to the main window for updating the statusbar on changes

Referenced by gtkterm_terminal_constructed(), gtkterm_terminal_port_signals_changed(), gtkterm_terminal_port_status_changed(), and gtkterm_terminal_set_property().

5.14.1.4 port_conf

`port_config_t* GtkTermTerminalPrivate::port_conf`

Port configuration used in this terminal

Referenced by gtkterm_terminal_constructed(), and gtkterm_terminal_dispose().

5.14.1.5 section

```
char* GtkTermTerminalPrivate::section
```

Section used in this terminal for configuration from config file

Referenced by gtkterm_terminal_constructed(), gtkterm_terminal_dispose(), gtkterm_terminal_port_status_changed(), and gtkterm_terminal_set_property().

5.14.1.6 serial_port

```
GtkTermSerialPort* GtkTermTerminalPrivate::serial_port
```

The active serial port for this terminal

Referenced by gtkterm_terminal_constructed(), gtkterm_terminal_port_status_changed(), and gtkterm_terminal_vte_data_received().

5.14.1.7 term_buffer

```
GtkTermBuffer* GtkTermTerminalPrivate::term_buffer
```

Terminal buffer for serial port and terminal

Referenced by gtkterm_terminal_constructed().

5.14.1.8 term_conf

```
term_config_t* GtkTermTerminalPrivate::term_conf
```

The configuration loaded from the keyfile

Referenced by gtkterm_terminal_constructed(), gtkterm_terminal_dispose(), and gtkterm_terminal_vte_data_received().

5.14.1.9 view_mode

```
GtkTermTerminalView GtkTermTerminalPrivate::view_mode
```

Referenced by gtkterm_terminal_buffer_updated(), and gtkterm_terminal_init().

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

- gtkterm_terminal.c

5.15 macro_t Struct Reference

Define macro structure type.

`#include <macros.h>`

Public Attributes

- char * shortcut
  Shortcut of the macro.
- char * action
  Command to perform.
- GClosure * closure

5.15.1 Detailed Description

Define macro structure type.

todo: Migrate to GObject

5.15.2 Member Data Documentation

5.15.2.1 action

`char* macro_t::action`

Command to perform.

Referenced by convert_macros_to_string(), and convert_string_to_macros().

5.15.2.2 closure

`GClosure* macro_t::closure`

5.15.2.3 shortcut

`char* macro_t::shortcut`

Shortcut of the macro.

Referenced by convert_macros_to_string(), and convert_string_to_macros().

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

- macros.h

5.16 port_config_t Struct Reference

The typedef for the serial configuration.

`#include <gtkterm_serial_port.h>`

Public Attributes

- char * port
- long int baudrate
- int bits
- int stopbits
- GtkTermSerialPortParity parity
- GtkTermSerialPortFlowControl flow_control
- int rs485_rts_time_before_transmit
- int rs485_rts_time_after_transmit
- bool disable_port_lock

5.16.1 Detailed Description

The typedef for the serial configuration.

5.16.2 Member Data Documentation

5.16.2.1 baudrate

`long int port_config_t::baudrate`

300 - 600 - 1200 - ... - 2000000

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_config(), and gtkterm_serial_port_get_string().

5.16.2.2 bits

`int port_config_t::bits`

5 - 6 - 7 - 8

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_config(), and gtkterm_serial_port_get_string().

5.16.2.3 disable_port_lock

```
bool port_config_t::disable_port_lock
```

Lock the serial port to one terminal (can cause garbage if no)

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_lock(), and gtkterm_serial_port_unlock().

5.16.2.4 flow_control

```
GtkTermSerialPortFlowControl port_config_t::flow_control
```

0 : None, 1 : Xon/Xoff, 2 : RTS/CTS, 3 : RS485halfduplex

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_config(), gtkterm_serial_port_read_signals(), and gtkterm_serial_port_serial_data_transmit().

5.16.2.5 parity

```
GtkTermSerialPortParity port_config_t::parity
```

0 : None, 1 : Odd, 2 : Even

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_config(), and gtkterm_serial_port_get_string().

5.16.2.6 port

```
char* port_config_t::port
```

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_device_monito gtkterm_serial_port_event_udev(), gtkterm_serial_port_get_string(), gtkterm_serial_port_lock(), and gtkterm_serial_port_open().

5.16.2.7 rs485_rts_time_after_transmit

```
int port_config_t::rs485_rts_time_after_transmit
```

Waiting time between end of transmit and RTS on

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), and gtkterm_serial_port_serial_dat

5.16.2.8 rs485_rts_time_before_transmit

```
int port_config_t::rs485_rts_time_before_transmit
```

Waiting time between RTS onand start to transmit

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), and gtkterm_serial_port_serial_dat

5.16.2.9 stopbits

```
int port_config_t::stopbits
```

1 - 2

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_serial_config(), gtkterm_serial_port_config(), and gtkterm_serial_port_get_string().

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

- gtkterm_serial_port.h

5.17 term_config_t Struct Reference

The typedef for the terminal configuration.

```
#include <gtkterm_terminal.h>
```

Public Attributes

- bool block_cursor
- bool show_cursor
- char char_queue
- bool echo
- bool auto_lf
- bool auto_cr
- bool timestamp
- int delay
- int rows
- int columns
- int scrollback
- bool visual_bell
- GdkRGBA foreground_color
- GdkRGBA background_color
- PangoFontDescription * font

5.17.1 Detailed Description

The typedef for the terminal configuration.

5.17.2 Member Data Documentation

5.17.2.1 auto_cr

```
bool term_config_t::auto_cr
```

auto line feed

Referenced by gtkterm_buffer_add_data(), gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.2 auto_lf

```
bool term_config_t::auto_lf
```

local echo

Referenced by gtkterm_buffer_add_data(), gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.3 background_color

```
GdkRGBA term_config_t::background_color
```

Terminal Background color

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.4 block_cursor

```
bool term_config_t::block_cursor
```

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.5 char_queue

```
char term_config_t::char_queue
```

Show cursor in window.

Todo This is not possible, so remove?

Referenced by gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.6 columns

```
int term_config_t::columns
```

Number of rows in terminal

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.7 delay

```
int term_config_t::delay
```

Show timestamp in output

Referenced by gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.8 echo

```
bool term_config_t::echo
```

character in queue

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_vte_data_received().

5.17.2.9 font

```
PangoFontDescription* term_config_t::font
```

Terminal Foreground color

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.10 foreground_color

```
GdkRGBA term_config_t::foreground_color
```

Visual bell

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.11 rows

```
int term_config_t::rows
```

end of char delay: in ms

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.12 scrollback

```
int term_config_t::scrollback
```

Number of cols in terminal

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

5.17.2.13 show_cursor

```
bool term_config_t::show_cursor
```

Show a block shape cursor

Referenced by gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.14 timestamp

```
bool term_config_t::timestamp
```

auto return

Referenced by gtkterm_buffer_add_data(), gtkterm_configuration_copy_section(), and gtkterm_configuration_load_terminal_config().

5.17.2.15 visual_bell

```
bool term_config_t::visual_bell
```

Number of scrollback lines

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_load_terminal_config(), and gtkterm_terminal_constructed().

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

- gtkterm_terminal.h

Chapter 6

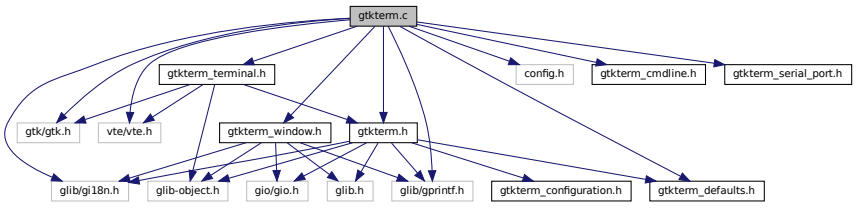
File Documentation

6.1 README_source.md File Reference

6.2 gtkterm.c File Reference

```
#include <gtk/gtk.h>
#include <vte/vte.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include "config.h"
#include "gtkterm_defaults.h"
#include "gtkterm.h"
#include "gtkterm_window.h"
#include "gtkterm_terminal.h"
#include "gtkterm_cmdline.h"
#include "gtkterm_serial_port.h"
```

Include dependency graph for gtkterm.c:



Functions

- static void on_gtkterm_quit (GSimpleAction ∗action, GVariant ∗parameter, gpointer user_data)

  Quitthe application.
- static void gtkterm_startup (GApplication ∗app)

  Startup the application.
- static void gtkterm_activate (GApplication ∗app)

  Activates the application.

- static void gtkterm_init (GtkTerm ∗app)

  The initialization of the application.
- static void gtkterm_class_init (GtkTermClass ∗class)

  Initializing the application class.
- int main (int argc, char ∗argv[ ])

  The main function.

Variables

- unsigned int gtkterm_signals [LAST_GTKTERM_SIGNAL]
- static GActionEntry gtkterm_entries [ ]

6.2.1 Function Documentation

6.2.1.1 gtkterm_activate()

```
static void gtkterm_activate (
            GApplication * app )  [static]
```

Activates the application.

Create the main window. The actual creation of the terminal will be done in the GtkTermWindow file.

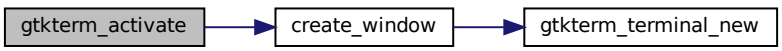
Parameters

app	The application

Todo embed create_window in the gtkapplication window

Referenced by gtkterm_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.2.1.2 gtkterm_class_init()

```
static void gtkterm_class_init (
            GtkTermClass * class )  [static]
```

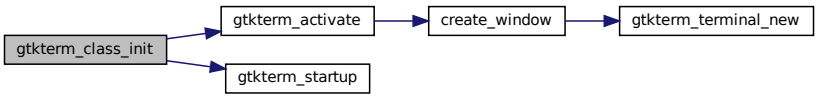
Initializing the application class.

Setting the signals and callback functions

Parameters

class	The application class

Here is the call graph for this function:



6.2.1.3 gtkterm_init()

```
static void gtkterm_init (
            GtkTerm * app )  [static]
```

The initialization of the application.

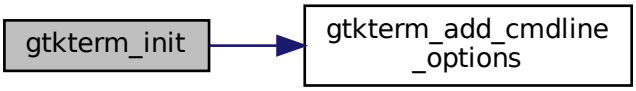
Setting the cli options and initialize the application variables

Parameters

app	The application

Set an action group for the app entries.

Initialize the config file and set the section to [default]Here is the call graph for this function:



6.2.1.4 gtkterm_startup()

```
static void gtkterm_startup (
            GApplication * app )  [static]
```

Startup the application.

Initiaze the builder and add menu resources to the application

Parameters

app	The application

Referenced by gtkterm_class_init().

Here is the caller graph for this function:

6.2.1.5 main()

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

The main function.

Parameters

argc	Number of cli arguments
argv	The cli arguments

6.2.1.6 on_gtkterm_quit()

```
static void on_gtkterm_quit (
            GSimpleAction * action,
            GVariant * parameter,
            gpointer user_data )  [static]
```

Quitthe application.

Is a callback function from the menubar and cleans up all memory, windows etc.

Parameters

action	Not used.
parameter	Not used.
user_data	The application we want to quit

Clean up memory

Todo : Should be part of the Gtkterm application struct

6.2.2 Variable Documentation

6.2.2.1 gtkterm_entries

```
GActionEntry gtkterm_entries[]  [static]
```

Initial value:

```
= {
  { "quit", on_gtkterm_quit, NULL, NULL, NULL },
}
```

Referenced by gtkterm_init().

6.2.2.2 gtkterm_signals

```
unsigned int gtkterm_signals[LAST_GTKTERM_SIGNAL]
```

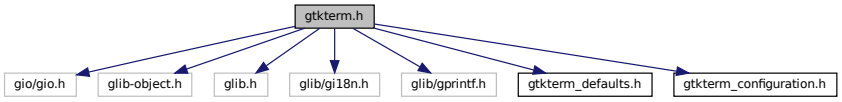
The gtkterm signals available

Referenced by gtkterm_buffer_add_data(), gtkterm_buffer_class_init(), gtkterm_class_init(), gtkterm_serial_port_class_constructed(), gtkterm_serial_port_serial_data_received(), gtkterm_terminal_class_init(), gtkterm_terminal_constructed(), gtkterm_terminal_port_signals_changed(), gtkterm_terminal_port_status_changed(), gtkterm_terminal_vte_data_received(), gtkterm_window_class_init(), on_list_config(), on_print_section(), on_remove_config(), and on_save_section().

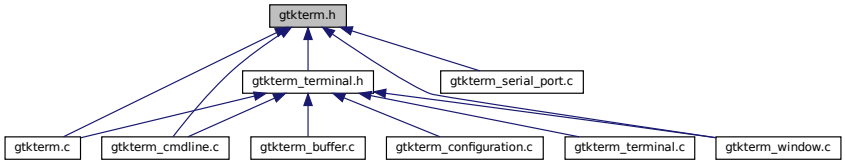
6.3 gtkterm.h File Reference

```
#include <gio/gio.h>
#include <glib-object.h>
#include <glib.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include "gtkterm_defaults.h"
#include "gtkterm_configuration.h"
```

Include dependency graph for gtkterm.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _GtkTerm

  The main GtkTerm application class.

Macros

- #define GTKTERM_TYPE_APP gtkterm_get_type()

Typedefs

- typedef struct _GtkTerm GtkTerm

Enumerations

- enum {
  SIGNAL_GTKTERM_LOAD_CONFIG , SIGNAL_GTKTERM_SAVE_CONFIG , SIGNAL_GTKTERM_LIST_CONFIG
  , SIGNAL_GTKTERM_REMOVE_SECTION ,
  SIGNAL_GTKTERM_PRINT_SECTION , SIGNAL_GTKTERM_COPY_SECTION , SIGNAL_GTKTERM_CONFIG_TERMINAL
  , SIGNAL_GTKTERM_CONFIG_SERIAL ,
  SIGNAL_GTKTERM_CONFIG_CHECK_FILE , SIGNAL_GTKTERM_TERMINAL_CHANGED , SIGNAL_GTKTERM_SERIAL_
  , SIGNAL_GTKTERM_VTE_DATA_RECEIVED ,
  SIGNAL_GTKTERM_SERIAL_DATA_RECEIVED , SIGNAL_GTKTERM_SERIAL_DATA_TRANSMIT ,
  SIGNAL_GTKTERM_SERIAL_SIGNALS_CHANGED , SIGNAL_GTKTERM_BUFFER_UPDATED ,
  LAST_GTKTERM_SIGNAL }

Variables

- unsigned int gtkterm_signals [ ]

6.3.1 Macro Definition Documentation

6.3.1.1 GTKTERM_TYPE_APP

```
#define GTKTERM_TYPE_APP gtkterm_get_type()
```

6.3.2 Typedef Documentation

6.3.2.1 GtkTerm

```
typedef struct _GtkTerm GtkTerm
```

6.3.3 Enumeration Type Documentation

6.3.3.1 anonymous enum

```
anonymous enum
```

The signals which are defined

Enumerator

SIGNAL_GTKTERM_LOAD_CONFIG	
SIGNAL_GTKTERM_SAVE_CONFIG	
SIGNAL_GTKTERM_LIST_CONFIG	
SIGNAL_GTKTERM_REMOVE_SECTION	
SIGNAL_GTKTERM_PRINT_SECTION	
SIGNAL_GTKTERM_COPY_SECTION	
SIGNAL_GTKTERM_CONFIG_TERMINAL	
SIGNAL_GTKTERM_CONFIG_SERIAL	
SIGNAL_GTKTERM_CONFIG_CHECK_FILE	
SIGNAL_GTKTERM_TERMINAL_CHANGED	
SIGNAL_GTKTERM_SERIAL_CONNECT	
SIGNAL_GTKTERM_VTE_DATA_RECEIVED	
SIGNAL_GTKTERM_SERIAL_DATA_RECEIVED	
SIGNAL_GTKTERM_SERIAL_DATA_TRANSMIT	
SIGNAL_GTKTERM_SERIAL_SIGNALS_CHANGED	
SIGNAL_GTKTERM_BUFFER_UPDATED	
LAST_GTKTERM_SIGNAL	

6.3.4 Variable Documentation

6.3.4.1 gtkterm_signals

```
unsigned int gtkterm_signals[]  [extern]
```

The gtkterm signals available

Referenced by gtkterm_buffer_add_data(), gtkterm_buffer_class_init(), gtkterm_class_init(), gtkterm_serial_port_class_constructed(), gtkterm_serial_port_serial_data_received(), gtkterm_terminal_class_init(), gtkterm_terminal_constructed(), gtkterm_terminal_port_signals_changed(), gtkterm_terminal_port_status_changed(), gtkterm_terminal_vte_data_received(), gtkterm_window_class_init(), on_list_config(), on_print_section(), on_remove_config(), and on_save_section().

6.4 gtkterm.h

Go to the documentation of this file.

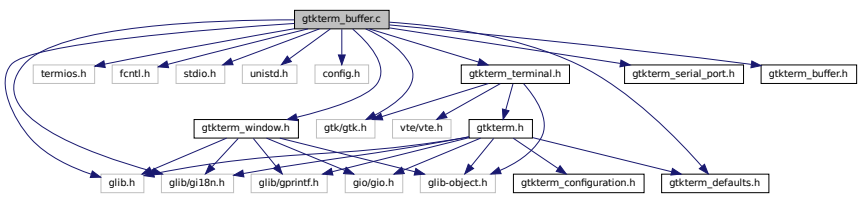
```
1
2 #ifndef GTKTERM_H
3 #define GTKTERM_H
4
5 #include <gio/gio.h>
6 #include <glib-object.h>
7 #include <glib.h>
8 #include <glib/gi18n.h>
9 #include <glib/gprintf.h>
10
11 #include "gtkterm_defaults.h"
12 #include "gtkterm_configuration.h"
13
15 enum {
16     SIGNAL_GTKTERM_LOAD_CONFIG,
17     SIGNAL_GTKTERM_SAVE_CONFIG,
```

```
18      SIGNAL_GTKTERM_LIST_CONFIG,
19      SIGNAL_GTKTERM_REMOVE_SECTION,
20      SIGNAL_GTKTERM_PRINT_SECTION,
21      SIGNAL_GTKTERM_COPY_SECTION,
22      SIGNAL_GTKTERM_CONFIG_TERMINAL,
23      SIGNAL_GTKTERM_CONFIG_SERIAL,
24      SIGNAL_GTKTERM_CONFIG_CHECK_FILE,
25      SIGNAL_GTKTERM_TERMINAL_CHANGED,
26      SIGNAL_GTKTERM_SERIAL_CONNECT,
27      SIGNAL_GTKTERM_VTE_DATA_RECEIVED,
28      SIGNAL_GTKTERM_SERIAL_DATA_RECEIVED,
29      SIGNAL_GTKTERM_SERIAL_DATA_TRANSMIT,
30      SIGNAL_GTKTERM_SERIAL_SIGNALS_CHANGED,
31      SIGNAL_GTKTERM_BUFFER_UPDATED,
32      LAST_GTKTERM_SIGNAL
33  };
34
35  extern unsigned int gtkterm_signals[];
36
37  G_BEGIN_DECLS
38
44  struct _GtkTerm {
45
46    GtkApplication parent_instance;
47
48    GOptionGroup *g_term_group;
49    GOptionGroup *g_port_group;
50    GOptionGroup *g_config_group;
51
52    GActionGroup *action_group;
53
54    GtkTermConfiguration *config;
55    char *section;
56  };
57
58  #define GTKTERM_TYPE_APP gtkterm_get_type()
59  G_DECLARE_FINAL_TYPE (GtkTerm, gtkterm, GTKTERM, APP, GtkApplication)
60  typedef struct _GtkTerm GtkTerm;
61
62  G_END_DECLS
63
64  #endif // GTKTERM_H
```

6.5 gtkterm_buffer.c File Reference

```
#include <gtk/gtk.h>
#include <glib.h>
#include <termios.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <config.h>
#include <glib/gi18n.h>
#include "gtkterm_window.h"
#include "gtkterm_defaults.h"
#include "gtkterm_terminal.h"
#include "gtkterm_serial_port.h"
#include "gtkterm_buffer.h"
```

Include dependency graph for gtkterm_buffer.c:



Classes

- struct GtkTermBufferPrivate
- struct _GtkTermBuffer
- struct _GtkTermBufferClass

Macros

- #define TIMESTAMP_SIZE 50

Enumerations

- enum {
  PROP_0 , PROP_SERIAL_PORT , PROP_TERMINAL , PROP_TERM_CONF ,
  N_PROPS }

Functions

- static GtkTermBufferState gtkterm_buffer_add_data (GObject *object, gpointer data, gpointer user_data)

  New data is available to add to the buffer.
- GtkTermBufferState gtkterm_buffer_set_status (GtkTermBuffer *self, GtkTermBufferState status, GError *error)

  Sets the status and error of the last operation.
- unsigned int insert_timestamp (char *buffer)

  Add a timestamp to the buffer.
- void gtkterm_buffer_repage (GtkTermBuffer *self)

  Repage the buffer to make room for new data.
- GtkTermBuffer * gtkterm_buffer_new (GtkTermSerialPort *serial_port, GtkTermTerminal *terminal, term_config_t *term_conf)

  Create a new buffer object.
- static void gtkterm_buffer_finalize (GObject *object)

  Finalizing the buffer class.
- static void gtkterm_buffer_constructed (GObject *object)

  Constructs the buffer.
- static void gtkterm_buffer_dispose (GObject *object)

  Called when distroying the buffer.
- static void gtkterm_buffer_set_property (GObject *object, unsigned int prop_id, const GValue *value, GParamSpec *pspec)

  Set the property of the GtkTermBuffer structure.
- static void gtkterm_buffer_class_init (GtkTermBufferClass *class)

  Initializing the buffer class.
- static void gtkterm_buffer_init (GtkTermBuffer *self)

  Initialize the buffer with size BUFFER_SIZE.
- GtkTermBufferState gtkterm_buffer_get_status (GtkTermBuffer *self)

  Return the latest status condiation for the buffer operation.
- GError * gtkterm_buffer_get_error (GtkTermBuffer *self)

  Return the latest error for the buffer operation.

Variables

- static GParamSpec * gtkterm_buffer_properties [N_PROPS] = {NULL}

6.5.1 Macro Definition Documentation

6.5.1.1 TIMESTAMP_SIZE

```
#define TIMESTAMP_SIZE 50
```

6.5.2 Enumeration Type Documentation

6.5.2.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_SERIAL_PORT	
PROP_TERMINAL	
PROP_TERM_CONF	
N_PROPS	

6.5.3 Function Documentation

6.5.3.1 gtkterm_buffer_add_data()

```
static GtkTermBufferState gtkterm_buffer_add_data (
            GObject * object,
            gpointer data,
            gpointer user_data )  [static]
```

New data is available to add to the buffer.

The buffer will signal the terminal new data is available.

Parameters

object	Not used.
data	The new byte-string of data for the buffer.
user_data	The buffer.

< Indicates from where to send data to the terminal

(When incoming size is larger than buffer, then just print the last BUFFER_SIZE characters and ignore all other at begin of buffer) In theory the str_size (8k) cannot be larger then the buffer size (128k)

Todo Make buffer also work with greater datastrings to make tbe above work.

When incoming size is larger than free space in the buffer then repage the buffer which will move the head. We use 2x the str_size to

If the auto CR or LF mode on, read the buffer to add LF before CR

If the previous character was a CR too, insert a newline

If we receive a normal char, and the previous one was a CR insert a newline

If we have timestamps configured and it is time to print one, print it

< remember until we have a new character to print

< Copy the string character to the buffer

< Increment for each stored character

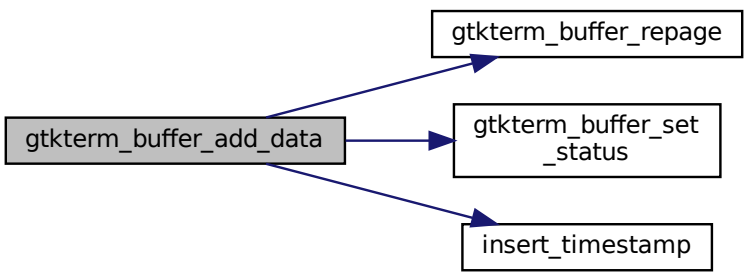
If we are growing out of the buffer, then repage

We dont need any timestamps or LF/CR so just copy the string into the buffer

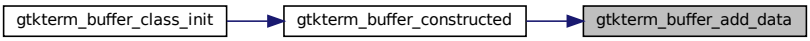
Send new data to the terminal

Referenced by gtkterm_buffer_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.2 gtkterm_buffer_class_init()

```
static void gtkterm_buffer_class_init (
            GtkTermBufferClass * class )  [static]
```

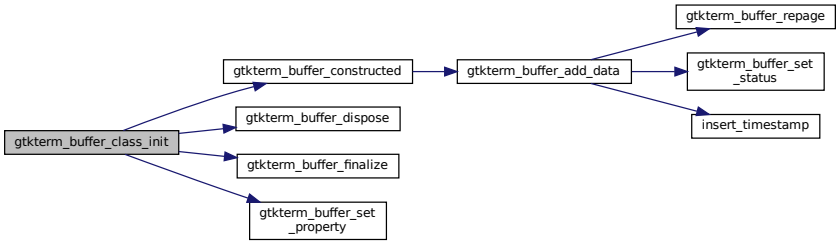
Initializing the buffer class.

Setting the properties and callback functions

Parameters

class	The buffer port class

Parameters to hand over at creation of the object We need the section to load the config from the keyfile.Here is the call graph for this function:



6.5.3.3 gtkterm_buffer_constructed()

```
static void gtkterm_buffer_constructed (
            GObject * object )  [static]
```

Constructs the buffer.

Setup signals etc.

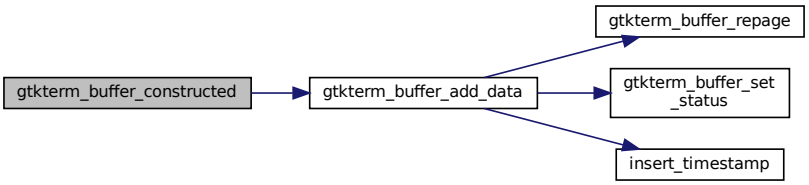
Parameters

object	The buffer object we are constructing.

Connect to data received signals from vte and serial

Referenced by gtkterm_buffer_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.5.3.4 gtkterm_buffer_dispose()

```
static void gtkterm_buffer_dispose (
            GObject * object )  [static]
```

Called when distroying the buffer.

This is used to clean up an freeing the variables in the buffer structure.

Parameters

object	The object.

Referenced by gtkterm_buffer_class_init().

Here is the caller graph for this function:

6.5.3.5 gtkterm_buffer_finalize()

```
static void gtkterm_buffer_finalize (
            GObject * object )  [static]
```

Finalizing the buffer class.

Clears the pointer of the buffer.

Parameters

object	The object which is finialized

Referenced by gtkterm_buffer_class_init().

Here is the caller graph for this function:

6.5.3.6 gtkterm_buffer_get_error()

```
GError * gtkterm_buffer_get_error (
            GtkTermBuffer * self )
```

Return the latest error for the buffer operation.

Parameters

self	The configuration for which the get the status for.

Returns

The latest error.

6.5.3.7 gtkterm_buffer_get_status()

```
GtkTermBufferState gtkterm_buffer_get_status (
            GtkTermBuffer * self )
```

Return the latest status condiation for the buffer operation.

Parameters

self	The configuration for which the get the status for.

Returns

The latest status.

6.5.3.8 gtkterm_buffer_init()

```
static void gtkterm_buffer_init (
            GtkTermBuffer * self )  [static]
```

Initialize the buffer with size BUFFER_SIZE.

Parameters

self	The buffer we are initializing.

6.5.3.9 gtkterm_buffer_new()

```
GtkTermBuffer * gtkterm_buffer_new (
            GtkTermSerialPort * serial_port,
            GtkTermTerminal * terminal,
            term_config_t * term_conf )
```

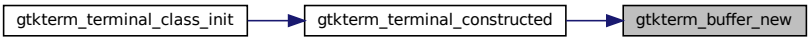
Create a new buffer object.

Returns

The buffer object.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.5.3.10 gtkterm_buffer_repage()

```
void gtkterm_buffer_repage (
            GtkTermBuffer * self )
```

Repage the buffer to make room for new data.

When repaging thebuffer is moved 2 pages up. The head is always placed after a CR to make sure we start with a new string.

Referenced by gtkterm_buffer_add_data().

Here is the caller graph for this function:

6.5.3.11 gtkterm_buffer_set_property()

```
static void gtkterm_buffer_set_property (
            GObject * object,
            unsigned int prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Set the property of the GtkTermBuffer structure.

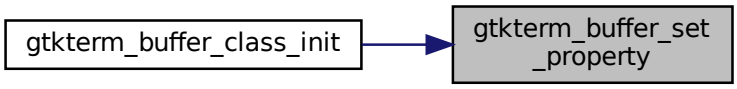
This is used to initialize the variables when creating a new buffer

Parameters

object	The object.
prop↩ _id	The id of the property to set.
value	The value for the property
pspec	Metadata for property setting.

Referenced by gtkterm_buffer_class_init().

Here is the caller graph for this function:



6.5.3.12 gtkterm_buffer_set_status()

```
GtkTermBufferState gtkterm_buffer_set_status (
            GtkTermBuffer * self,
            GtkTermBufferState status,
            GError * error )
```

Sets the status and error of the last operation.

Parameters

self	The configuration for which the get the status for.
status	The status to be set.
error	The error message (can be NULL)

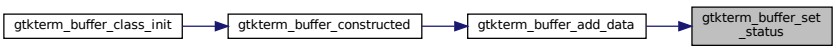
Returns

The latest status.

If there is a previous error, clear it

Referenced by gtkterm_buffer_add_data().

Here is the caller graph for this function:



6.5.3.13 insert_timestamp()

```
unsigned int insert_timestamp (
            char * buffer )
```

Add a timestamp to the buffer.

Assumes that buffer always has space for timestamp (TIMESTAMP_SIZE)

Parameters

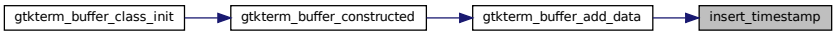
buffer	Points to location where timestamp will be inserted.

Returns

unsigned int Length of the timestamp added.

Referenced by gtkterm_buffer_add_data().

Here is the caller graph for this function:



6.5.4 Variable Documentation

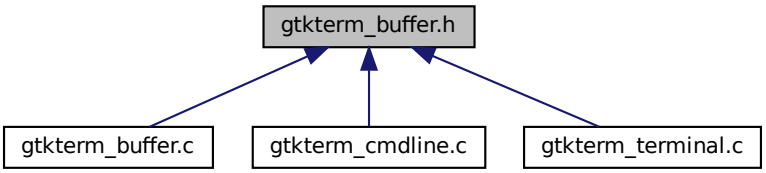
6.5.4.1 gtkterm_buffer_properties

```
GParamSpec* gtkterm_buffer_properties[N_PROPS] = {NULL}  [static]
```

Referenced by gtkterm_buffer_class_init().

6.6 gtkterm_buffer.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define GTKTERM_BUFFER_H
- #define GTKTERM_TYPE_BUFFER gtkterm_buffer_get_type ()

Typedefs

- typedef struct _GtkTermBuffer GtkTermBuffer

Enumerations

- enum GtkTermBufferState { GTKTERM_BUFFER_SUCCESS , GTKTERM_BUFFER_NOT_INITALIZED , GTKTERM_BUFFER_OVERFLOW , GTKTERM_BUFFER_LAST }

  Enum buffer_error id.

Functions

- GtkTermBuffer * gtkterm_buffer_new (GtkTermSerialPort *, GtkTermTerminal *, term_config_t *)

  Create a new buffer object.
- G_END_DECLS GtkTermBufferState gtkterm_buffer_get_status (GtkTermBuffer *)

  Return the latest status condiation for the buffer operation.
- GError * gtkterm_buffer_get_error (GtkTermBuffer *)

  Return the latest error for the buffer operation.

6.6.1 Macro Definition Documentation

6.6.1.1 GTKTERM_BUFFER_H

```
#define GTKTERM_BUFFER_H
```

6.6.1.2 GTKTERM_TYPE_BUFFER

```
#define GTKTERM_TYPE_BUFFER gtkterm_buffer_get_type ()
```

6.6.2 Typedef Documentation

6.6.2.1 GtkTermBuffer

```
typedef struct _GtkTermBuffer GtkTermBuffer
```

6.6.3 Enumeration Type Documentation

6.6.3.1 GtkTermBufferState

```
enum GtkTermBufferState
```

Enum buffer_error id.

Many of the gtk_buffer functions return an error id.

Enumerator

GTKTERM_BUFFER_SUCCESS	
GTKTERM_BUFFER_NOT_INITALIZED	
GTKTERM_BUFFER_OVERFLOW	
GTKTERM_BUFFER_LAST	

6.6.4 Function Documentation

6.6.4.1 gtkterm_buffer_get_error()

```
GError * gtkterm_buffer_get_error (
            GtkTermBuffer * self )
```

Return the latest error for the buffer operation.

Parameters

self	The configuration for which the get the status for.

Returns

The latest error.

6.6.4.2 gtkterm_buffer_get_status()

```
G_END_DECLS GtkTermBufferState gtkterm_buffer_get_status (
            GtkTermBuffer * self )
```

Return the latest status condiation for the buffer operation.

Parameters

self	The configuration for which the get the status for.

Returns

The latest status.

6.6.4.3 gtkterm_buffer_new()

```
GtkTermBuffer * gtkterm_buffer_new (
            GtkTermSerialPort * serial_port,
            GtkTermTerminal * terminal,
            term_config_t * term_conf )
```

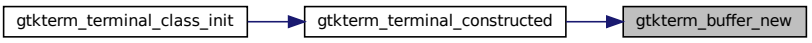
Create a new buffer object.

Returns

The buffer object.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.7 gtkterm_buffer.h

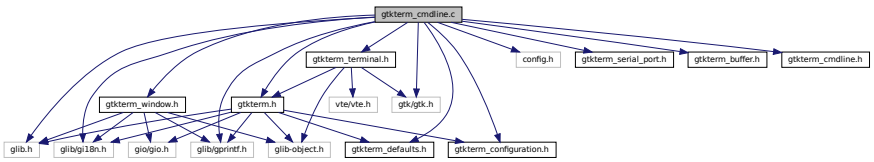
Go to the documentation of this file.

```
1 #ifndef GTMTERM_BUFFER_H
2 #define GTKTERM_BUFFER_H
3
10 typedef enum {
11     GTKTERM_BUFFER_SUCCESS,
12     GTKTERM_BUFFER_NOT_INITALIZED,
13     GTKTERM_BUFFER_OVERFLOW,
14     GTKTERM_BUFFER_LAST
15
16 } GtkTermBufferState;
17
18 G_BEGIN_DECLS
19
20 #define GTKTERM_TYPE_BUFFER gtkterm_buffer_get_type ()
21 G_DECLARE_FINAL_TYPE (GtkTermBuffer, gtkterm_buffer, GTKTERM, BUFFER, GObject)
22 typedef struct _GtkTermBuffer GtkTermBuffer;
23
24 GtkTermBuffer *gtkterm_buffer_new (GtkTermSerialPort *, GtkTermTerminal *, term_config_t *);
25
26 G_END_DECLS
27
28 GtkTermBufferState gtkterm_buffer_get_status (GtkTermBuffer *);
29 GError *gtkterm_buffer_get_error (GtkTermBuffer*);
30
31 #endif // GTKTERM_BUFFER_H
```

6.8 gtkterm_cmdline.c File Reference

```
#include <glib.h>
#include <glib/gi18n.h>
#include <gtk/gtk.h>
#include <glib/gprintf.h>
#include <config.h>
#include "gtkterm_defaults.h"
#include "gtkterm.h"
#include "gtkterm_window.h"
#include "gtkterm_terminal.h"
#include "gtkterm_serial_port.h"
#include "gtkterm_buffer.h"
#include "gtkterm_configuration.h"
#include "gtkterm_cmdline.h"
```

Include dependency graph for gtkterm_cmdline.c:



Functions

- static bool on_remove_config (const char *name, const char *value, gpointer data, GError **error)

  Removes a configuration sectons.
- static bool on_save_section (const char *name, const char *value, gpointer data, GError **error)

  Saves a configuration sectons.

- static bool on_print_section (const char *name, const char *value, gpointer data, GError **error)

  Prints a configuration sectons.
- static bool on_list_config (const char *name, const char *value, gpointer data, GError **error)

  List all configurations.
- static bool on_use_config (const char *name, const char *value, gpointer data, GError **error)

  Sets the use of a configuration section.
- void gtkterm_add_cmdline_options (GtkTerm *app)

  Add the commandline options.

Variables

- static GOptionEntry gtkterm_config_options [ ]

  GOptionEntry mappings.
- static GOptionEntry gtkterm_term_options [ ]

  Longname CLI options.
- static GOptionEntry gtkterm_port_options [ ]

6.8.1 Function Documentation

6.8.1.1 gtkterm_add_cmdline_options()

```
void gtkterm_add_cmdline_options (
            GtkTerm * app )
```

Add the commandline options.

Commandline options are grouped. So each group is created. Each group has app as parameter passed through the callback.

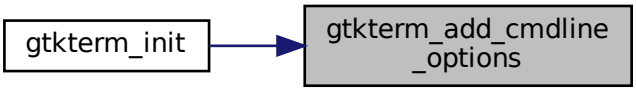
Parameters

app	The application

Pass app as data to the new created group

Referenced by gtkterm_init().

Here is the caller graph for this function:



6.8.1.2 on_list_config()

```
static bool on_list_config (
             const char * name,
             const char * value,
             gpointer data,
             GError ** error )  [static]
```

List all configurations.

The functions emits a signal which is connected to list all configurations. After printing, the g_application_quit is called for exiting the application (we only want to list the configs, not to start up GTKTerm)

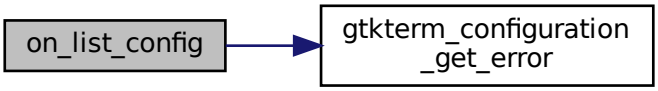
Parameters

name	Not used.
value	Not used.
data	The application app.
error	Error (not used).

Returns

true on succes (we will not get there).

Signal to list the comfigurations and dump it to the cliHere is the call graph for this function:



6.8.1.3 on_print_section()

```
static bool on_print_section (
             const char * name,
             const char * value,
             gpointer data,
             GError ** error )  [static]
```

Prints a configuration sectons.

The functions emits a signal which is connected to the config print function. After printing, the g_application_quit is called for exiting the application (we only want to print the section, not to start up GTKTerm)

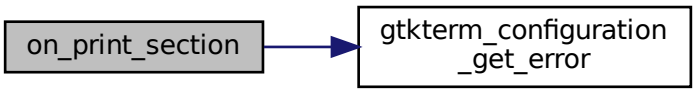
Parameters

name	Not used.
value	The section we want to print.
data	The application app.
error	Error (not used).

Returns

true on succes (we will not get there).

Signal to load the configuration and dump it to the cliHere is the call graph for this function:



6.8.1.4 on_remove_config()

```
static bool on_remove_config (
             const char * name,
             const char * value,
             gpointer data,
             GError ** error )  [static]
```

Removes a configuration sectons.

The functions emits a signal which is connected to the config remove function. After removing, the g_application↩
_quit is called for exiting the application (we only want to remove the section, not to start up GTKTerm).

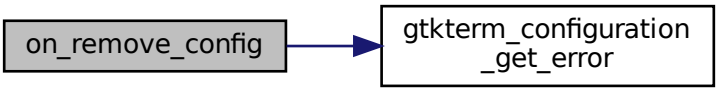
Parameters

name	Not used.
value	The section we want to remove.
data	The application app.
error	Error (not used).

Returns

true on succes (we will not get there).

Signal to load the configuration and dump it to the cliHere is the call graph for this function:



6.8.1.5 on_save_section()

```
static bool on_save_section (
             const char * name,
             const char * value,
             gpointer data,
             GError ** error )  [static]
```

Saves a configuration sectons.

The functions emits a signal which is connected to the config save function. After saving, the g_application_quit is called for exiting the application (we only want to save the section, not to start up GTKTerm). If we want to save cli options we have to put the save option as last parameter.

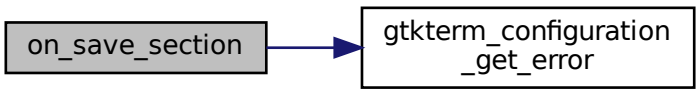
Parameters

name	Not used.
value	The section we want to save.
data	The application app.
error	Error (not used).

Returns

true on succes (we will not get there).

Signal to save the configuration and dump it to the cliHere is the call graph for this function:



6.8.1.6 on_use_config()

```
static bool on_use_config (
             const char * name,
             const char * value,
             gpointer data,
             GError ** error )  [static]
```

Sets the use of a configuration section.

This is used as input for config options or starting GTKTerm with the section active.

Parameters

name	Not used.
value	The section we want to use.
data	The application app.
error	Error (not used).

Returns

true on succes (continues startup). False if the configurationname is too long.

6.8.2 Variable Documentation

6.8.2.1 gtkterm_config_options

```
GOptionEntry gtkterm_config_options[]  [static]
```

Initial value:

```
= {
    {"show_config", 'V', 0, G_OPTION_ARG_CALLBACK, on_print_section, N_("Show configuration"),
      "[configuration]"},
    {"save_config", 'S', G_OPTION_FLAG_NO_ARG, G_OPTION_ARG_CALLBACK, on_save_section, N_("Save
      configuration"), NULL},
    {"remove_config", 'R', 0, G_OPTION_ARG_CALLBACK, on_remove_config, N_("Remove configuration"),
      "[configuration]"},
    {"use_config", 'U', 0, G_OPTION_ARG_CALLBACK, on_use_config, N_("Use configuration (must be first
      argument)"), "[configuration]"},
    {"list_config", 'L', G_OPTION_FLAG_NO_ARG, G_OPTION_ARG_CALLBACK, on_list_config, N_("List all
      configurations"), NULL},
    {NULL}
}
```

GOptionEntry mappings.

We use callback in GOptionEntry. So we can directly put them in the Terminal configuration instead of handing over a pointer from the config.

Todo Update gtkterm.1.

Referenced by gtkterm_add_cmdline_options().

6.8.2.2 gtkterm_port_options

```
GOptionEntry gtkterm_port_options[]  [static]
```

Initial value:

```
= {
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_PARITY], 'a', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("Parity(default = none)"), "<odd|even|none>"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_BITS], 'b', 0, G_OPTION_ARG_CALLBACK, on_set_config_options,
      N_("Number of bits (default 8)"), "<7|8> "},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_PORT], 'p', 0, G_OPTION_ARG_CALLBACK, on_set_config_options,
      N_("Serial port device (default /dev/ttyS0)"), "[device]"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_BAUDRATE], 's', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("Serial port baudrate(default 9600bps)"), "[baudrate]",},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_STOPBITS], 't', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options,  N_("Number of stopbits (default 1)"), "<1|2>"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_FLOW_CONTROL], 'w', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("Flow control (default none)"), "<Xon|RTS|RS485|none>"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_RS485_RTS_TIME_BEFORE_TX], 'x', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("For RS485, time in ms before transmit with RTS on"), "[ms]"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_RS485_RTS_TIME_AFTER_TX], 'y', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("For RS485, time in ms after transmit with RTS on"), "[ms]"},
    {GtkTermConfigurationItems[CONF_ITEM_SERIAL_DISABLE_PORT_LOCK], 'l', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options,  N_("Disable lock serial port "), "<on|off>"},
    {NULL}
}
```

Referenced by gtkterm_add_cmdline_options().

6.8.2.3 gtkterm_term_options

```
GOptionEntry gtkterm_term_options[]  [static]
```

Initial value:

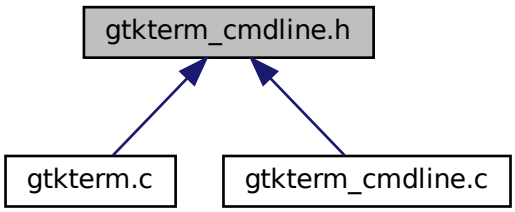
```
= {
    {GtkTermConfigurationItems[CONF_ITEM_TERM_WAIT_DELAY], 'd', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("End of line delay in ms (default none)"), "<ms>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_ECHO], 'e', 0, G_OPTION_ARG_CALLBACK, on_set_config_options,
      N_("Local echo"),  "<on|off>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_RAW_FILENAME], 'f', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options,  N_ ("Default file to send (default none)"), "<filename>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_WAIT_CHAR], 'r', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_("Wait for a special char at end of line (default none)"), "<character in
      HEX>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_ROWS], 'o', 0, G_OPTION_ARG_CALLBACK, on_set_config_options,
      N_ ("Terminal rows (default 25)"), "<rows>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_COLS], 'c', 0, G_OPTION_ARG_CALLBACK, on_set_config_options,
      N_ ("Terminal cols (default 80)"), "<cols>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_AUTO_CR], 'g', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_ ("Auto LF (default on)"), "<on|off>"},
    {GtkTermConfigurationItems[CONF_ITEM_TERM_AUTO_LF], 'h', 0, G_OPTION_ARG_CALLBACK,
      on_set_config_options, N_ ("Auto CR (default on)"), "<on|off>"},
    {NULL}
}
```

Longname CLI options.

Referenced by gtkterm_add_cmdline_options().

6.9 gtkterm_cmdline.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void gtkterm_add_cmdline_options (GtkTerm *app)

  Add the commandline options.

6.9.1 Function Documentation

6.9.1.1 gtkterm_add_cmdline_options()

```
void gtkterm_add_cmdline_options (
            GtkTerm * app )
```

Add the commandline options.

Commandline options are grouped. So each group is created. Each group has app as parameter passed through the callback.

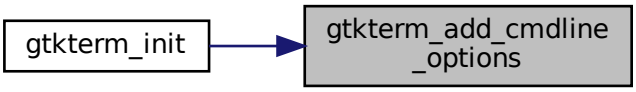
Parameters

app	The application

Pass app as data to the new created group

Referenced by gtkterm_init().

Here is the caller graph for this function:



6.10 gtkterm_cmdline.h

Go to the documentation of this file.

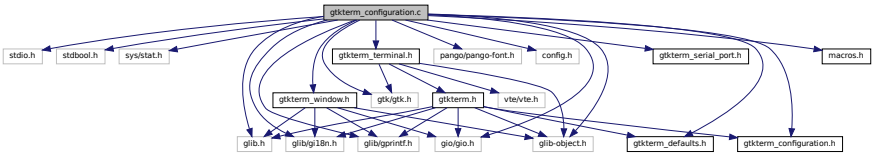
```
1 /*****************************************************************************/
2 /* gtkterm_cmdline.h                                                         */
3 /* ---------                                                                 */
4 /*           GTKTerm Software                                                */
5 /*                       (c) Julien Schmitt                                  */
6 /*                                                                           */
7 /* ------------------------------------------------------------------------- */
8 /*                                                                           */
9 /*   Purpose                                                                 */
10 /*      Reads the command line                                                */
11 /*      - Header file -                                                       */
12 /*                                                                           */
13 /*   ChangeLog                                                               */
14 /*      - 2.0 :  migrated to GTK4                                             */
15 /*      - 0.98 :  file creation by Julien                                     */
16 /*                                                                           */
17 /*****************************************************************************/
18
19 #ifndef GTKTERM_CMDLINE_H
20 #define GTKTERM_CMDLINE_H
21
22 void gtkterm_add_cmdline_options (GtkTerm *app);
23
24 #endif // GTKTERM_CMDLINE_H
```

6.11 gtkterm_configuration.c File Reference

```
#include <stdio.h>
#include <stdbool.h>
#include <sys/stat.h>
#include <glib.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include <glib-object.h>
#include <gtk/gtk.h>
#include <gio/gio.h>
#include <pango/pango-font.h>
#include "config.h"
#include "gtkterm_defaults.h"
#include "gtkterm_window.h"
#include "gtkterm_serial_port.h"
#include "gtkterm_terminal.h"
#include "gtkterm_configuration.h"
```

```
#include "macros.h"
```

Include dependency graph for gtkterm_configuration.c:



Classes

- struct GtkTermConfigurationPrivate
- struct _GtkTermConfiguration
- struct _GtkTermConfigurationClass

Functions

- static int gtkterm_configuration_load_keyfile (GtkTermConfiguration *, gpointer)

  Callback functions for signals.
- static int gtkterm_configuration_save_keyfile (GtkTermConfiguration *self, gpointer user_data)

  Save the in momeory keyfile to file).
- static int gtkterm_configuration_list_config (GtkTermConfiguration *self, gpointer user_data)

  Lists all sections in the keyfile.
- static int gtkterm_configuration_print_section (GtkTermConfiguration *self, gpointer data, gpointer user_data)

  Print the section to CLI.
- static int gtkterm_configuration_remove_section (GtkTermConfiguration *self, gpointer data, gpointer user↩ _data)

  Remove a section from the GKeyFile.
- static int gtkterm_configuration_set_config_file (GtkTermConfiguration *self, gpointer user_data)

  Check if the file exists in the old/new location.
- static term_config_t * gtkterm_configuration_load_terminal_config (GtkTermConfiguration *self, gpointer data, gpointer user_data)

  Load the terminal configuration from keyfile.
- static port_config_t * gtkterm_configuration_load_serial_config (GtkTermConfiguration *self, gpointer data, gpointer user_data)

  Load the portconfiguration from keyfile.
- static int gtkterm_configuration_copy_section (GtkTermConfiguration *self, gpointer sect, gpointer pc, gpointer tc, gpointer user_data)

  Copy the active configuration into [section] of the Key file.
- static void set_color (GdkRGBA *, float, float, float, float)

  Functions for internal use only.
- static GtkTermConfigurationState gtkterm_configuration_check_configuration_file (GtkTermConfiguration *self)

  Check if the configuration file exists on disk.
- void gtkterm_configuration_default_configuration (GtkTermConfiguration *self, char *section)

  Create a new configuration with defaults.
- GtkTermConfigurationState gtkterm_configuration_validate (GtkTermConfiguration *self, char *section)

  validate the configuration, given by the section.

- GtkTermConfigurationState gtkterm_configuration_set_status (GtkTermConfiguration ∗self, GtkTermConfigurationState status, GError ∗error)

  Sets the status and error of the last operation.
- static void gtkterm_configuration_finalize (GObject ∗object)

  Remote all pointers when removing the object.
- static void gtkterm_configuration_class_constructed (GObject ∗object)

  Connect callback functions to signals.
- static void gtkterm_configuration_class_init (GtkTermConfigurationClass ∗class)

  Initialize the class functions.
- static void gtkterm_configuration_init (GtkTermConfiguration ∗self)

  Initialize the class members.
- GtkTermConfigurationState check_keyfile (GtkTermConfiguration ∗self, char ∗section)

  Check if the keyfile is loaded into memory.
- GtkTermConfigurationState on_set_config_options (const char ∗name, const char ∗value, gpointer data, GError ∗∗error)

  Set the config option in the keyfile.
- GtkTermConfigurationState gtkterm_configuration_get_status (GtkTermConfiguration ∗self)

  Return the latest status condiation for the file operation.
- GError ∗ gtkterm_configuration_get_error (GtkTermConfiguration ∗self)

  Return the latest error for the file operation.

Variables

- const char GtkTermConfigurationItems [ ][CONF_ITEM_LENGTH]

  Configuration options.
- const char GtkTermCLIShortOption [ ][CONF_ITEM_LENGTH]

6.11.1 Function Documentation

6.11.1.1 check_keyfile()

```
GtkTermConfigurationState check_keyfile (
            GtkTermConfiguration * self,
            char * section )
```

Check if the keyfile is loaded into memory.

Loads the keyfile and checks if the section we want to access, exists.

Parameters

self	The configuration for which the get the status for.
section	The section we want the configuration to read from

Returns

: The status of this operation

Load keyfile if it is nog loaded yet

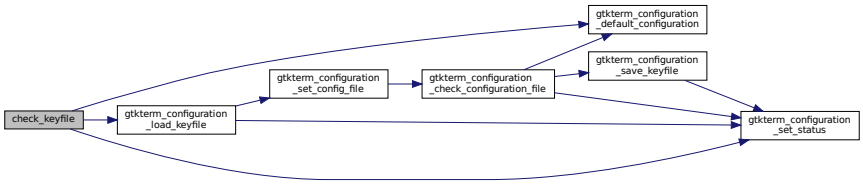
Check if the [section] exists in the key file.

we did not find the section so we create it

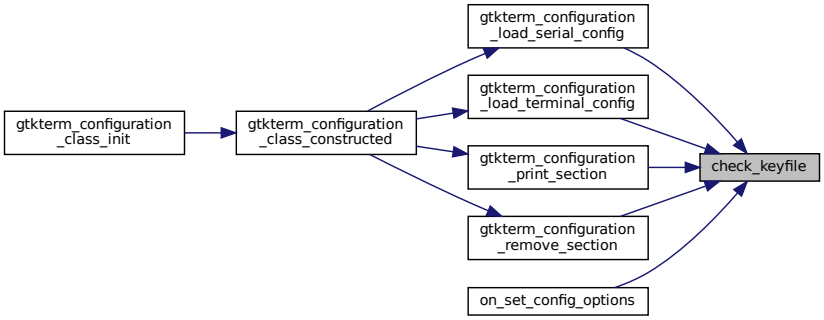
Set the dirty flag

Referenced by gtkterm_configuration_load_serial_config(), gtkterm_configuration_load_terminal_config(), gtkterm_configuration_print_section(), gtkterm_configuration_remove_section(), and on_set_config_options().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.2 gtkterm_configuration_check_configuration_file()

```
static GtkTermConfigurationState gtkterm_configuration_check_configuration_file (
            GtkTermConfiguration * self )  [static]
```

Check if the configuration file exists on disk.

If not it creates and new default one and save it to disk.

Parameters

self	The configuration class.

Returns

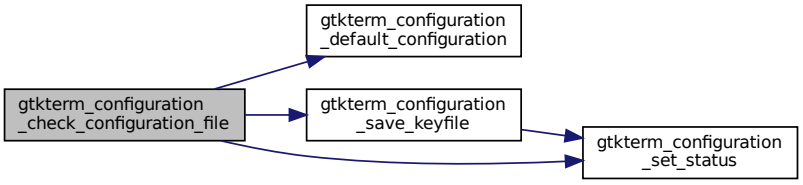
The result of the operation

is configuration file present if not, create it, with the [default] section

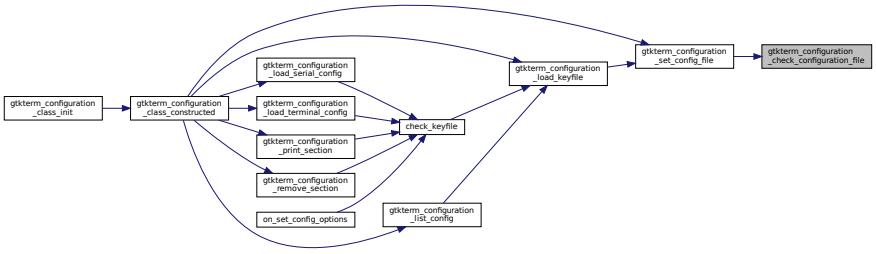
And save the keyfile

Referenced by gtkterm_configuration_set_config_file().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.3 gtkterm_configuration_class_constructed()

```
static void gtkterm_configuration_class_constructed (
            GObject * object )  [static]
```

Connect callback functions to signals.

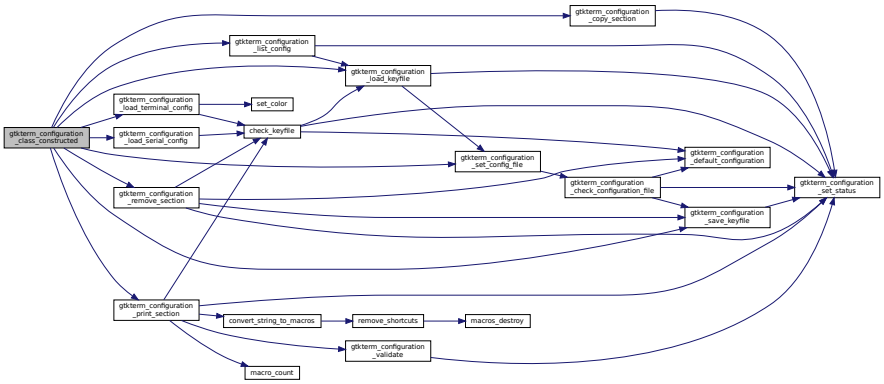
The callback functions performs the operation on the keyfile or configuration.

Parameters

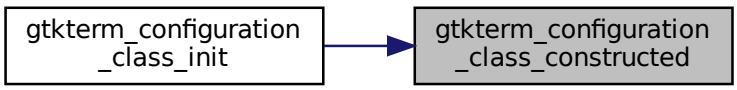
object	The configuration class object.

Referenced by gtkterm_configuration_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.4 gtkterm_configuration_class_init()

```
static void gtkterm_configuration_class_init (
            GtkTermConfigurationClass * class )  [static]
```

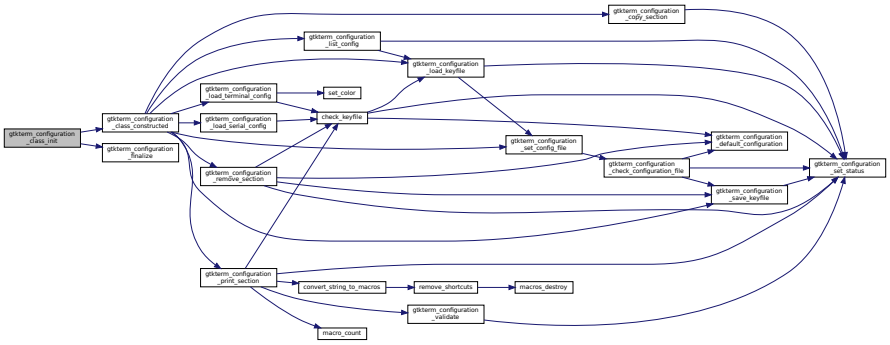
Initialize the class functions.

Nothing special to do here.

Parameters

class	The configuration.

Here is the call graph for this function:



6.11.1.5 gtkterm_configuration_copy_section()

```
static int gtkterm_configuration_copy_section (
            GtkTermConfiguration * self,
            gpointer sect,
            gpointer pc,
            gpointer tc,
            gpointer user_data )  [static]
```

Copy the active configuration into [section] of the Key file.

The pc and tc are the config items from a terminal window. They stay in memory until an explicite save is given.

Parameters

self	The configuration class.
sect	Section we want to copy the config into.
pc	The port configuration.
tc	Terminal configuration.
user_data	Not used.

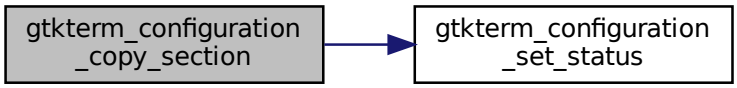
Returns

The result of the operation.

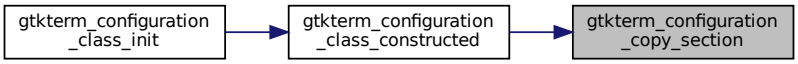
Macros are an array of strings, so we have to convert it All macros ends up in the string_list

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.6 gtkterm_configuration_default_configuration()

```
void gtkterm_configuration_default_configuration (
            GtkTermConfiguration * self,
            char * section )
```

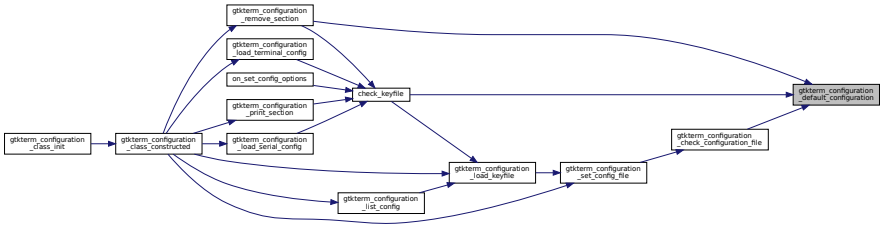
Create a new configuration with defaults.

Parameters

self	The configuration class.
section	The section we want to get the config for.

Referenced by check_keyfile(), gtkterm_configuration_check_configuration_file(), and gtkterm_configuration_remove_section().

Here is the caller graph for this function:



6.11.1.7 gtkterm_configuration_finalize()

```
static void gtkterm_configuration_finalize (
            GObject * object )  [static]
```

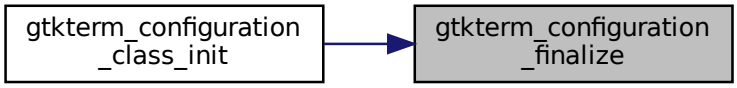
Remote all pointers when removing the object.

Parameters

object	The pointer to the configuration object.

Referenced by gtkterm_configuration_class_init().

Here is the caller graph for this function:



6.11.1.8 gtkterm_configuration_get_error()

```
GError * gtkterm_configuration_get_error (
            GtkTermConfiguration * self )
```

Return the latest error for the file operation.

Parameters

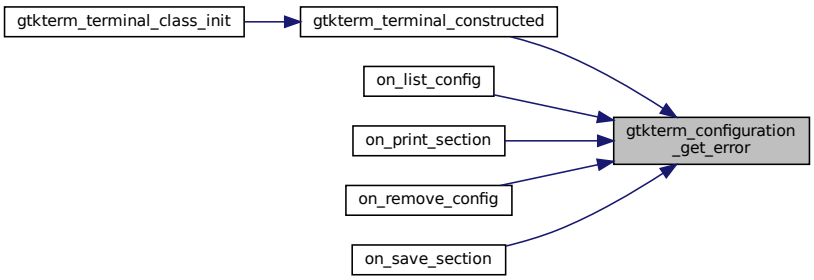
self	The configuration for which the get the status for.

Returns

The latest error.

Referenced by gtkterm_terminal_constructed(), on_list_config(), on_print_section(), on_remove_config(), and on_save_section().

Here is the caller graph for this function:



6.11.1.9 gtkterm_configuration_get_status()

```
GtkTermConfigurationState gtkterm_configuration_get_status (
            GtkTermConfiguration * self )
```

Return the latest status condiation for the file operation.

Parameters

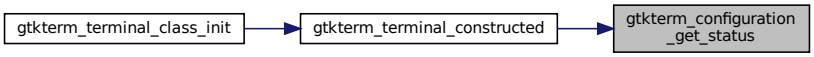
self	The configuration for which the get the status for.

Returns

The latest status.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.11.1.10 gtkterm_configuration_init()

```
static void gtkterm_configuration_init (
            GtkTermConfiguration * self )  [static]
```

Initialize the class members.

Parameters

self	the configuration.

Initialize to NULL so we can detect if it is loaded.

6.11.1.11 gtkterm_configuration_list_config()

```
static int gtkterm_configuration_list_config (
            GtkTermConfiguration * self,
            gpointer user_data )  [static]
```

Lists all sections in the keyfile.

Parameters

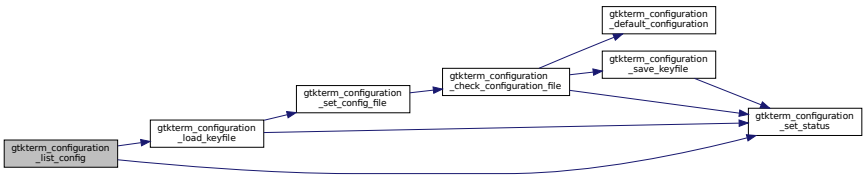
self	The configuration class.
user_data	Not used.

Returns

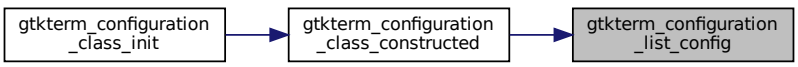
The result of the operation.

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.12 gtkterm_configuration_load_keyfile()

```
static int gtkterm_configuration_load_keyfile (
            GtkTermConfiguration * self,
            gpointer user_data )  [static]
```

Callback functions for signals.

Load the key file into memory.

The keyfile with all sections are loaded into memory. It is in raw format. All on/off etc are not translated yet.

Parameters

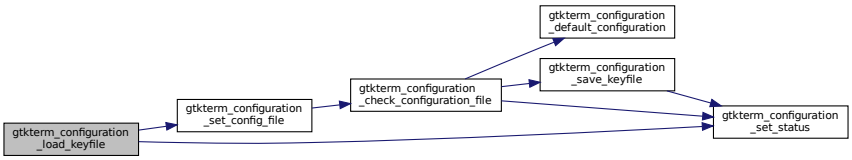
self	The configuration class.
user_data	Not used.

Returns

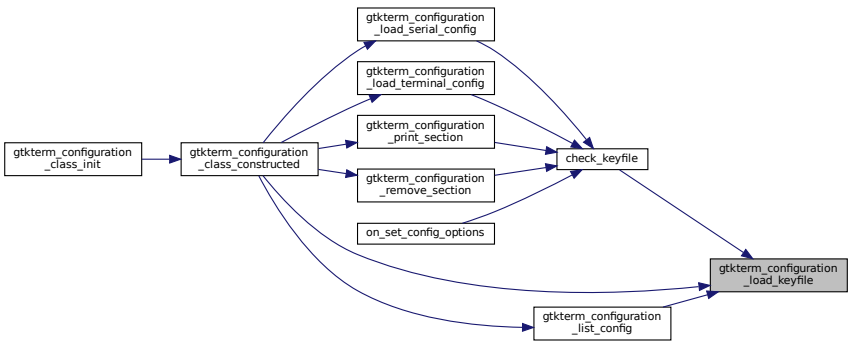
The result of the operation.

Referenced by check_keyfile(), gtkterm_configuration_class_constructed(), and gtkterm_configuration_list_config().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.13 gtkterm_configuration_load_serial_config()

```
static port_config_t * gtkterm_configuration_load_serial_config (
            GtkTermConfiguration * self,
            gpointer data,
            gpointer user_data )  [static]
```

Load the portconfiguration from keyfile.

Load the port configuration from [section] into the term config. If it does not exists it creates one from the defaults

Parameters

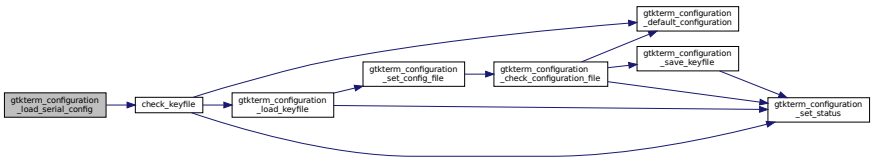
self	The configuration class.
data	The section we want to get the config from.
user_data	Not used.

Returns

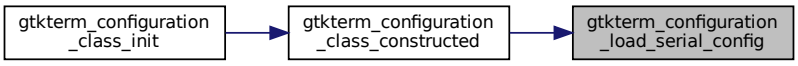
The port configuration which ends up as the last param in the signal. NULL on error.

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.14 gtkterm_configuration_load_terminal_config()

```
static term_config_t * gtkterm_configuration_load_terminal_config (
            GtkTermConfiguration * self,
            gpointer data,
            gpointer user_data )  [static]
```

Load the terminal configuration from keyfile.

Load the terminal configuration from [section] into the term config. If it does not exists it creates one from the defaults

Parameters

self	The configuration class.
data	The section we want to get the config from.
user_data	Not used.

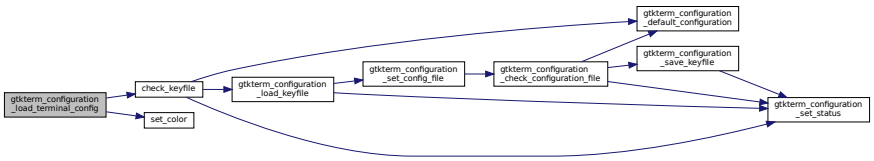
Returns

The terminal configuration which ends up as the last param in the signal. NULL on error.

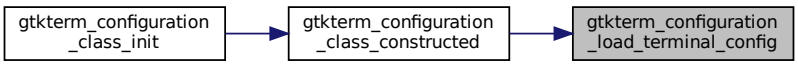
The Font is a Pango structure. This only can be added to a terminal So we have to convert it.

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.15 gtkterm_configuration_print_section()

```
static int gtkterm_configuration_print_section (
            GtkTermConfiguration * self,
            gpointer data,
            gpointer user_data )  [static]
```

Print the section to CLI.

Parameters

self	The configuration class.
data	Pointer to the section we want to show
user_data	Not used.

Returns

The result of the operation.

Print the serial port items

Print the terminal items

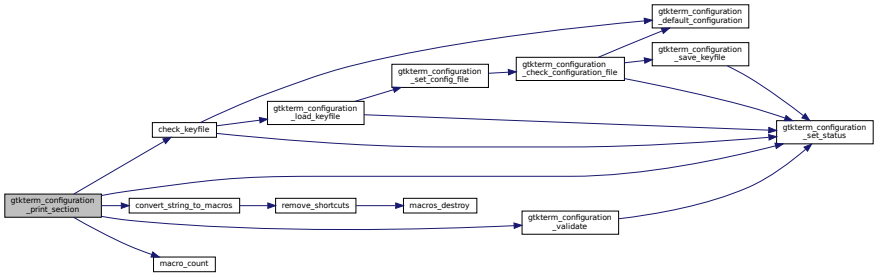
Convert the stringlist to macros. Existing shortcuts will be deleted from convert_string_to_macros

... and the macro's

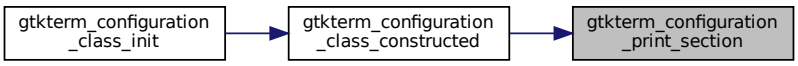
Inverse the return due to the handling return value of the callback

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.16 gtkterm_configuration_remove_section()

```
static int gtkterm_configuration_remove_section (
            GtkTermConfiguration * self,
            gpointer data,
            gpointer user_data )  [static]
```

Remove a section from the GKeyFile.

If it is the active section then switch back to default. If it is the default section then create a new 'default' default section

Parameters

self	The configuration class.
data	Pointer to the section we want to remove.
user_data	Not used.

Returns

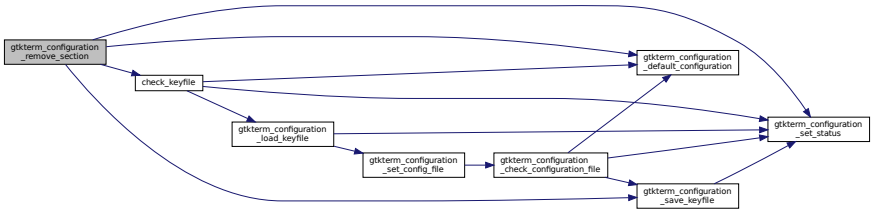
The result of the operation.

If we remove the DEFAULT_SECTION then create a new one

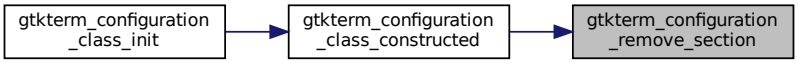
Remove the group from GKeyFile

Referenced by gtkterm_configuration_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.17 gtkterm_configuration_save_keyfile()

```
static int gtkterm_configuration_save_keyfile (
            GtkTermConfiguration * self,
            gpointer user_data )  [static]
```

Save the in momeory keyfile to file).

The keyfile with all sections saved to file

Parameters

self	The configuration class.
user_data	Not used.

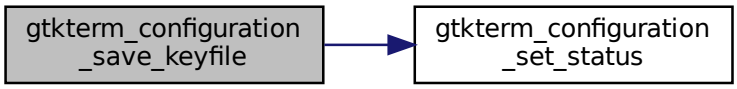
Returns

The result of the operation.

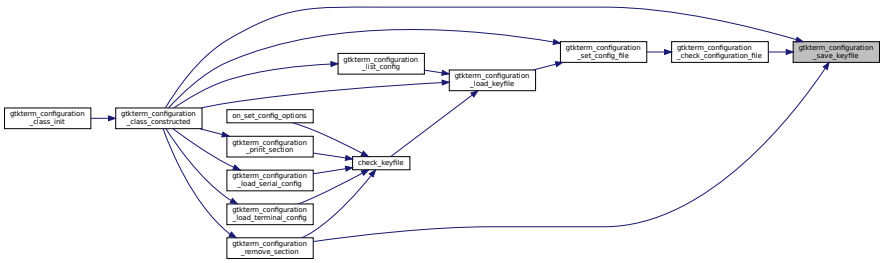
Reset the dirty flag now we saved the keyfile

Referenced by gtkterm_configuration_check_configuration_file(), gtkterm_configuration_class_constructed(), and gtkterm_configuration_remove_section().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.18 gtkterm_configuration_set_config_file()

```
static int gtkterm_configuration_set_config_file (
            GtkTermConfiguration * self,
            gpointer user_data )  [static]
```

Check if the file exists in the old/new location.

Old location of configuration file was $HOME/.gtktermrc. New location is $XDG_CONFIG_HOME/.gtktermrc. If configuration file exists at new location, use that one. Otherwise, if file exists at old location, move file to new location.

Version 2.0: Because we have to use gtkterm_conv, the file is always at the user directory. So we can skip eventually moving the file.

Parameters

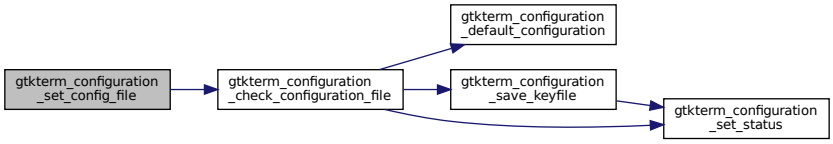
self	The configuration class.
user_data	Not used.

Returns

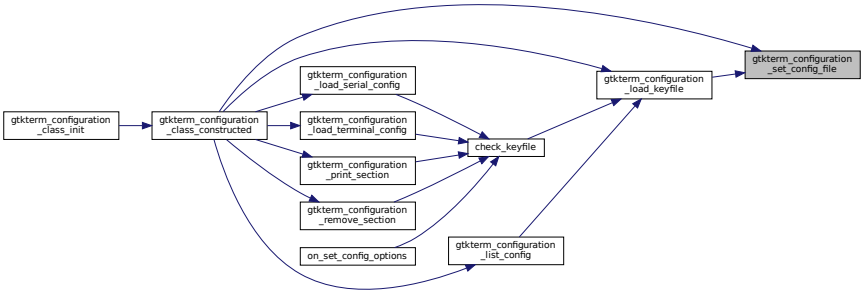
The result of the operation.

Referenced by gtkterm_configuration_class_constructed(), and gtkterm_configuration_load_keyfile().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.19 gtkterm_configuration_set_status()

```
GtkTermConfigurationState gtkterm_configuration_set_status (
            GtkTermConfiguration * self,
            GtkTermConfigurationState status,
            GError * error )
```

Sets the status and error of the last operation.

Parameters

self	The configuration for which the get the status for.
status	The status to be set.
error	The error message (can be NULL)

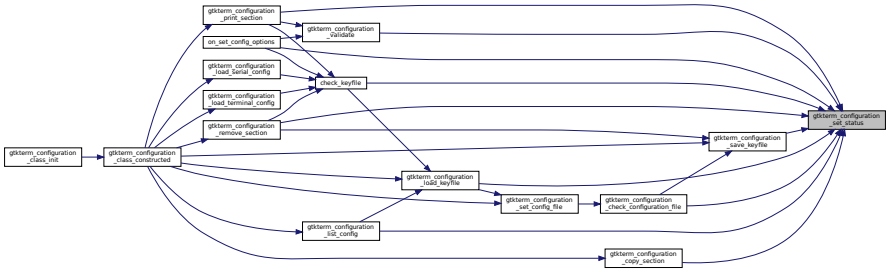
Returns

The latest status.

If there is a previous error, clear it

Referenced by check_keyfile(), gtkterm_configuration_check_configuration_file(), gtkterm_configuration_copy_section(), gtkterm_configuration_list_config(), gtkterm_configuration_load_keyfile(), gtkterm_configuration_print_section(), gtkterm_configuration_remove_section(), gtkterm_configuration_save_keyfile(), gtkterm_configuration_validate(), and on_set_config_options().

Here is the caller graph for this function:



6.11.1.20 gtkterm_configuration_validate()

```
GtkTermConfigurationState gtkterm_configuration_validate (
            GtkTermConfiguration * self,
            char * section )
```

validate the configuration, given by the section.

If not it creates and new default one and save it to disk. When it finds an invalid config option it returns with an error for which item the configuration check fails.

Parameters

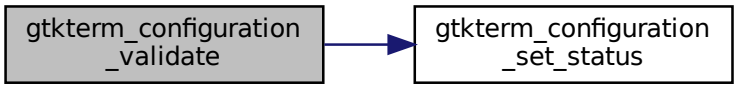
self	The configuration class.
section	The section we want to validate.

Returns

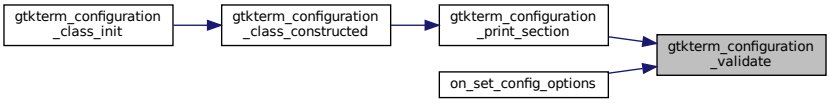
The result of the operation

Referenced by gtkterm_configuration_print_section(), and on_set_config_options().

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.1.21 on_set_config_options()

```
GtkTermConfigurationState on_set_config_options (
            const char * name,
            const char * value,
            gpointer data,
            GError ** error )
```

Set the config option in the keyfile.

All option which are given from the CLI are stored into the keyfile with [section] Options are not saved to disk.

Parameters

name	The configoption we want to set.
value	The value for this option.
data	The section we want to get the config from.
error	Error (not used).

Returns

The inversed (0 -> 1, 1 -> 0) result of the operation. Because of the handling of the return value from GOptionEntry. GOptionEntry contuinues if callback returns 1.

First check and load the keyfile

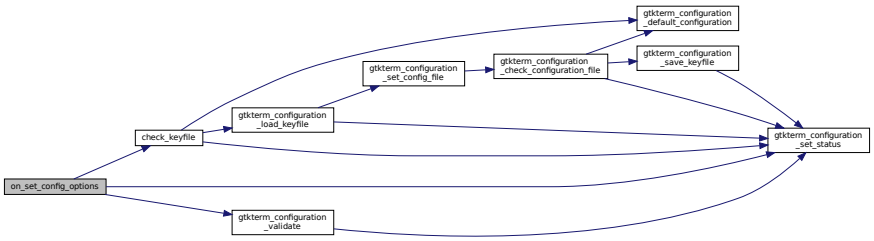
Check if we use the long or short option. For the long option, the option start at position 3 (–option). So add 2. For the short option the option start at positon 2 (-o) so add 1.

Search index for the option we want to set

Check for max path length. Exit if it is to long. Note: Serial port is also a path to a device.

We should not get here.

Set the dirty flagHere is the call graph for this function:



6.11.1.22 set_color()

```
static void set_color (
             GdkRGBA * color,
             float R,
             float G,
             float B,
             float A )  [static]
```

Functions for internal use only.

Convert the colors RGB to internal color scheme.

Parameters

color	The composed color
R	The red component
G	The green component
B	The blue component
A	Alpha Convert the colors RGB to internal color scheme

Referenced by gtkterm_configuration_load_terminal_config().

Here is the caller graph for this function:

6.11.2 Variable Documentation

6.11.2.1 GtkTermCLIShortOption

```
const char GtkTermCLIShortOption[][CONF_ITEM_LENGTH]
```

Referenced by on_set_config_options().

6.11.2.2 GtkTermConfigurationItems

```
const char GtkTermConfigurationItems[][CONF_ITEM_LENGTH]
```

Configuration options.

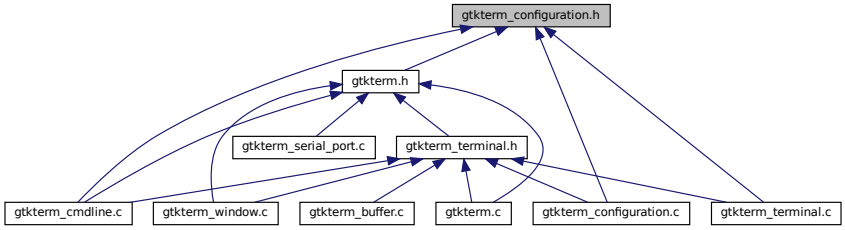
Used configuration options to hold consistency between load/save functions Also used as long-option when configuring by CLI

Todo Add the short option.

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_default_configuration(), gtkterm_configuration_load_seri
gtkterm_configuration_load_terminal_config(), gtkterm_configuration_print_section(), gtkterm_configuration_validate(),
and on_set_config_options().

6.12 gtkterm_configuration.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define GTKTERM_TYPE_CONFIGURATION gtkterm_configuration_get_type ()

Typedefs

- typedef struct _GtkTermConfiguration GtkTermConfiguration

Enumerations

- enum {
  CONF_ITEM_SERIAL_PORT , CONF_ITEM_SERIAL_BAUDRATE , CONF_ITEM_SERIAL_BITS ,
  CONF_ITEM_SERIAL_STOPBITS ,
  CONF_ITEM_SERIAL_PARITY , CONF_ITEM_SERIAL_FLOW_CONTROL , CONF_ITEM_TERM_WAIT_DELAY
  , CONF_ITEM_TERM_WAIT_CHAR ,
  CONF_ITEM_SERIAL_RS485_RTS_TIME_BEFORE_TX , CONF_ITEM_SERIAL_RS485_RTS_TIME_AFTER_TX
  , CONF_ITEM_TERM_MACROS , CONF_ITEM_TERM_RAW_FILENAME ,
  CONF_ITEM_TERM_ECHO , CONF_ITEM_TERM_AUTO_LF , CONF_ITEM_TERM_AUTO_CR ,
  CONF_ITEM_SERIAL_DISABLE_PORT_LOCK ,
  CONF_ITEM_TERM_FONT , CONF_ITEM_TERM_TIMESTAMP , CONF_ITEM_TERM_BLOCK_CURSOR
  , CONF_ITEM_TERM_SHOW_CURSOR ,
  CONF_ITEM_TERM_ROWS , CONF_ITEM_TERM_COLS , CONF_ITEM_TERM_SCROLLBACK ,
  CONF_ITEM_TERM_VISUAL_BELL ,
  CONF_ITEM_TERM_FOREGROUND_RED , CONF_ITEM_TERM_FOREGROUND_GREEN , CONF_ITEM_TERM_FOREGR
  , CONF_ITEM_TERM_FOREGROUND_ALPHA ,
  CONF_ITEM_TERM_BACKGROUND_RED , CONF_ITEM_TERM_BACKGROUND_GREEN , CONF_ITEM_TERM_BACKGR
  , CONF_ITEM_TERM_BACKGROUND_ALPHA ,
  CONF_ITEM_LAST }

  Enum items for configuration.
- enum GtkTermConfigurationState {
  GTKTERM_CONFIGURATION_SUCCESS , GTKTERM_CONFIGURATION_FILE_CREATED , GTKTERM_CONFIGURATION
  , GTKTERM_CONFIGURATION_FILE_SAVED ,
  GTKTERM_CONFIGURATION_FILE_NOT_SAVED , GTKTERM_CONFIGURATION_NO_KEYFILE_LOADED
  , GTKTERM_CONFIGURATION_SECTION_REMOVED , GTKTERM_CONFIGURATION_SECTION_NOT_REMOVED
  ,
  GTKTERM_CONFIGURATION_SECTION_UNKNOWN , GTKTERM_CONFIGURATION_INVALID_BAUDRATE
  , GTKTERM_CONFIGURATION_INVALID_BITS , GTKTERM_CONFIGURATION_INVALID_STOPBITS ,
  GTKTERM_CONFIGURATION_INVALID_DELAY , GTKTERM_CONFIGURATION_FILNAME_TO_LONG ,
  GTKTERM_CONFIGURATION_UNKNOWN_OPTION , GTKTERM_CONFIGURATION_LAST }

  Enum config_error id.

Functions

- GtkTermConfiguration * gtkterm_configuration_new (void)
- GtkTermConfigurationState on_set_config_options (const char *, const char *, gpointer, GError **)

  Set the config option in the keyfile.
- GtkTermConfigurationState gtkterm_configuration_get_status (GtkTermConfiguration *)

  Return the latest status condiation for the file operation.
- GError * gtkterm_configuration_get_error (GtkTermConfiguration *)

  Return the latest error for the file operation.

Variables

- const char GtkTermConfigurationItems [ ][CONF_ITEM_LENGTH]

  Configuration options.

6.12.1 Macro Definition Documentation

6.12.1.1 GTKTERM_TYPE_CONFIGURATION

```
#define GTKTERM_TYPE_CONFIGURATION gtkterm_configuration_get_type ()
```

6.12.2 Typedef Documentation

6.12.2.1 GtkTermConfiguration

```
typedef struct _GtkTermConfiguration GtkTermConfiguration
```

6.12.3 Enumeration Type Documentation

6.12.3.1 anonymous enum

```
anonymous enum
```

Enum items for configuration.

Define all configuration items which are used in the resource file. it is an index to ConfigurationItem. Configuration item names.

Enumerator

CONF_ITEM_SERIAL_PORT	
CONF_ITEM_SERIAL_BAUDRATE	
CONF_ITEM_SERIAL_BITS	
CONF_ITEM_SERIAL_STOPBITS	
CONF_ITEM_SERIAL_PARITY	
CONF_ITEM_SERIAL_FLOW_CONTROL	
CONF_ITEM_TERM_WAIT_DELAY	
CONF_ITEM_TERM_WAIT_CHAR	
CONF_ITEM_SERIAL_RS485_RTS_TIME_BEFORE_TX	
CONF_ITEM_SERIAL_RS485_RTS_TIME_AFTER_TX	

Enumerator

CONF_ITEM_TERM_MACROS	
CONF_ITEM_TERM_RAW_FILENAME	
CONF_ITEM_TERM_ECHO	
CONF_ITEM_TERM_AUTO_LF	
CONF_ITEM_TERM_AUTO_CR	
CONF_ITEM_SERIAL_DISABLE_PORT_LOCK	
CONF_ITEM_TERM_FONT	
CONF_ITEM_TERM_TIMESTAMP	
CONF_ITEM_TERM_BLOCK_CURSOR	
CONF_ITEM_TERM_SHOW_CURSOR	
CONF_ITEM_TERM_ROWS	
CONF_ITEM_TERM_COLS	
CONF_ITEM_TERM_SCROLLBACK	
CONF_ITEM_TERM_VISUAL_BELL	
CONF_ITEM_TERM_FOREGROUND_RED	
CONF_ITEM_TERM_FOREGROUND_GREEN	
CONF_ITEM_TERM_FOREGROUND_BLUE	
CONF_ITEM_TERM_FOREGROUND_ALPHA	
CONF_ITEM_TERM_BACKGROUND_RED	
CONF_ITEM_TERM_BACKGROUND_GREEN	
CONF_ITEM_TERM_BACKGROUND_BLUE	
CONF_ITEM_TERM_BACKGROUND_ALPHA	
CONF_ITEM_LAST	Checking as last item in the list.

6.12.3.2 GtkTermConfigurationState

```
enum GtkTermConfigurationState
```

Enum config_error id.

Many of the gtk_configuration functions return an error id.

Enumerator

GTKTERM_CONFIGURATION_SUCCESS	
GTKTERM_CONFIGURATION_FILE_CREATED	
GTKTERM_CONFIGURATION_FILE_CONFIG_LOAD	
GTKTERM_CONFIGURATION_FILE_SAVED	
GTKTERM_CONFIGURATION_FILE_NOT_SAVED	
GTKTERM_CONFIGURATION_NO_KEYFILE_LOADED	
GTKTERM_CONFIGURATION_SECTION_REMOVED	
GTKTERM_CONFIGURATION_SECTION_NOT_REMOVED	
GTKTERM_CONFIGURATION_SECTION_UNKNOWN	
GTKTERM_CONFIGURATION_INVALID_BAUDRATE	
GTKTERM_CONFIGURATION_INVALID_BITS	
GTKTERM_CONFIGURATION_INVALID_STOPBITS	

Enumerator

GTKTERM_CONFIGURATION_INVALID_DELAY	
GTKTERM_CONFIGURATION_FILNAME_TO_LONG	
GTKTERM_CONFIGURATION_UNKNOWN_OPTION	
GTKTERM_CONFIGURATION_LAST	

6.12.4 Function Documentation

6.12.4.1 gtkterm_configuration_get_error()

```
GError * gtkterm_configuration_get_error (
            GtkTermConfiguration * self )
```

Return the latest error for the file operation.

Parameters

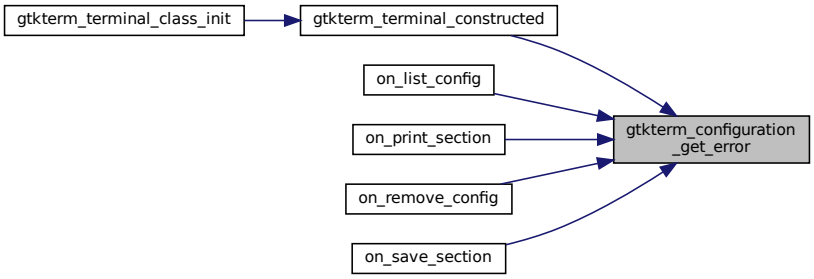
self	The configuration for which the get the status for.

Returns

The latest error.

Referenced by gtkterm_terminal_constructed(), on_list_config(), on_print_section(), on_remove_config(), and on_save_section().

Here is the caller graph for this function:



6.12.4.2 gtkterm_configuration_get_status()

```
GtkTermConfigurationState gtkterm_configuration_get_status (
            GtkTermConfiguration * self )
```

Return the latest status condiation for the file operation.

Parameters

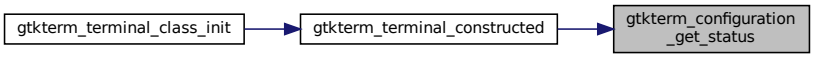
self	The configuration for which the get the status for.

Returns

The latest status.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.12.4.3 gtkterm_configuration_new()

```
GtkTermConfiguration * gtkterm_configuration_new (
            void  )
```

6.12.4.4 on_set_config_options()

```
GtkTermConfigurationState on_set_config_options (
            const char * name,
            const char * value,
            gpointer data,
            GError ** error )
```

Set the config option in the keyfile.

All option which are given from the CLI are stored into the keyfile with [section] Options are not saved to disk.

Parameters

name	The configoption we want to set.
value	The value for this option.
data	The section we want to get the config from.
error	Error (not used).

Returns

The inversed (0 -> 1, 1 -> 0) result of the operation. Because of the handling of the return value from GOptionEntry. GOptionEntry contuinues if callback returns 1.

First check and load the keyfile

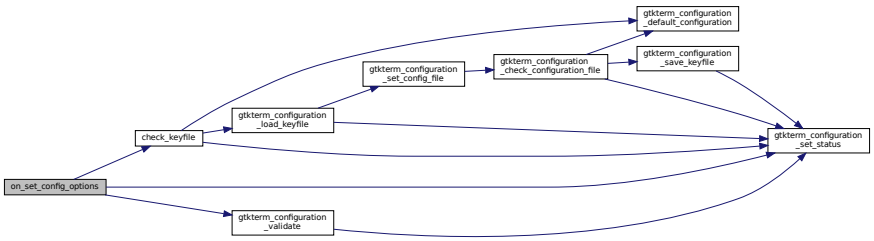
Check if we use the long or short option. For the long option, the option start at position 3 (–option). So add 2. For the short option the option start at positon 2 (-o) so add 1.

Search index for the option we want to set

Check for max path length. Exit if it is to long. Note: Serial port is also a path to a device.

We should not get here.

Set the dirty flagHere is the call graph for this function:



6.12.5 Variable Documentation

6.12.5.1 GtkTermConfigurationItems

```
const char GtkTermConfigurationItems[][CONF_ITEM_LENGTH]  [extern]
```

Configuration options.

Used configuration options to hold consistency between load/save functions Also used as long-option when configuring by CLI

Todo Add the short option.

Referenced by gtkterm_configuration_copy_section(), gtkterm_configuration_default_configuration(), gtkterm_configuration_load_seri
gtkterm_configuration_load_terminal_config(), gtkterm_configuration_print_section(), gtkterm_configuration_validate(),
and on_set_config_options().

6.13 gtkterm_configuration.h

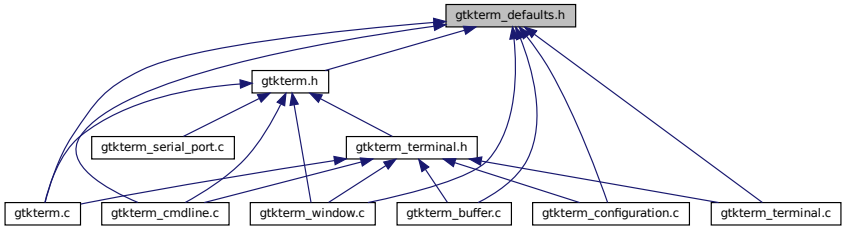
Go to the documentation of this file.

```
1 /*************************************************************************
2 * gtkterm_configuration.h
3 *
4 * GTKTerm Software (c) Julien Schmitt
5 *
6 *
7 * @brief Purpose
8 *      Load and save configuration file
9 *      - Header file
10 *
11 *************************************************************************/
12
13 #ifndef GTKTERM_CONFIGURATION_H
14 #define GTKTERM_CONFIGURATION_H
15
23 enum {
24         CONF_ITEM_SERIAL_PORT,
25         CONF_ITEM_SERIAL_BAUDRATE,
26         CONF_ITEM_SERIAL_BITS,
27         CONF_ITEM_SERIAL_STOPBITS,
28         CONF_ITEM_SERIAL_PARITY,
29         CONF_ITEM_SERIAL_FLOW_CONTROL,
30         CONF_ITEM_TERM_WAIT_DELAY,
31         CONF_ITEM_TERM_WAIT_CHAR,
32         CONF_ITEM_SERIAL_RS485_RTS_TIME_BEFORE_TX,
33         CONF_ITEM_SERIAL_RS485_RTS_TIME_AFTER_TX,
34         CONF_ITEM_TERM_MACROS,
35         CONF_ITEM_TERM_RAW_FILENAME,
36         CONF_ITEM_TERM_ECHO,
37         CONF_ITEM_TERM_AUTO_LF,
38         CONF_ITEM_TERM_AUTO_CR,
39         CONF_ITEM_SERIAL_DISABLE_PORT_LOCK,
40         CONF_ITEM_TERM_FONT,
41         CONF_ITEM_TERM_TIMESTAMP,
42         CONF_ITEM_TERM_BLOCK_CURSOR,
43         CONF_ITEM_TERM_SHOW_CURSOR,
44         CONF_ITEM_TERM_ROWS,
45         CONF_ITEM_TERM_COLS,
46         CONF_ITEM_TERM_SCROLLBACK,
47         CONF_ITEM_TERM_VISUAL_BELL,
48         CONF_ITEM_TERM_FOREGROUND_RED,
49         CONF_ITEM_TERM_FOREGROUND_GREEN,
50         CONF_ITEM_TERM_FOREGROUND_BLUE,
51         CONF_ITEM_TERM_FOREGROUND_ALPHA,
52         CONF_ITEM_TERM_BACKGROUND_RED,
53         CONF_ITEM_TERM_BACKGROUND_GREEN,
54         CONF_ITEM_TERM_BACKGROUND_BLUE,
55         CONF_ITEM_TERM_BACKGROUND_ALPHA,
56         CONF_ITEM_LAST
57 };
58
65 typedef enum {
66     GTKTERM_CONFIGURATION_SUCCESS,
67     GTKTERM_CONFIGURATION_FILE_CREATED,
68     GTKTERM_CONFIGURATION_FILE_CONFIG_LOAD,
69     GTKTERM_CONFIGURATION_FILE_SAVED,
70     GTKTERM_CONFIGURATION_FILE_NOT_SAVED,
71     GTKTERM_CONFIGURATION_NO_KEYFILE_LOADED,
72     GTKTERM_CONFIGURATION_SECTION_REMOVED,
73     GTKTERM_CONFIGURATION_SECTION_NOT_REMOVED,
74     GTKTERM_CONFIGURATION_SECTION_UNKNOWN,
75     GTKTERM_CONFIGURATION_INVALID_BAUDRATE,
76     GTKTERM_CONFIGURATION_INVALID_BITS,
77     GTKTERM_CONFIGURATION_INVALID_STOPBITS,
78     GTKTERM_CONFIGURATION_INVALID_DELAY,
79     GTKTERM_CONFIGURATION_FILNAME_TO_LONG,
80     GTKTERM_CONFIGURATION_UNKNOWN_OPTION,
81     GTKTERM_CONFIGURATION_LAST
82
83 } GtkTermConfigurationState;
84
85 extern const char GtkTermConfigurationItems [][CONF_ITEM_LENGTH];
86
87 G_BEGIN_DECLS
88
89 #define GTKTERM_TYPE_CONFIGURATION gtkterm_configuration_get_type ()
90 G_DECLARE_FINAL_TYPE (GtkTermConfiguration, gtkterm_configuration, GTKTERM, CONFIGURATION, GObject)
91 typedef struct _GtkTermConfiguration GtkTermConfiguration;
92
93 GtkTermConfiguration *gtkterm_configuration_new (void);
94
95 GtkTermConfigurationState on_set_config_options (const char *, const char *, gpointer,  GError **);
```

```
96 GtkTermConfigurationState gtkterm_configuration_get_status (GtkTermConfiguration *);
97 GError *gtkterm_configuration_get_error (GtkTermConfiguration *);
98
99 G_END_DECLS
100
101 #endif //GTKTERM_CONFIGURATION
```

6.14 gtkterm_defaults.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define DEFAULT_FONT "Monospace 12"
- #define DEFAULT_SCROLLBACK 10000
- #define DEFAULT_DELAY 0
- #define DEFAULT_CHAR -1
- #define DEFAULT_DELAY_RS485 30
- #define DEFAULT_ECHO "false"
- #define DEFAULT_VISUAL_BELL "false"
- #define DEFAULT_PORT "/dev/ttyS0"
- #define DEFAULT_BAUDRATE 115200
- #define DEFAULT_PARITY "none"
- #define DEFAULT_BITS 8
- #define DEFAULT_STOPBITS 1
- #define DEFAULT_FLOW "none"
- #define GTKTERM_SERIAL_PORT_RECEIVE_BUFFER_SIZE 8192
- #define GTKTERM_SERIAL_PORT_TRANSMIT_BUFFER_SIZE 4096
- #define LINE_FEED 0x0A
- #define POLL_DELAY 100
- #define BUFFER_LENGTH 256
- #define MAX_SECTION_LENGTH 32
- #define GTKTERM_MESSAGE_LENGTH 128
- #define DEFAULT_SECTION "default"
- #define CONFIGURATION_FILENAME ".gtktermrc"
- #define CONF_ITEM_LENGTH 32
- #define DEFAULT_STRING_LEN 32
- #define ASCII_VIEW 0
- #define HEXADECIMAL_VIEW 1
- #define BUFFER_SIZE (128 * 1024)

6.14.1 Macro Definition Documentation

6.14.1.1 ASCII_VIEW

```
#define ASCII_VIEW 0
```

Type of terminal view

6.14.1.2 BUFFER_LENGTH

```
#define BUFFER_LENGTH 256
```

Generic defaults

6.14.1.3 BUFFER_SIZE

```
#define BUFFER_SIZE (128 * 1024)
```

Size of the buffer between the terminal and port

6.14.1.4 CONF_ITEM_LENGTH

```
#define CONF_ITEM_LENGTH 32
```

6.14.1.5 CONFIGURATION_FILENAME

```
#define CONFIGURATION_FILENAME ".gtktermrc"
```

Name of the resource file

6.14.1.6 DEFAULT_BAUDRATE

```
#define DEFAULT_BAUDRATE 115200
```

6.14.1.7 DEFAULT_BITS

```
#define DEFAULT_BITS 8
```

6.14.1.8 DEFAULT_CHAR

```
#define DEFAULT_CHAR -1
```

6.14.1.9 DEFAULT_DELAY

```
#define DEFAULT_DELAY 0
```

6.14.1.10 DEFAULT_DELAY_RS485

```
#define DEFAULT_DELAY_RS485 30
```

6.14.1.11 DEFAULT_ECHO

```
#define DEFAULT_ECHO "false"
```

6.14.1.12 DEFAULT_FLOW

```
#define DEFAULT_FLOW "none"
```

6.14.1.13 DEFAULT_FONT

```
#define DEFAULT_FONT "Monospace 12"
```

Defaults for VTE-terminal

6.14.1.14 DEFAULT_PARITY

```
#define DEFAULT_PARITY "none"
```

6.14.1.15 DEFAULT_PORT

```
#define DEFAULT_PORT "/dev/ttyS0"
```

Defaults for serial ports

6.14.1.16 DEFAULT_SCROLLBACK

```
#define DEFAULT_SCROLLBACK 10000
```

6.14.1.17 DEFAULT_SECTION

```
#define DEFAULT_SECTION "default"
```

Default section if not specified

6.14.1.18 DEFAULT_STOPBITS

```
#define DEFAULT_STOPBITS 1
```

6.14.1.19 DEFAULT_STRING_LEN

```
#define DEFAULT_STRING_LEN 32
```

6.14.1.20 DEFAULT_VISUAL_BELL

```
#define DEFAULT_VISUAL_BELL "false"
```

6.14.1.21 GTKTERM_MESSAGE_LENGTH

```
#define GTKTERM_MESSAGE_LENGTH 128
```

6.14.1.22 GTKTERM_SERIAL_PORT_RECEIVE_BUFFER_SIZE

```
#define GTKTERM_SERIAL_PORT_RECEIVE_BUFFER_SIZE 8192
```

The buffers for receive and transmit are internal buffers for the communication API. It is not the buffersize for the terminal/serialport communication Size of the receive buffer for the serial port

6.14.1.23 GTKTERM_SERIAL_PORT_TRANSMIT_BUFFER_SIZE

```
#define GTKTERM_SERIAL_PORT_TRANSMIT_BUFFER_SIZE 4096
```

Size of the transmit buuffer for the serial port

6.14.1.24 HEXADECIMAL_VIEW

```
#define HEXADECIMAL_VIEW 1
```

6.14.1.25 LINE_FEED

```
#define LINE_FEED 0x0A
```

6.14.1.26 MAX_SECTION_LENGTH

```
#define MAX_SECTION_LENGTH 32
```

6.14.1.27 POLL_DELAY

```
#define POLL_DELAY 100
```

in ms (for control signals)

6.15 gtkterm_defaults.h

Go to the documentation of this file.

```
1 #ifndef GTKTERM_DEFAULTS_H
2 #define GTKTERM_DEFAULTS_H
3
5 #define DEFAULT_FONT            "Monospace 12"
6 #define DEFAULT_SCROLLBACK      10000
7 #define DEFAULT_DELAY           0
8 #define DEFAULT_CHAR            -1
9 #define DEFAULT_DELAY_RS485     30
10 #define DEFAULT_ECHO            "false"
11 #define DEFAULT_VISUAL_BELL     "false"
12
14 #define DEFAULT_PORT            "/dev/ttyS0"
15 #define DEFAULT_BAUDRATE        115200
16 #define DEFAULT_PARITY          "none"
17 #define DEFAULT_BITS            8
18 #define DEFAULT_STOPBITS        1
19 #define DEFAULT_FLOW            "none"
20
25 #define GTKTERM_SERIAL_PORT_RECEIVE_BUFFER_SIZE  8192
26 #define GTKTERM_SERIAL_PORT_TRANSMIT_BUFFER_SIZE 4096
28 #define LINE_FEED               0x0A
29 #define POLL_DELAY              100
32 #define BUFFER_LENGTH           256
33 #define MAX_SECTION_LENGTH      32
34 #define GTKTERM_MESSAGE_LENGTH  128
35 #define DEFAULT_SECTION        "default"
36 #define CONFIGURATION_FILENAME ".gtktermrc"
37 #define CONF_ITEM_LENGTH        32
38 #define DEFAULT_STRING_LEN      32
39
41 #define ASCII_VIEW              0
42 #define HEXADECIMAL_VIEW        1
43
44 #define BUFFER_SIZE             (128 * 1024)
46 #endif // GTKTERM_DEFAULTS_H
```

6.16 gtkterm_serial_port.c File Reference

```
#include <gtk/gtk.h>
#include <glib.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <pwd.h>
#include <termios.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdbool.h>
#include <unistd.h>
#include <config.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include <glib-unix.h>
#include <gio/gio.h>
#include <gio-unix-2.0/gio/gunixinputstream.h>
#include <gio-unix-2.0/gio/gunixoutputstream.h>
#include <gudev/gudev.h>
#include "gtkterm.h"
```

```
#include "gtkterm_serial_port.h"
```

Include dependency graph for gtkterm_serial_port.c:

Classes

- struct GtkTermSerialPortPrivate
- struct _GtkTermSerialPort
- struct _GtkTermSerialPortClass

Macros

- #define GTKTERM_SERIAL_PORT_CONTROL_POLL_DELAY 100

Enumerations

- enum {
  PROP_0 , PROP_PORT_CONFIG , PROP_PORT_STATUS , PROP_PORT_SIGNALS ,
  N_PROPS }

Functions

- static int gtkterm_serial_port_open (GtkTermSerialPort ∗self)

  Connects to the serial port.
- static int gtkterm_serial_port_close (GtkTermSerialPort ∗self)

  Closes the serial port.
- int gtkterm_serial_port_lock (GtkTermSerialPort ∗self, GError ∗∗error)

  Locks the serial port.
- int gtkterm_serial_port_unlock (GtkTermSerialPort ∗self)

  Unlocks the serial port.
- void gtkterm_serial_port_set_status (GtkTermSerialPort ∗self, GtkTermSerialPortState new_status, GError ∗error)

  Set the status of the serial port.
- static void gtkterm_serial_port_serial_data_received (GObject ∗source, GAsyncResult ∗res, gpointer user↩ _data)

  Callback where data is received from the input stream.
- static int gtkterm_serial_port_serial_data_transmit (GObject ∗object, gpointer data, unsigned int length, gpointer user_data)

  Callback for signal where data is transmitted to the output stream of the port.
- static int gtkterm_serial_port_control_signals_read (gpointer data)

  Reads the serial port signals (DTR, etc)
- void gtkterm_serial_port_set_signals (GtkTermSerialPort ∗self, unsigned int param)

  Set the signals for the Serial Port structure.
- static bool gtkterm_serial_port_handle_usr1 (gpointer user_data)

  Handles USR1 signal. It opens the port.
- static bool gtkterm_serial_port_handle_usr2 (gpointer user_data)

Handles USR2 signal. It closes the port.

- GtkTermSerialPort * gtkterm_serial_port_new (port_config_t *port_conf)

  Create a new serial port object.
- static void gtkterm_serial_port_set (GtkTermSerialPort *self, GtkTermSerialPortStatus status)

  Opens or closes the serial port.
- static void gtkterm_serial_port_handle (GtkTermSerialPort *self, const char *action)

  Based on the action return from uevent we handle to open of close the port.
- void gtkterm_serial_port_event_udev (GUdevClient *client, const char *action, GUdevDevice *device, gpointer user_data)

  Callback for the uevent signal.
- void gtkterm_serial_port_device_monitor (GtkTermSerialPort *self)
- static bool gtkterm_serial_port_config (GtkTermSerialPort *self, struct termios *termios_p, GError **error)
- char * gtkterm_serial_port_get_string (GtkTermSerialPort *self)

  Convert port config to a string.
- static void gtkterm_serial_port_class_constructed (GObject *object)

  Connect callback functions to signals.
- static void gtkterm_serial_port_get_property (GObject *object, unsigned int prop_id, GValue *value, GParamSpec *pspec)

  get the property of the GtkTermSerialPort structure
- static void gtkterm_serial_port_set_property (GObject *object, unsigned int prop_id, const GValue *value, GParamSpec *pspec)

  Set the property of the GtkTermSerialPort structure.
- static void gtkterm_serial_port_finalize (GObject *object)

  Remote all pointers when removing the object.
- static void gtkterm_serial_port_class_init (GtkTermSerialPortClass *class)

  Initializing the serial_port class.
- static void gtkterm_serial_port_init (GtkTermSerialPort *self)

  Initialize the serial with the config parameters.
- GtkTermSerialPortState gtkterm_serial_port_get_status (GtkTermSerialPort *self)

  Return the status of the serial port.
- GError * gtkterm_serial_port_get_error (GtkTermSerialPort *self)

  Return the last error which occured.
- unsigned int gtkterm_serial_port_get_signals (GtkTermSerialPort *self)

  Get the signals from the Serial Port structure.
- static int gtkterm_serial_port_read_signals (GtkTermSerialPort *self)

  Does the actual reading of the serial signals.

Variables

- const char GtkTermSerialPortStateString [ ][DEFAULT_STRING_LEN]
- static GParamSpec * gtkterm_serial_port_properties [N_PROPS] = {NULL}

6.16.1 Macro Definition Documentation

6.16.1.1 GTKTERM_SERIAL_PORT_CONTROL_POLL_DELAY

```
#define GTKTERM_SERIAL_PORT_CONTROL_POLL_DELAY 100
```

In milliseconds for control signals

6.16.2 Enumeration Type Documentation

6.16.2.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_PORT_CONFIG	
PROP_PORT_STATUS	
PROP_PORT_SIGNALS	
N_PROPS	

6.16.3 Function Documentation

6.16.3.1 gtkterm_serial_port_class_constructed()

```
static void gtkterm_serial_port_class_constructed (
            GObject * object )  [static]
```

Connect callback functions to signals.

The callback functions performs the operation on the keyfile or configuration.

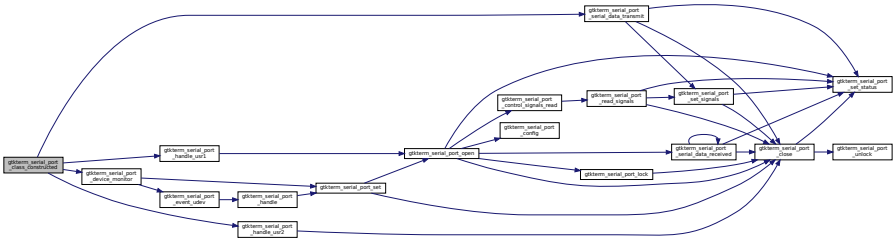
Parameters

object	The configuration class object.

Install the user signals

Referenced by gtkterm_serial_port_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.16.3.2 gtkterm_serial_port_class_init()

```
static void gtkterm_serial_port_class_init (
            GtkTermSerialPortClass * class )  [static]
```

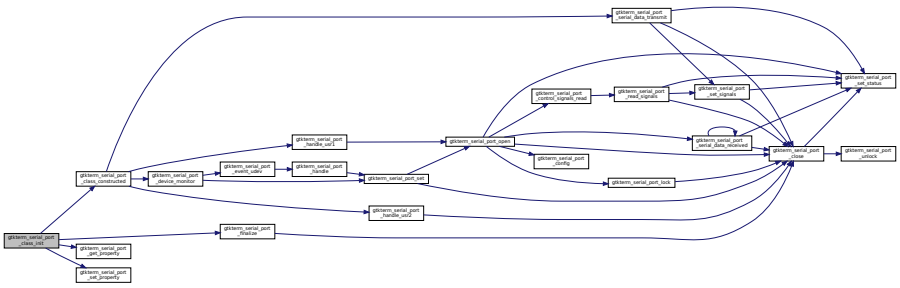
Initializing the serial_port class.

Setting the properties and callback functions

Parameters

class	The serial_portclass

Parameters to hand over at creation of the objectHere is the call graph for this function:



6.16.3.3 gtkterm_serial_port_close()

```
static int gtkterm_serial_port_close (
            GtkTermSerialPort * self )  [static]
```

Closes the serial port.

It check is there is an open port and if yes it closes the port. Port will also be unlocked (if locked).

Parameters

self	The Serial Port structure.

Returns

int Result of the operation

Set the saved termios back to the port

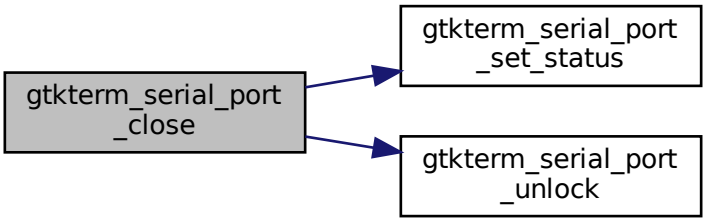
Flush input and output data

Unlock the port

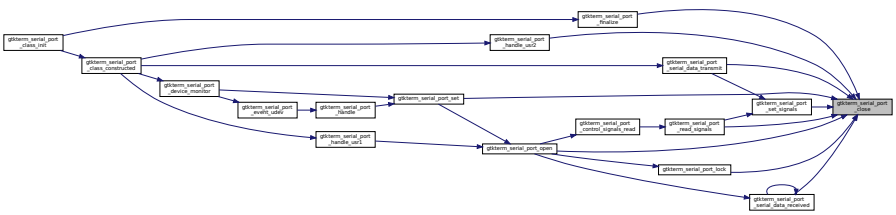
Close streams and port

Referenced by gtkterm_serial_port_finalize(), gtkterm_serial_port_handle_usr2(), gtkterm_serial_port_lock(), gtkterm_serial_port_open(), gtkterm_serial_port_read_signals(), gtkterm_serial_port_serial_data_received(), gtkterm_serial_port_serial_data_transmit(), gtkterm_serial_port_set(), and gtkterm_serial_port_set_signals().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.4 gtkterm_serial_port_config()

```
static bool gtkterm_serial_port_config (
            GtkTermSerialPort * self,
            struct termios * termios_p,
            GError ** error )  [static]
```

set control / enable receiver

ignore break and framing errors

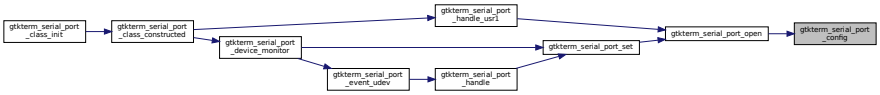
clear output and local mode

< Timeout in deciseconds for noncanonical read

< Minimal characters for noncanonical read

Referenced by gtkterm_serial_port_open().

Here is the caller graph for this function:



6.16.3.5 gtkterm_serial_port_control_signals_read()

```
static int gtkterm_serial_port_control_signals_read (
            gpointer data )  [static]
```

Reads the serial port signals (DTR, etc)

Parameters

data	The serial port to read the signals for.

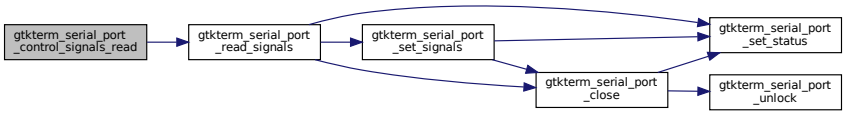
Returns

int The result of readings.

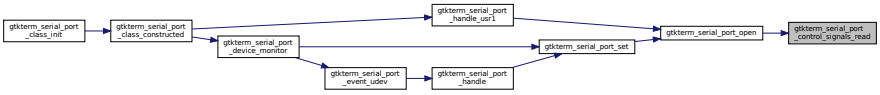
If the signals are changed then notify the terminal so the statusbar can be updated

Referenced by gtkterm_serial_port_open().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.6 gtkterm_serial_port_device_monitor()

```
void gtkterm_serial_port_device_monitor (
            GtkTermSerialPort * self )
```

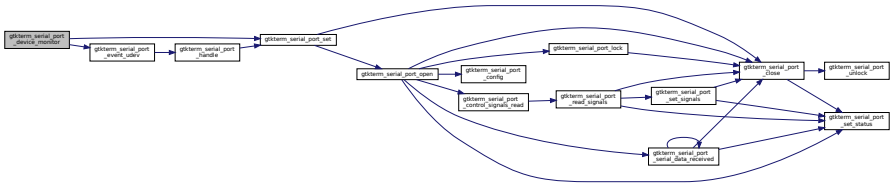
Create the udev client

Get the initial status of the device

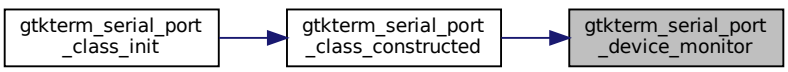
Monitor udev devices We add self as parameter so we can access the port configuration if needed.

Referenced by gtkterm_serial_port_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.7 gtkterm_serial_port_event_udev()

```
void gtkterm_serial_port_event_udev (
            GUdevClient * client,
            const char * action,
            GUdevDevice * device,
            gpointer user_data )
```

Callback for the uevent signal.

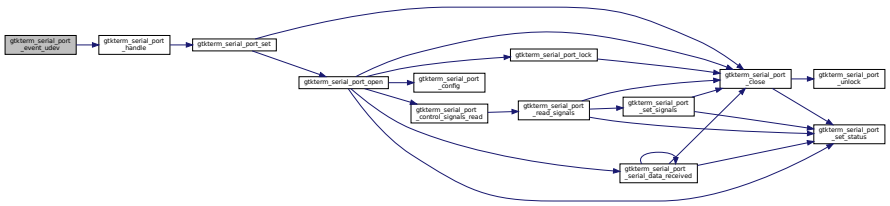
Parameters

client	The udev-client
action	The action it detects
device	The device.
user_data	The GtkTermSerialPort structure.

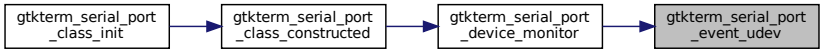
Check if the uevent device file matches the port of this configuration

Referenced by gtkterm_serial_port_device_monitor().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.8 gtkterm_serial_port_finalize()

```
static void gtkterm_serial_port_finalize (
            GObject * object )  [static]
```

Remote all pointers when removing the object.

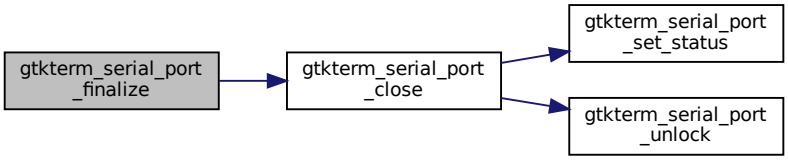
Parameters

object	The pointer to the serial port object.

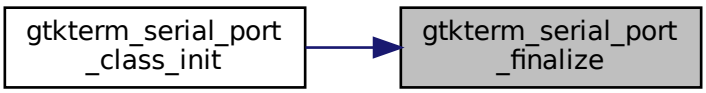
Close port, clear error

Referenced by gtkterm_serial_port_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.9 gtkterm_serial_port_get_error()

```
GError * gtkterm_serial_port_get_error (
            GtkTermSerialPort * self )
```

Return the last error which occured.

Parameters

self	The serial port

Returns

GERrror The pointer to the GError struct

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:

6.16.3.10 gtkterm_serial_port_get_property()

```
static void gtkterm_serial_port_get_property (
            GObject * object,
            unsigned int prop_id,
            GValue * value,
            GParamSpec * pspec )  [static]
```

get the property of the GtkTermSerialPort structure

This is used to update the properties. For now it is uses to update the notify.

Parameters

object	The object.
prop↩ _id	The id of the property to get.
value	The value for the property
pspec	Metadata for property setting.

Referenced by gtkterm_serial_port_class_init().

Here is the caller graph for this function:

6.16.3.11 gtkterm_serial_port_get_signals()

```
unsigned int gtkterm_serial_port_get_signals (
            GtkTermSerialPort * self )
```

Get the signals from the Serial Port structure.

This is also used for external reading.

Parameters

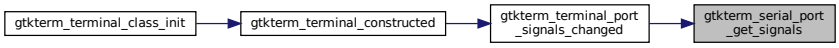
self	The serial port to read the signals for.

Returns

unsigned int The latest serial port signals

Referenced by gtkterm_terminal_port_signals_changed().

Here is the caller graph for this function:



6.16.3.12 gtkterm_serial_port_get_status()

```
GtkTermSerialPortState gtkterm_serial_port_get_status (
            GtkTermSerialPort * self )
```

Return the status of the serial port.

Parameters

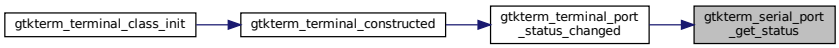
self	The serial port

Returns

GtkTermSerialPortState The status of the serial port.

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.16.3.13 gtkterm_serial_port_get_string()

```
char * gtkterm_serial_port_get_string (
            GtkTermSerialPort * self )
```

Convert port config to a string.

This is used for setting the configuration in the statusbar and window title.

Parameters

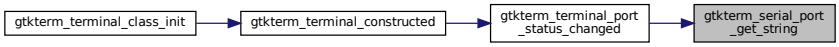
self	The SerialPort structure

Returns

The port configuration as string.

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.16.3.14 gtkterm_serial_port_handle()

```
static void gtkterm_serial_port_handle (
            GtkTermSerialPort * self,
            const char * action )  [inline], [static]
```

Based on the action return from uevent we handle to open of close the port.

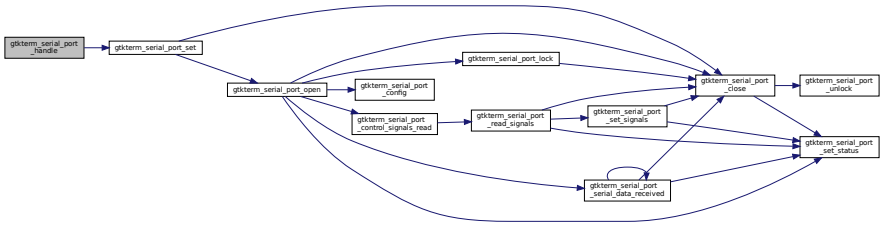
Note: This is an inline function. Without inline it wont work!.

Parameters

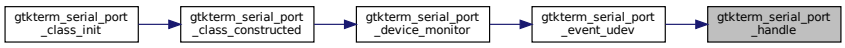
self	Pointer to the serial port.
action	The action we are performing with this device.

Referenced by gtkterm_serial_port_event_udev().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.15 gtkterm_serial_port_handle_usr1()

```
static bool gtkterm_serial_port_handle_usr1 (
            gpointer user_data )  [static]
```

Handles USR1 signal. It opens the port.

Parameters

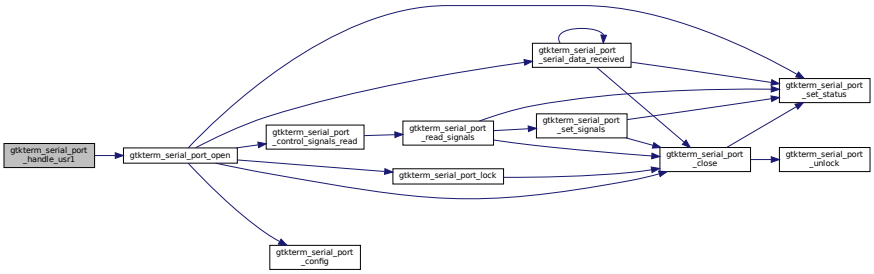
user_data	The serial port structure. Last param when installing the signal

Returns

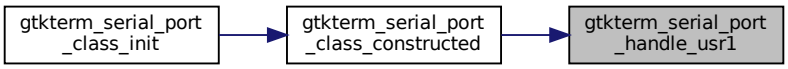
Continue the signal operation.

Referenced by gtkterm_serial_port_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.16 gtkterm_serial_port_handle_usr2()

```
static bool gtkterm_serial_port_handle_usr2 (
            gpointer user_data )  [static]
```

Handles USR2 signal. It closes the port.

Parameters

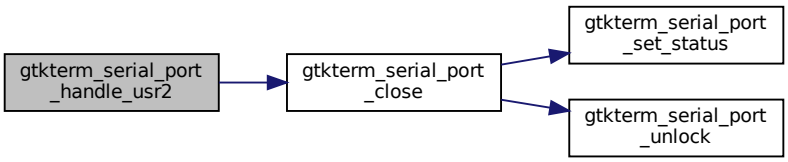
user_data	The serial port structure. Last param when installing the signal

Returns

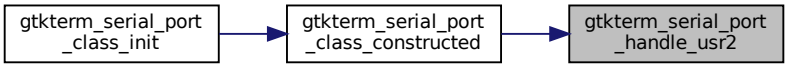
Continue the signal operation.

Referenced by gtkterm_serial_port_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.17 gtkterm_serial_port_init()

```
static void gtkterm_serial_port_init (
            GtkTermSerialPort * self )  [static]
```

Initialize the serial with the config parameters.

Parameters

self	The port we are initializing.

Not yet connected

6.16.3.18 gtkterm_serial_port_lock()

```
int gtkterm_serial_port_lock (
            GtkTermSerialPort * self,
            GError ** error )
```

Locks the serial port.

Parameters

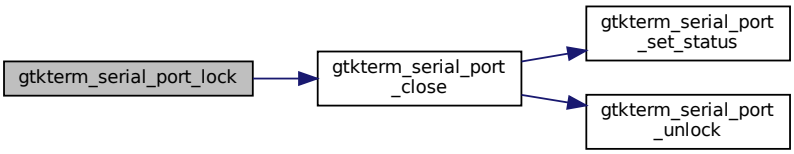
self	The serial port
error	The error occured when locking the port

Returns

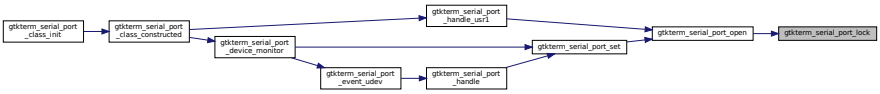
int The result of the lock operation.

Referenced by gtkterm_serial_port_open().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.19 gtkterm_serial_port_new()

```
GtkTermSerialPort * gtkterm_serial_port_new (
            port_config_t * port_conf )
```

Create a new serial port object.

This also binds the parameter to the properties of the serial port.

Parameters

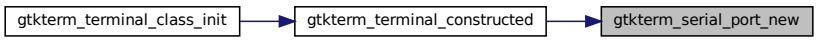
port_conf	The section for the configuration in this terminal

Returns

The serial_port object.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.16.3.20 gtkterm_serial_port_open()

```
static int gtkterm_serial_port_open (
            GtkTermSerialPort * self )  [static]
```

Connects to the serial port.

Local functions

The settings for this port will be set. If needed port will be locked.

Parameters

self	The configuration class.

Returns

The result of the operation.

Handle to cancel async read operaton

Lock the port

get termios for the file descriptor

And back it up

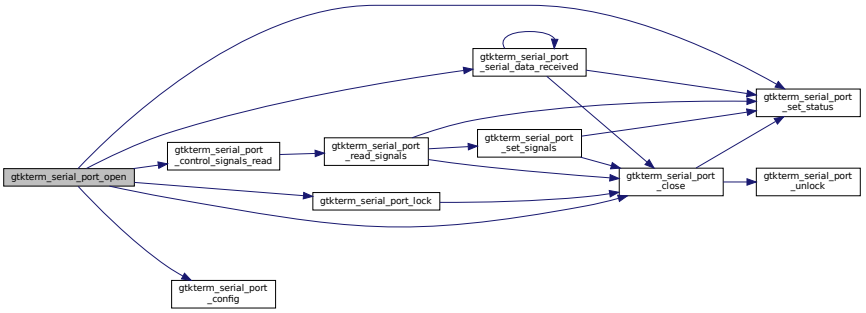
Set the created termios to the file descriptor

Flush the in- output data which is not written/read

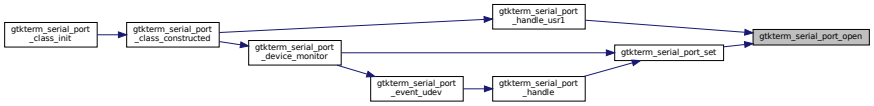
Set the streams for communicating with the device

Referenced by gtkterm_serial_port_handle_usr1(), and gtkterm_serial_port_set().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.21 gtkterm_serial_port_read_signals()

```
static int gtkterm_serial_port_read_signals (
            GtkTermSerialPort * self )  [static]
```

Does the actual reading of the serial signals.

Parameters

self	The serial port to read the signals for.

Returns

int The terminal status bits.

reset RTS (default = receive)

Check if we have af valid file desciptor and if the file destriptor points to a terminal device (tty∗)

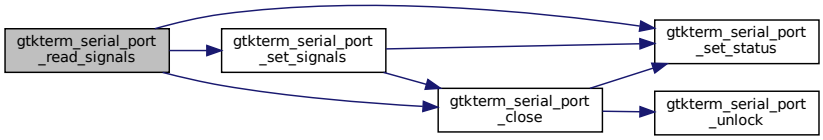
Get the terminal statusbits

Ignore EINVAL, as some serial ports genuinely lack these lines Thanks to Elie De Brauwer on ubuntu launchpad

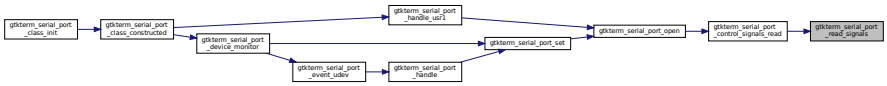
Apparently recently trying this on Linux PTYs fails with ENOTTY instead, so exempt this as well

Referenced by gtkterm_serial_port_control_signals_read().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.22 gtkterm_serial_port_serial_data_received()

```
static void gtkterm_serial_port_serial_data_received (
            GObject * source,
            GAsyncResult * res,
            gpointer user_data )  [static]
```

Callback where data is received from the input stream.

Because of the async operation the listener has to restart at the end of this callback. It isn't a continious loop.

Parameters

source	The inputstream
res	The result of the async function
user_data	The serial port struct.

Read bytes from the inputstream

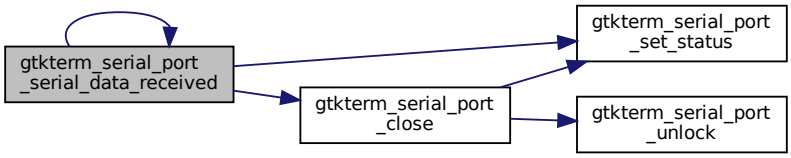
Todo send to terminal window and show in infobar

Send signal to buffer when new data is arrived

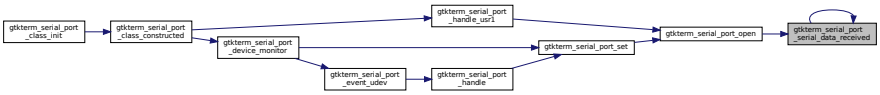
Restart listener

Referenced by gtkterm_serial_port_open(), and gtkterm_serial_port_serial_data_received().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.23 gtkterm_serial_port_serial_data_transmit()

```
static int gtkterm_serial_port_serial_data_transmit (
            GObject * object,
            gpointer data,
            unsigned int length,
            gpointer user_data )  [static]
```

Callback for signal where data is transmitted to the output stream of the port.

If necessary the RS485 signal are set.

Parameters

object	Not used
data	The string with data to send.
length	The length of the string.
user_data	The serial port struct.

Returns

int The bytes writen to the outputstream

Check if we have a string $>$ 0 and an active open port

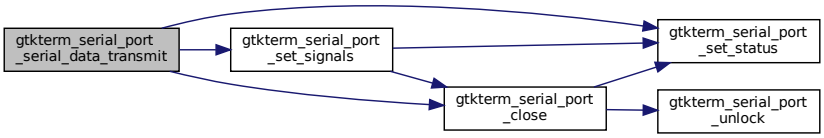
Are we in RS485 half-duplex mode

Wait till all chars are send

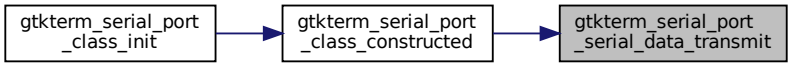
Reset RTS (end of send, now receiving back)

Referenced by gtkterm_serial_port_class_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.24 gtkterm_serial_port_set()

```
static void gtkterm_serial_port_set (
            GtkTermSerialPort * self,
            GtkTermSerialPortStatus status )  [inline], [static]
```

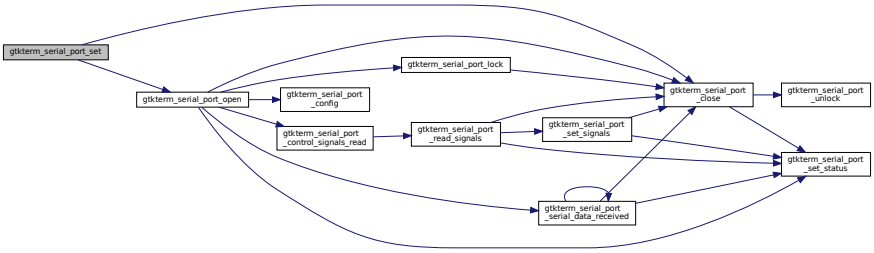
Opens or closes the serial port.

Parameters

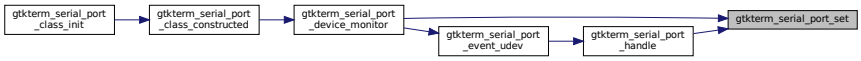
self	The serial port structure.
status	The new status for this port

Referenced by gtkterm_serial_port_device_monitor(), and gtkterm_serial_port_handle().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.25 gtkterm_serial_port_set_property()

```
static void gtkterm_serial_port_set_property (
            GObject * object,
            unsigned int prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Set the property of the GtkTermSerialPort structure.

This is used to initialize the variables when creating a new serial_port

Parameters

object	The object.
prop↩ _id	The id of the property to set.
value	The value for the property
pspec	Metadata for property setting.

Referenced by gtkterm_serial_port_class_init().

Here is the caller graph for this function:

6.16.3.26 gtkterm_serial_port_set_signals()

```
void gtkterm_serial_port_set_signals (
            GtkTermSerialPort * self,
            unsigned int param )
```

Set the signals for the Serial Port structure.

Parameters

self	The serial port structure.
param	The signals for the serial port.

Get the terminal status

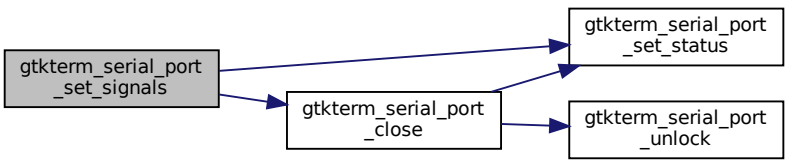
Set the DTR signal

Failure during setting the signal

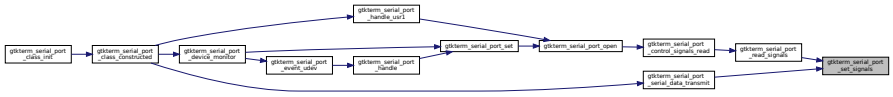
Set the RTS signal

Referenced by gtkterm_serial_port_read_signals(), and gtkterm_serial_port_serial_data_transmit().

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.27 gtkterm_serial_port_set_status()

```
void gtkterm_serial_port_set_status (
            GtkTermSerialPort * self,
            GtkTermSerialPortState new_status,
            GError * error )
```

Set the status of the serial port.

Parameters

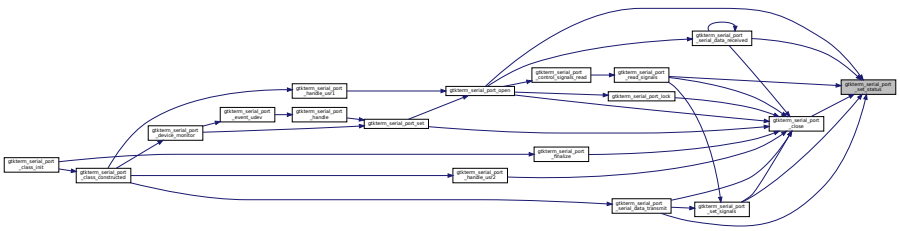
self	The serial port
new_status	The new status of the serial port.
error	The port error to set.

If there is a previous error, clear it

Notify that we have a change on the port

Referenced by gtkterm_serial_port_close(), gtkterm_serial_port_open(), gtkterm_serial_port_read_signals(), gtkterm_serial_port_serial_data_received(), gtkterm_serial_port_serial_data_transmit(), and gtkterm_serial_port_set_signals().

Here is the caller graph for this function:



6.16.3.28 gtkterm_serial_port_unlock()

```
int gtkterm_serial_port_unlock (
            GtkTermSerialPort * self )
```

Unlocks the serial port.

Parameters

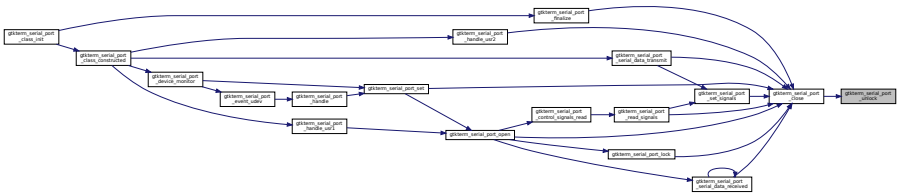
self	The serial port

Returns

int The result of the lock operation.

Referenced by gtkterm_serial_port_close().

Here is the caller graph for this function:



6.16.4 Variable Documentation

6.16.4.1 gtkterm_serial_port_properties

```
GParamSpec* gtkterm_serial_port_properties[N_PROPS] = {NULL}  [static]
```

Referenced by gtkterm_serial_port_class_init().

6.16.4.2 GtkTermSerialPortStateString

```
const char GtkTermSerialPortStateString[][DEFAULT_STRING_LEN]
```

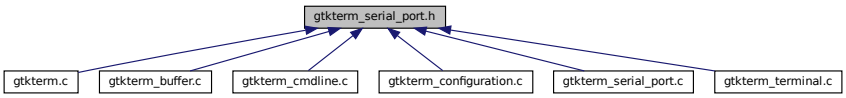
Initial value:

```
= {
    "Connected",
    "Disconnected",
    "Error"
}
```

Referenced by gtkterm_terminal_port_status_changed().

6.17 gtkterm_serial_port.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct port_config_t

  The typedef for the serial configuration.

Macros

- #define GTKTERM_TYPE_SERIAL_PORT gtkterm_serial_port_get_type ()

Typedefs

- typedef struct _GtkTermSerialPort GtkTermSerialPort

Enumerations

- enum GtkTermSerialPortStatus { GTKTERM_SERIAL_PORT_OPEN , GTKTERM_SERIAL_PORT_CLOSE }
- enum GtkTermSerialPortParity { GTKTERM_SERIAL_PORT_PARITY_NONE , GTKTERM_SERIAL_PORT_PARITY_EVEN , GTKTERM_SERIAL_PORT_PARITY_ODD }
- enum GtkTermSerialPortFlowControl { GTKTERM_SERIAL_PORT_FLOWCONTROL_NONE , GTKTERM_SERIAL_PORT_FL , GTKTERM_SERIAL_PORT_FLOWCONTROL_RTS_CTS , GTKTERM_SERIAL_PORT_FLOWCONTROL_RS485_HD }
- enum GtkTermSerialPortState { GTKTERM_SERIAL_PORT_CONNECTED , GTKTERM_SERIAL_PORT_DISCONNECTED , GTKTERM_SERIAL_PORT_ERROR }

Functions

- GtkTermSerialPort * gtkterm_serial_port_new (port_config_t *)

  Create a new serial port object.
- char * gtkterm_serial_port_get_string (GtkTermSerialPort *)

  Convert port config to a string.
- GtkTermSerialPortState gtkterm_serial_port_get_status (GtkTermSerialPort *)

  Return the status of the serial port.
- unsigned int gtkterm_serial_port_get_signals (GtkTermSerialPort *)

  Get the signals from the Serial Port structure.
- GError * gtkterm_serial_port_get_error (GtkTermSerialPort *)

  Return the last error which occured.

Variables

- const char GtkTermSerialPortStateString [ ][DEFAULT_STRING_LEN]

6.17.1 Macro Definition Documentation

6.17.1.1 GTKTERM_TYPE_SERIAL_PORT

```
#define GTKTERM_TYPE_SERIAL_PORT gtkterm_serial_port_get_type ()
```

6.17.2 Typedef Documentation

6.17.2.1 GtkTermSerialPort

```
typedef struct _GtkTermSerialPort GtkTermSerialPort
```

6.17.3 Enumeration Type Documentation

6.17.3.1 GtkTermSerialPortFlowControl

```
enum GtkTermSerialPortFlowControl
```

Enumerator

GTKTERM_SERIAL_PORT_FLOWCONTROL_NONE	
GTKTERM_SERIAL_PORT_FLOWCONTROL_XON_XOFF	
GTKTERM_SERIAL_PORT_FLOWCONTROL_RTS_CTS	
GTKTERM_SERIAL_PORT_FLOWCONTROL_RS485_HD	

6.17.3.2 GtkTermSerialPortParity

```
enum GtkTermSerialPortParity
```

Enumerator

GTKTERM_SERIAL_PORT_PARITY_NONE	
GTKTERM_SERIAL_PORT_PARITY_EVEN	
GTKTERM_SERIAL_PORT_PARITY_ODD	

6.17.3.3 GtkTermSerialPortState

```
enum GtkTermSerialPortState
```

Enumerator

GTKTERM_SERIAL_PORT_CONNECTED	
GTKTERM_SERIAL_PORT_DISCONNECTED	
GTKTERM_SERIAL_PORT_ERROR	

6.17.3.4 GtkTermSerialPortStatus

```
enum GtkTermSerialPortStatus
```

Enumerator

GTKTERM_SERIAL_PORT_OPEN	
GTKTERM_SERIAL_PORT_CLOSE	

6.17.4 Function Documentation

6.17.4.1 gtkterm_serial_port_get_error()

```
GError * gtkterm_serial_port_get_error (
            GtkTermSerialPort * self )
```

Return the last error which occured.

Parameters

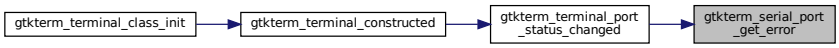
self	The serial port

Returns

GERrror The pointer to the GError struct

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.17.4.2 gtkterm_serial_port_get_signals()

```
unsigned int gtkterm_serial_port_get_signals (
            GtkTermSerialPort * self )
```

Get the signals from the Serial Port structure.

This is also used for external reading.

Parameters

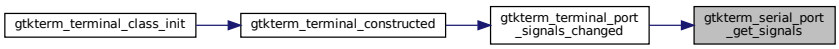
self	The serial port to read the signals for.

Returns

unsigned int The latest serial port signals

Referenced by gtkterm_terminal_port_signals_changed().

Here is the caller graph for this function:



6.17.4.3 gtkterm_serial_port_get_status()

```
GtkTermSerialPortState gtkterm_serial_port_get_status (
            GtkTermSerialPort * self )
```

Return the status of the serial port.

Parameters

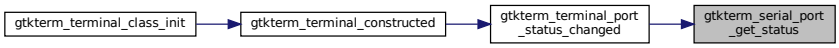
self	The serial port

Returns

GtkTermSerialPortState The status of the serial port.

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.17.4.4 gtkterm_serial_port_get_string()

```
char * gtkterm_serial_port_get_string (
            GtkTermSerialPort * self )
```

Convert port config to a string.

Global functions

This is used for setting the configuration in the statusbar and window title.

Parameters

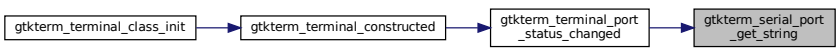
self	The SerialPort structure

Returns

The port configuration as string.

Referenced by gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.17.4.5 gtkterm_serial_port_new()

```
GtkTermSerialPort * gtkterm_serial_port_new (
            port_config_t * port_conf )
```

Create a new serial port object.

This also binds the parameter to the properties of the serial port.

Parameters

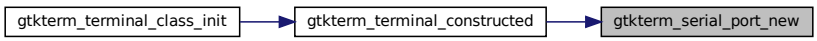
port_conf	The section for the configuration in this terminal

Returns

The serial_port object.

Referenced by gtkterm_terminal_constructed().

Here is the caller graph for this function:



6.17.5 Variable Documentation

6.17.5.1 GtkTermSerialPortStateString

```
const char GtkTermSerialPortStateString[][DEFAULT_STRING_LEN]  [extern]
```

Referenced by gtkterm_terminal_port_status_changed().

6.18 gtkterm_serial_port.h

Go to the documentation of this file.

```
1 /*************************************************************************/
2 /* gtkterm_serial_port.h                                                 */
3 /* -------                                                               */
4 /*           GTKTerm Software                                            */
5 /*                     (c) Julien Schmitt                                */
6 /*                                                                       */
7 /* --------------------------------------------------------------------- */
8 /*                                                                       */
9 /*   Purpose                                                             */
10 /*      Serial port access functions                                      */
11 /*      - Header file -                                                   */
12 /*                                                                       */
13 /*************************************************************************/
```

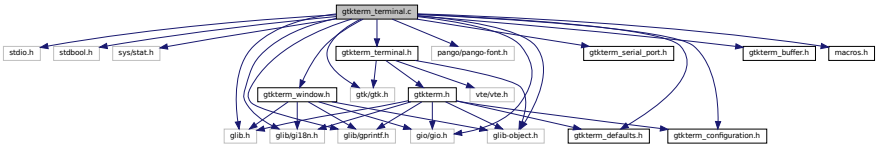
```
14
15 #ifndef GTKTERM_SERIAL_H_
16 #define GTKTERM_SERIAL_H_
17
18 typedef enum {
19     GTKTERM_SERIAL_PORT_OPEN,
20     GTKTERM_SERIAL_PORT_CLOSE
21
22 } GtkTermSerialPortStatus;
23
24 typedef enum {
25     GTKTERM_SERIAL_PORT_PARITY_NONE,
26     GTKTERM_SERIAL_PORT_PARITY_EVEN,
27     GTKTERM_SERIAL_PORT_PARITY_ODD
28
29 } GtkTermSerialPortParity;
30
31 typedef enum {
32     GTKTERM_SERIAL_PORT_FLOWCONTROL_NONE,
33     GTKTERM_SERIAL_PORT_FLOWCONTROL_XON_XOFF,
34     GTKTERM_SERIAL_PORT_FLOWCONTROL_RTS_CTS,
35     GTKTERM_SERIAL_PORT_FLOWCONTROL_RS485_HD,
36
37 } GtkTermSerialPortFlowControl;
38
39 typedef enum {
40     GTKTERM_SERIAL_PORT_CONNECTED,
41     GTKTERM_SERIAL_PORT_DISCONNECTED,
42     GTKTERM_SERIAL_PORT_ERROR
43
44 } GtkTermSerialPortState;
45
50 typedef struct {
51
52     char *port;
53     long int baudrate;
54     int bits;
55     int stopbits;
56     GtkTermSerialPortParity parity;
57     GtkTermSerialPortFlowControl flow_control;
58     int rs485_rts_time_before_transmit;
59     int rs485_rts_time_after_transmit;
60     bool disable_port_lock;
62 } port_config_t;
63
64 G_BEGIN_DECLS
65
66 #define GTKTERM_TYPE_SERIAL_PORT gtkterm_serial_port_get_type ()
67 G_DECLARE_FINAL_TYPE (GtkTermSerialPort, gtkterm_serial_port, GTKTERM, SERIAL_PORT, GObject)
68 typedef struct _GtkTermSerialPort GtkTermSerialPort;
69
70 GtkTermSerialPort *gtkterm_serial_port_new (port_config_t *);
71
73 char* gtkterm_serial_port_get_string (GtkTermSerialPort *);
74 GtkTermSerialPortState gtkterm_serial_port_get_status (GtkTermSerialPort *);
75 unsigned int gtkterm_serial_port_get_signals (GtkTermSerialPort *);
76 GError *gtkterm_serial_port_get_error (GtkTermSerialPort *);
77
78 extern const char GtkTermSerialPortStateString [][DEFAULT_STRING_LEN];
79
80 G_END_DECLS
81
82 #endif // GTKTERM_SERIAL_H
```

6.19 gtkterm_terminal.c File Reference

```
#include <stdio.h>
#include <stdbool.h>
#include <sys/stat.h>
#include <glib.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include <glib-object.h>
#include <gtk/gtk.h>
#include <gio/gio.h>
#include <pango/pango-font.h>
```

```
#include "gtkterm_defaults.h"
#include "gtkterm_window.h"
#include "gtkterm_serial_port.h"
#include "gtkterm_terminal.h"
#include "gtkterm_buffer.h"
#include "macros.h"
#include "gtkterm_configuration.h"
```

Include dependency graph for gtkterm_terminal.c:



Classes

- struct GtkTermTerminalPrivate
- struct _GtkTermTerminal
- struct _GtkTermTerminalClass

Enumerations

- enum GtkTermTerminalView { GTKTERM_TERMINAL_VIEW_TEXT , GGTKTERM_TERMINAL_VIEW_HEX }
- enum { PROP_0 , PROP_SECTION , PROP_GTKTERM_APP , PROP_MAIN_WINDOW , N_PROPS }

Functions

- void gtkterm_terminal_view_ascii (GtkTermTerminal *, char *, uint)
- void gtkterm_terminal_view_hex (GtkTermTerminal *, char *, uint)
- GtkTermTerminal * gtkterm_terminal_new (char *section, GtkTerm *gtkterm_app, GtkTermWindow *main_window)

  Create a new terminal object.
- static void gtkterm_terminal_vte_data_received (VteTerminal *widget, char *text, unsigned int length, gpointer ptr)

  Callback when new data from the VTE widget is received.
- static void gtkterm_terminal_port_signals_changed (GObject *object, GParamSpec *pspec, gpointer user_data)

  When signalsof the serial port is changed we get a signal and have to update GtkTermWindow.
- static void gtkterm_terminal_port_status_changed (GObject *object, GParamSpec *pspec, gpointer user_data)

  When the status of the serial port is changed we get a signal and have to update GtkTermWindow.
- static void gtkterm_terminal_buffer_updated (GObject *object, gpointer data, unsigned int length, gpointer user_data)

  When the buffer is updated the terminal is notified data new data is available.
- static void gtkterm_terminal_constructed (GObject *object)

  Constructs the terminal.

- static void gtkterm_terminal_dispose (GObject *object)

  Called when distroying the terminal.
- static void gtkterm_terminal_set_property (GObject *object, unsigned int prop_id, const GValue *value, GParamSpec *pspec)

  Set the property of the GtkTermTerminal structure.
- static void gtkterm_terminal_class_init (GtkTermTerminalClass *class)

  Initializing the terminal class.
- static void gtkterm_terminal_init (GtkTermTerminal *self)

  Initialize the terminal with the actions, etc.
- void gtkterm_terminal_view_ascii (GtkTermTerminal *self, char *data, unsigned int length)

  Outputs a string to the terminal widget in ASCII.
- void gtkterm_terminal_view_hex (GtkTermTerminal *self, char *data, unsigned int length)

  Outputs a string to the terminal widget in HEX layout.

Variables

- static GParamSpec * gtkterm_terminal_properties [N_PROPS] = {NULL}

6.19.1 Enumeration Type Documentation

6.19.1.1 anonymous enum

```
anonymous enum
```

Enumerator

PROP_0	
PROP_SECTION	
PROP_GTKTERM_APP	
PROP_MAIN_WINDOW	
N_PROPS	

6.19.1.2 GtkTermTerminalView

```
enum GtkTermTerminalView
```

Enumerator

GTKTERM_TERMINAL_VIEW_TEXT	
GGTKTERM_TERMINAL_VIEW_HEX	

6.19.2 Function Documentation

6.19.2.1 gtkterm_terminal_buffer_updated()

```
static void gtkterm_terminal_buffer_updated (
            GObject * object,
            gpointer data,
            unsigned int length,
            gpointer user_data )  [static]
```

When the buffer is updated the terminal is notified data new data is available.

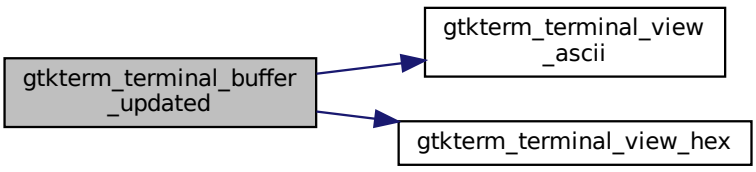
Depending of the setting the output will be in ASCII for HEX.

Parameters

object	Not used.
data	The new string of data in the buffer.
length	The length of the new string
user_data	The terminal.

Referenced by gtkterm_terminal_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:

6.19.2.2 gtkterm_terminal_class_init()

```
static void gtkterm_terminal_class_init (
            GtkTermTerminalClass * class )  [static]
```

Initializing the terminal class.

Setting the properties and callback functions

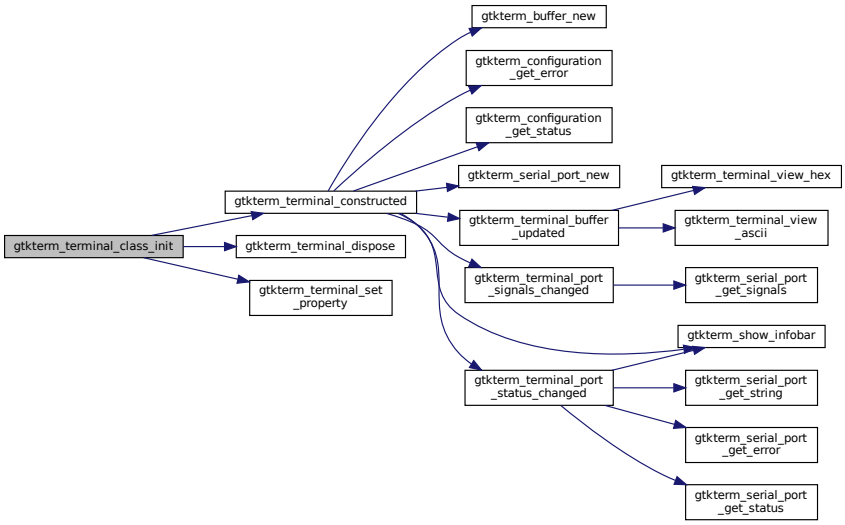
Parameters

class	The terminal class

Connect the serial port

We received data from the VTE widget and send it to the serial port

Parameters to hand over at creation of the object We need the section to load the config from the keyfile.Here is the call graph for this function:



6.19.2.3 gtkterm_terminal_constructed()

```
static void gtkterm_terminal_constructed (
            GObject * object )  [static]
```

Constructs the terminal.

Setup signals, initialize terminal etc.

Parameters

object	The terminal object we are constructing.

Check if the config file exists, if not it will be created

Todo : convert to notify on message

Load the configuration from [Section] into the port and terminal config. Take [section] as input, term/port conf are the pointers to the return values.

Create the serial port. The buffer will be a propertie, so they can exchange data without interference of the terminal.

Create a buffer for the terminal

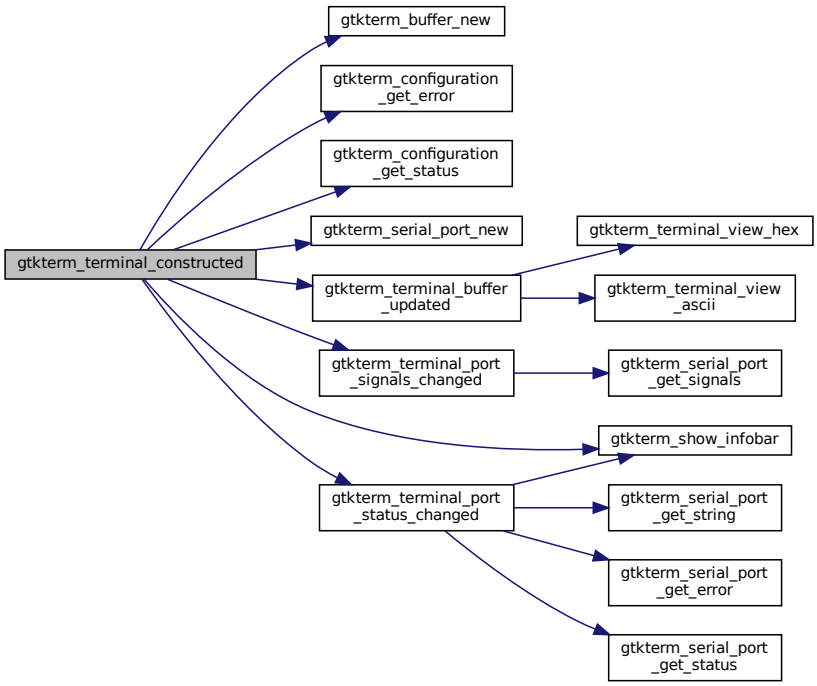
Send initial notify to update the status bar

Set terminal properties.

Todo : make configurable from the config file

Referenced by gtkterm_terminal_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.19.2.4 gtkterm_terminal_dispose()

```
static void gtkterm_terminal_dispose (
            GObject * object )  [static]
```

Called when distroying the terminal.

This is used to clean up an freeing the variables in the terminal structure.

Parameters

object	The object.

Referenced by gtkterm_terminal_class_init().

Here is the caller graph for this function:

6.19.2.5 gtkterm_terminal_init()

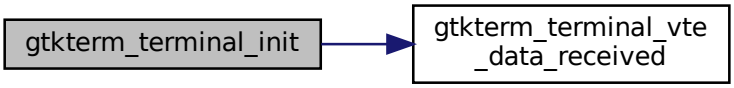
```
static void gtkterm_terminal_init (
            GtkTermTerminal * self )  [static]
```

Initialize the terminal with the actions, etc.

Parameters

self	The terminal we are initializing.

Here is the call graph for this function:



6.19.2.6 gtkterm_terminal_new()

```
GtkTermTerminal * gtkterm_terminal_new (
            char * section,
            GtkTerm * gtkterm_app,
            GtkTermWindow * main_window )
```

Create a new terminal object.

This also binds the parameter to the properties of the terminal.

Parameters

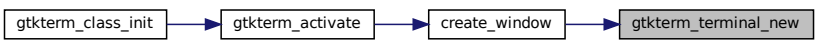
section	The section for the configuration in this terminal
gtkterm_app	The GTKTerm application
main_window	The main_window this terminal is attached to.

Returns

The terminal object.

Referenced by create_window().

Here is the caller graph for this function:



6.19.2.7 gtkterm_terminal_port_signals_changed()

```
static void gtkterm_terminal_port_signals_changed (
            GObject * object,
            GParamSpec * pspec,
            gpointer user_data )  [static]
```

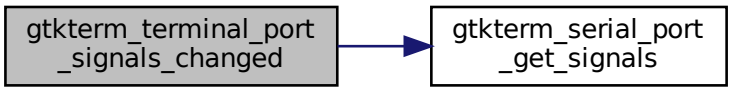
When signalsof the serial port is changed we get a signal and have to update GtkTermWindow.

Parameters

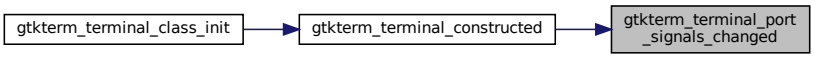
object	The serial port.
pspec	Not used.
user_data	The active terminal.

Referenced by gtkterm_terminal_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.19.2.8 gtkterm_terminal_port_status_changed()

```
static void gtkterm_terminal_port_status_changed (
            GObject * object,
            GParamSpec * pspec,
            gpointer user_data )  [static]
```

When the status of the serial port is changed we get a signal and have to update GtkTermWindow.

Parameters

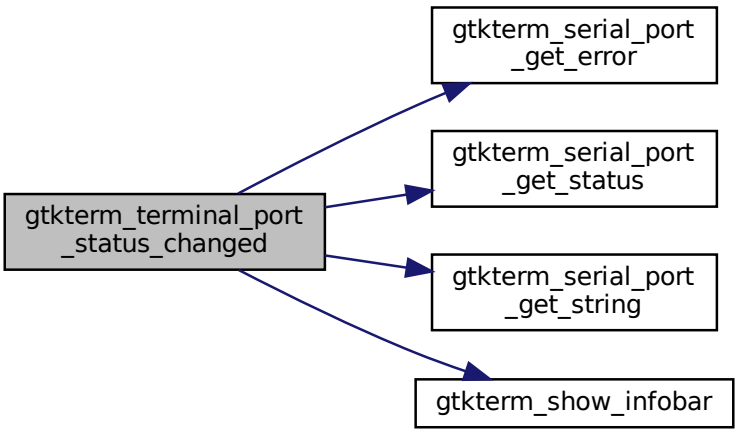
object	The serial port.
pspec	Not used.
user_data	The active terminal.

Todo convert to notify signal on message...

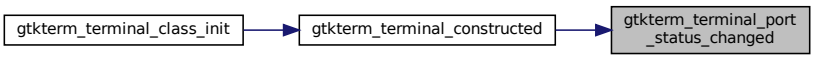
Update the statusbar and main window title

Referenced by gtkterm_terminal_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.19.2.9 gtkterm_terminal_set_property()

```
static void gtkterm_terminal_set_property (
            GObject * object,
            unsigned int prop_id,
            const GValue * value,
            GParamSpec * pspec )  [static]
```

Set the property of the GtkTermTerminal structure.

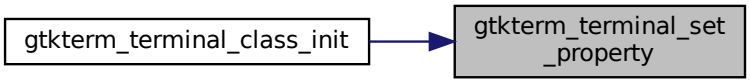
This is used to initialize the variables when creating a new terminal

Parameters

object	The object.
prop↩ _id	The id of the property to set.
value	The value for the property
pspec	Metadata for property setting.

Referenced by gtkterm_terminal_class_init().

Here is the caller graph for this function:



6.19.2.10 gtkterm_terminal_view_ascii() [1/2]

```
void gtkterm_terminal_view_ascii (
            GtkTermTerminal * ,
            char * ,
            uint  )
```

Referenced by gtkterm_terminal_buffer_updated().

Here is the caller graph for this function:

6.19.2.11 gtkterm_terminal_view_ascii() [2/2]

```
void gtkterm_terminal_view_ascii (
            GtkTermTerminal * self,
            char * data,
            unsigned int length )
```

Outputs a string to the terminal widget in ASCII.

Parameters

self	The terminal.
data	The string of data we want to show
length	The length of the string

6.19.2.12 gtkterm_terminal_view_hex() [1/2]

```
void gtkterm_terminal_view_hex (
            GtkTermTerminal * ,
            char * ,
            uint  )
```

Referenced by gtkterm_terminal_buffer_updated().

Here is the caller graph for this function:

6.19.2.13 gtkterm_terminal_view_hex() [2/2]

```
void gtkterm_terminal_view_hex (
            GtkTermTerminal * self,
            char * data,
            unsigned int length )
```

Outputs a string to the terminal widget in HEX layout.

Parameters

self	The terminal.
data	The string of data we want to show
length	The length of the string

6.19.2.14 gtkterm_terminal_vte_data_received()

```
static void gtkterm_terminal_vte_data_received (
            VteTerminal * widget,
```

```
char * text,
unsigned int length,
gpointer ptr )  [static]
```

Callback when new data from the VTE widget is received.

If echo is enabled the string is also send to the buffer. Which will update the terminal by sending the new data signal.

Parameters

widget	The VTE widget.
text	The text which is entered
length	The length of the text
ptr	Not used.

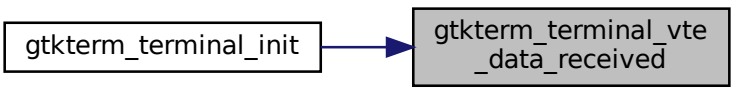
On echo send data to the buffer which will update the terminal

Convert to GBytes, needed for the buffer

Send signal to buffer when new data is arrived

Referenced by gtkterm_terminal_init().

Here is the caller graph for this function:



6.19.3 Variable Documentation

6.19.3.1 gtkterm_terminal_properties

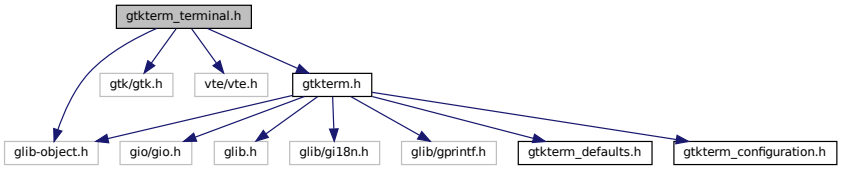
```
GParamSpec* gtkterm_terminal_properties[N_PROPS] = {NULL}  [static]
```

Referenced by gtkterm_terminal_class_init().

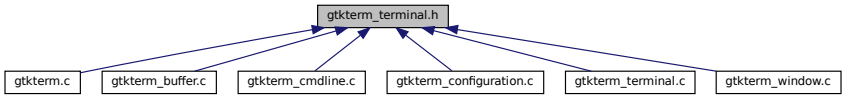
6.20 gtkterm_terminal.h File Reference

```
#include <glib-object.h>
#include <gtk/gtk.h>
#include <vte/vte.h>
#include "gtkterm.h"
```

Include dependency graph for gtkterm_terminal.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct term_config_t

  The typedef for the terminal configuration.

Macros

- #define GTKTERM_TYPE_TERMINAL gtkterm_terminal_get_type()

Typedefs

- typedef struct _GtkTermTerminal GtkTermTerminal

Functions

- GtkTermTerminal * gtkterm_terminal_new (char *, GtkTerm *, GtkTermWindow *)

  Create a new terminal object.

6.20.1 Macro Definition Documentation

6.20.1.1 GTKTERM_TYPE_TERMINAL

```
#define GTKTERM_TYPE_TERMINAL gtkterm_terminal_get_type()
```

6.20.2 Typedef Documentation

6.20.2.1 GtkTermTerminal

```
typedef struct _GtkTermTerminal GtkTermTerminal
```

6.20.3 Function Documentation

6.20.3.1 gtkterm_terminal_new()

```
GtkTermTerminal * gtkterm_terminal_new (
            char * section,
            GtkTerm * gtkterm_app,
            GtkTermWindow * main_window )
```

Create a new terminal object.

This also binds the parameter to the properties of the terminal.

Parameters

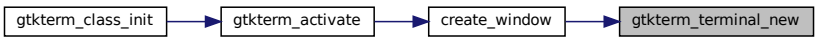
section	The section for the configuration in this terminal
gtkterm_app	The GTKTerm application
main_window	The main_window this terminal is attached to.

Returns

The terminal object.

Referenced by create_window().

Here is the caller graph for this function:



6.21 gtkterm_terminal.h

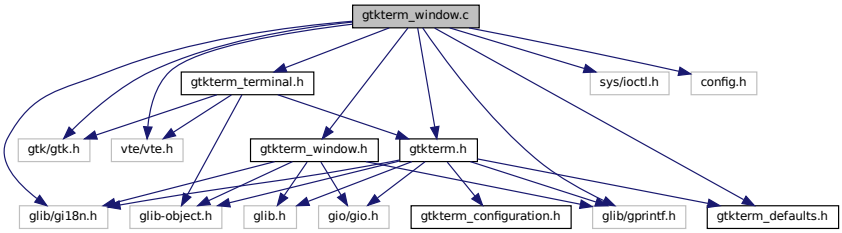
Go to the documentation of this file.

```
1 /*****************************************************************************/
2 /* gtkterm_terminal.h                                                        */
3 /* --------                                                                  */
4 /*           GTKTerm Software                                                */
5 /*                       (c) Julien Schmitt                                  */
6 /*                                                                           */
7 /* ------------------------------------------------------------------------- */
8 /*                                                                           */
9 /*   Purpose                                                                 */
10 /*      Handles all VTE in/output to/from serial port                         */
11 /*      - Header file -                                                       */
12 /*                                                                           */
13 /*****************************************************************************/
14 #ifndef GTKTERM_TERMINAL_H
15 #define GTKTERM_TERMINAL_H
16
17 #include <glib-object.h>
18 #include <gtk/gtk.h>
19 #include <vte/vte.h>
20
21 #include "gtkterm.h"
22
27 typedef struct {
28
29     bool block_cursor;
30     bool show_cursor;
31     char char_queue;
32     bool echo;
33     bool auto_lf;
34     bool auto_cr;
35     bool timestamp;
36     int delay;
37     int rows;
38     int columns;
39     int scrollback;
40     bool visual_bell;
41     GdkRGBA foreground_color;
42     GdkRGBA background_color;
43     PangoFontDescription *font;
45 } term_config_t;
46
47 G_BEGIN_DECLS
48
49 #define GTKTERM_TYPE_TERMINAL gtkterm_terminal_get_type()
50 G_DECLARE_FINAL_TYPE (GtkTermTerminal, gtkterm_terminal, GTKTERM, TERMINAL, VteTerminal)
51 typedef struct _GtkTermTerminal GtkTermTerminal;
52
53 GtkTermTerminal *gtkterm_terminal_new (char *, GtkTerm *, GtkTermWindow *);
54
55 G_END_DECLS
56
57 #endif // GTKTERM_TERMINAL_H
```

6.22 gtkterm_window.c File Reference

```
#include <gtk/gtk.h>
#include <vte/vte.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
#include <sys/ioctl.h>
#include "config.h"
#include "gtkterm_defaults.h"
#include "gtkterm.h"
#include "gtkterm_window.h"
#include "gtkterm_terminal.h"
```

Include dependency graph for gtkterm_window.c:



Classes

- struct _GtkTermWindow

  MainWindow specific variables here.

Macros

- #define SERIAL_SIGNALS 6

Functions

- static void gtkterm_window_update_statusbar (GtkTermWindow *window, gpointer section, gpointer serial_config_string, gpointer serial_status, gpointer user_data)

  Callbackfunction for updating window title and statusbus.
- static void config_status_bar (GtkTermWindow *window)

  Set the statusbar with all relevant fields.
- static void update_statusbar (GtkTermWindow *window, gpointer section, gpointer serial_config_string, gpointer serial_status)

  Updates the statusbar with the active terminal configuration.
- void set_window_title (GtkTermWindow *window, gpointer serial_config_string)

  Sets title of the window.
- static void on_gtkterm_about (GSimpleAction *action, GVariant *parameter, gpointer user_data)

  Show the About dialog.
- static void on_gtkterm_toggle_state (GSimpleAction *, GVariant *, gpointer)
- static void on_gtkterm_toggle_dark (GSimpleAction *action, GVariant *state, gpointer user_data)

  Toggles the dark mode setting.
- void create_window (GApplication *app, GtkTermWindow *window)
- void gtkterm_show_infobar (GtkTermWindow *window, char *message, int message_type)

  Shows a message into the Infobar.
- static void open_response_cb (GtkNativeDialog *dialog, int response_id, gpointer user_data)
- static void on_gtkterm_send_raw (GSimpleAction *action, GVariant *parameter, gpointer user_data)
- static void on_gtkterm_toggle_radio (GSimpleAction *action, GVariant *parameter, gpointer user_data)
- static void gtkterm_window_set_signals (GtkTermWindow *window, unsigned int port_signals, gpointer user_data)

  Set the serial signals (DTR etc) in the status bar.
- static void on_gtkterm_toggle_radio_state (GSimpleAction *action, GVariant *state, gpointer user_data)

  Toggles the radio option in the menubar.

- static void clicked_cb (GtkWidget *widget, GtkTermWindow *window)
- static void gtkterm_window_store_state (GtkTermWindow *window)

  Stores the setting of the window in the Gnome Settings.
- static void gtkterm_window_load_state (GtkTermWindow *window)

  Loads the setting of the window from the Gnome Settings.
- static void gtkterm_window_init (GtkTermWindow *window)

  Initialize the window with the actions, tools etc.
- static void gtkterm_window_constructed (GObject *object)

  Constructs the window.
- static void gtkterm_window_size_allocate (GtkWidget *widget, int width, int height, int baseline)

  Assigns size an position to the widget.
- static void surface_state_changed (GtkWidget *widget)

  Called when the surface state is changed (min/max).
- static void gtkterm_window_realize (GtkWidget *widget)

  Windows realize.
- static void gtkterm_window_unrealize (GtkWidget *widget)

  Windows unrealize.
- static void gtkterm_window_dispose (GObject *object)

  Called when distroying the window.
- static void gtkterm_window_class_init (GtkTermWindowClass *class)

  Initializing the window class.

Variables

- static unsigned int signal_flags [ ] = {TIOCM_DTR, TIOCM_RTS, TIOCM_CTS, TIOCM_CD, TIOCM_DSR, TIOCM_RI}
- static char const * serial_signal [ ] = {"DTR", "RTS", "CTS", "CD", "DSR", "RI"}
- static GActionEntry gtkterm_window_entries [ ]
- static GActionEntry win_entries [ ]

6.22.1 Macro Definition Documentation

6.22.1.1 SERIAL_SIGNALS

```
#define SERIAL_SIGNALS 6
```

6.22.2 Function Documentation

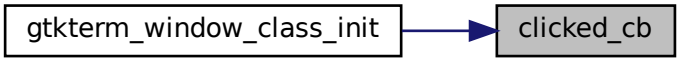
6.22.2.1 clicked_cb()

```
static void clicked_cb (
             GtkWidget * widget,
             GtkTermWindow * window )  [static]
```

To be implemented

Referenced by gtkterm_window_class_init().

Here is the caller graph for this function:



6.22.2.2 config_status_bar()

```
void config_status_bar (
             GtkTermWindow * window )  [static]
```

Set the statusbar with all relevant fields.

Parameters

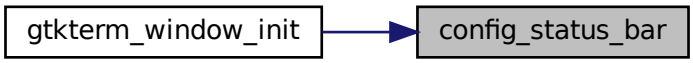
window	The GtkTermWindow with the statusbar.

Fields for the configuration and port

Fill in the serial signals The signals are appended at the statusbox so they can glide along when resizing the window

Referenced by gtkterm_window_init().

Here is the caller graph for this function:



6.22.2.3 create_window()

```
void create_window (
            GApplication * app,
            GtkTermWindow * window )
```

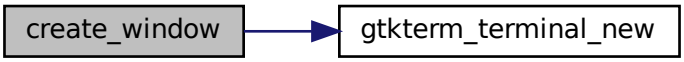
Todo remove and set it with properties.

Create a new terminal window and send section and keyfile as parameter GTKTERM_TERMINAL then can load the right section.

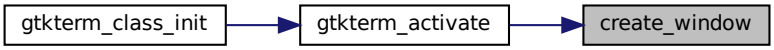
Make the VTE window scrollable

Referenced by gtkterm_activate().

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.2.4 gtkterm_show_infobar()

```
void gtkterm_show_infobar (
            GtkTermWindow * window,
            char * message,
            int message_type )
```

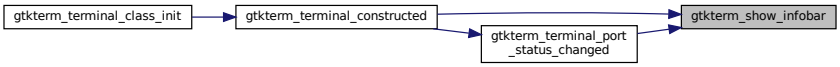
Shows a message into the Infobar.

Parameters

window	The window with the infobar property.
message	The message we want to show
message_type	The type of message GTK_MESSAGE_*

Referenced by gtkterm_terminal_constructed(), and gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.22.2.5 gtkterm_window_class_init()

```
static void gtkterm_window_class_init (
            GtkTermWindowClass * class )  [static]
```

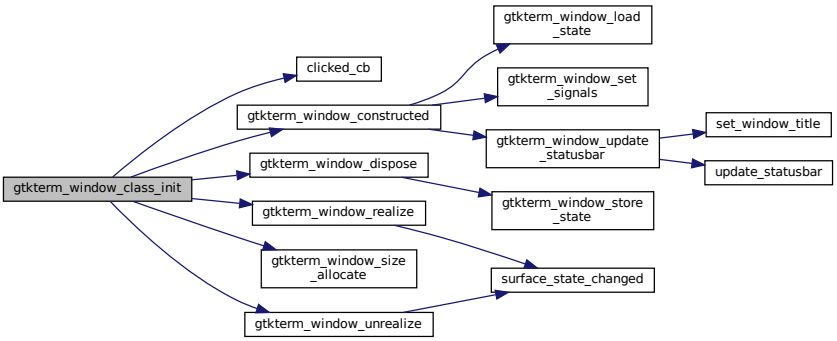
Initializing the window class.

Setting the signals, the UI and callback functions

Parameters

class	The window class

Here is the call graph for this function:



6.22.2.6 gtkterm_window_constructed()

```
static void gtkterm_window_constructed (
            GObject * object )  [static]
```

Constructs the window.

Parameters

object	The window object we are constructing.

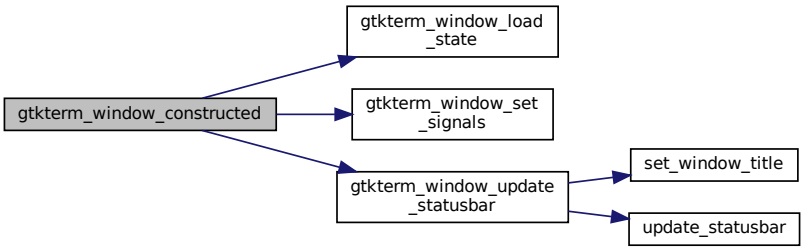
Create a new terminal window and send section and keyfile as parameter GTKTERM_TERMINAL then can load the right section.

Make the VTE window scrollable

Connect to the terminal_changed so we can update the statusbar and window title

Referenced by gtkterm_window_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.22.2.7 gtkterm_window_dispose()

```
static void gtkterm_window_dispose (
            GObject * object )  [static]
```

Called when distroying the window.

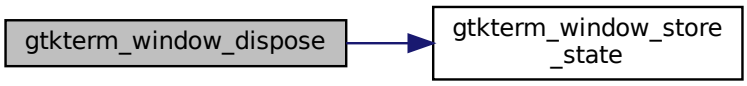
This is used to clean up an freeing the variables in the window structure.

Parameters

object	The object.

Referenced by gtkterm_window_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.22.2.8 gtkterm_window_init()

```
static void gtkterm_window_init (
            GtkTermWindow * window )  [static]
```

Initialize the window with the actions, tools etc.

Parameters

window	The window we are initializing.

Todo : Rename it

Here is the call graph for this function:

6.22.2.9 gtkterm_window_load_state()

```
static void gtkterm_window_load_state (
            GtkTermWindow * window )  [static]
```

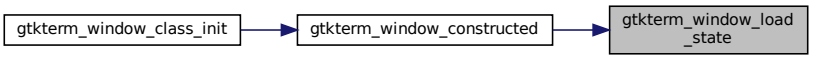
Loads the setting of the window from the Gnome Settings.

Parameters

window	The window we load the settings for.

Referenced by gtkterm_window_constructed().

Here is the caller graph for this function:



6.22.2.10 gtkterm_window_realize()

```
static void gtkterm_window_realize (
            GtkWidget * widget )  [static]
```

Windows realize.

Operations to finish realizing the widget.

Parameters

widget	The widget.

Referenced by gtkterm_window_class_init().

Here is the call graph for this function:

Here is the caller graph for this function:

6.22.2.11 gtkterm_window_set_signals()

```
static void gtkterm_window_set_signals (
            GtkTermWindow * window,
            unsigned int port_signals,
            gpointer user_data )  [static]
```

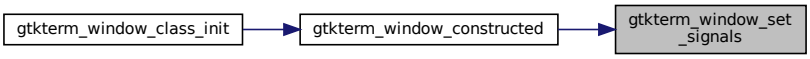
Set the serial signals (DTR etc) in the status bar.

Parameters

window	The GtkTermWindow with the statusbar.
port_signals	The port signals to set.
user_data	Not used.

Referenced by gtkterm_window_constructed().

Here is the caller graph for this function:



6.22.2.12 gtkterm_window_size_allocate()

```
static void gtkterm_window_size_allocate (
            GtkWidget * widget,
            int width,
            int height,
            int baseline )  [static]
```

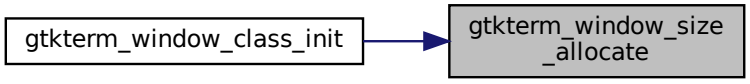
Assigns size an position to the widget.

Parameters

widget	The widget.
width	The width of the widget.
height	The height of the widget.
baseline	The baseline for this widget.

Referenced by gtkterm_window_class_init().

Here is the caller graph for this function:



6.22.2.13 gtkterm_window_store_state()

```
static void gtkterm_window_store_state (
            GtkTermWindow * window )  [static]
```

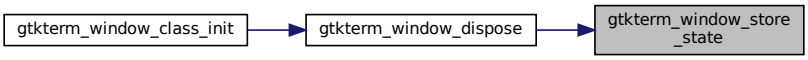
Stores the setting of the window in the Gnome Settings.

Parameters

window	The window we store the settings for.

Referenced by gtkterm_window_dispose().

Here is the caller graph for this function:



6.22.2.14 gtkterm_window_unrealize()

```
static void gtkterm_window_unrealize (
            GtkWidget * widget )  [static]
```

Windows unrealize.

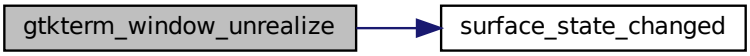
Operations when removing the widget.

Parameters

widget	The widget.

Referenced by gtkterm_window_class_init().

Here is the call graph for this function:



Here is the caller graph for this function:

6.22.2.15 gtkterm_window_update_statusbar()

```
static void gtkterm_window_update_statusbar (
            GtkTermWindow * window,
            gpointer section,
            gpointer serial_config_string,
            gpointer serial_status,
            gpointer user_data )  [static]
```

Callbackfunction for updating window title and statusbus.

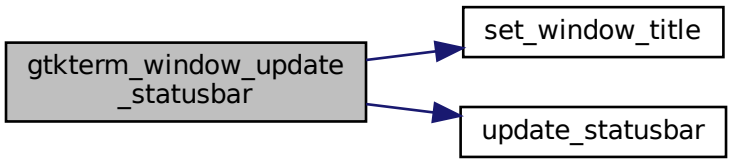
Internal functions

Parameters

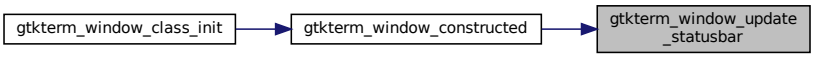
window	The GtkTermWindow which we update
section	The active section of the terminal
serial_config_string	The connectionstring of the serial port
serial_status	The status of the serial port.
user_data	Not used.

Referenced by gtkterm_window_constructed().

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.2.16 on_gtkterm_about()

```
static void on_gtkterm_about (
             GSimpleAction * action,
             GVariant * parameter,
             gpointer user_data )  [static]
```

Show the About dialog.

Menu callbacks

Parameters

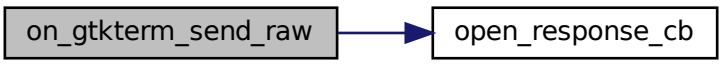
action	Not used.
parameter	Not used.
user_data	Pointer to the GtkTermWindow.

6.22.2.17 on_gtkterm_send_raw()

```
static void on_gtkterm_send_raw (
             GSimpleAction * action,
             GVariant * parameter,
             gpointer user_data )  [static]
```

Todo rewrite for GTKTerm

Here is the call graph for this function:



6.22.2.18 on_gtkterm_toggle_dark()

```
static void on_gtkterm_toggle_dark (
             GSimpleAction * action,
             GVariant * state,
             gpointer user_data )  [static]
```

Toggles the dark mode setting.

Parameters

action	The action interface.
state	The new dark-mode setting
user_data	Not used.

6.22.2.19 on_gtkterm_toggle_radio()

```
static void on_gtkterm_toggle_radio (
             GSimpleAction * action,
             GVariant * parameter,
             gpointer user_data )  [static]
```

Todo rewrite for GTKTerm

6.22.2.20 on_gtkterm_toggle_radio_state()

```
static void on_gtkterm_toggle_radio_state (
             GSimpleAction * action,
             GVariant * state,
             gpointer user_data )  [static]
```

Toggles the radio option in the menubar.

Parameters

action	The action interface.
state	The new radio setting
user_data	Not used.

6.22.2.21 on_gtkterm_toggle_state()

```
static void on_gtkterm_toggle_state (
             GSimpleAction * action,
             GVariant * parameter,
             gpointer user_data )  [static]
```

Todo rewrite for GTKTerm

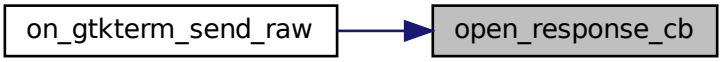
6.22.2.22 open_response_cb()

```
static void open_response_cb (
            GtkNativeDialog * dialog,
            int response_id,
            gpointer user_data )  [static]
```

Todo rewrite for gtkterm

Referenced by on_gtkterm_send_raw().

Here is the caller graph for this function:



6.22.2.23 set_window_title()

```
void set_window_title (
            GtkTermWindow * window,
            gpointer serial_config_string )
```

Sets title of the window.

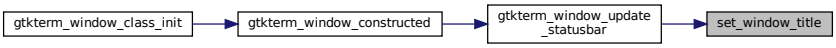
The title of the window is concatenated with the serial options of the active terminal window.

Parameters

window	The GtkTermWindow for which we set the title
serial_config_string	The connectionstring of the serial port.

Referenced by gtkterm_window_update_statusbar().

Here is the caller graph for this function:



6.22.2.24 surface_state_changed()

```
static void surface_state_changed (
            GtkWidget * widget )  [static]
```

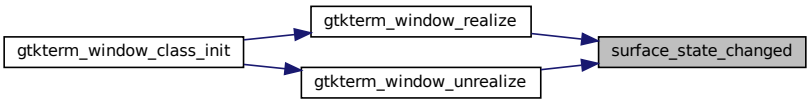
Called when the surface state is changed (min/max).

Parameters

widget	The widget.

Referenced by gtkterm_window_realize(), and gtkterm_window_unrealize().

Here is the caller graph for this function:



6.22.2.25 update_statusbar()

```
static void update_statusbar (
            GtkTermWindow * window,
            gpointer section,
            gpointer serial_config_string,
            gpointer serial_status )  [static]
```

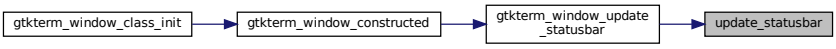
Updates the statusbar with the active terminal configuration.

Parameters

window	The GtkTermWindow with the statusbar.
section	The active section of the terminal
serial_config_string	The connectionstring of the serial port
serial_status	The status of the serial port.

Referenced by gtkterm_window_update_statusbar().

Here is the caller graph for this function:



6.22.3 Variable Documentation

6.22.3.1 gtkterm_window_entries

```
GActionEntry gtkterm_window_entries[]  [static]
```

Menu definitions and callbacks

Referenced by gtkterm_window_init().

6.22.3.2 serial_signal

```
char const* serial_signal[] = {"DTR", "RTS", "CTS", "CD", "DSR", "RI"}  [static]
```

Referenced by config_status_bar().

6.22.3.3 signal_flags

```
unsigned int signal_flags[] = {TIOCM_DTR, TIOCM_RTS, TIOCM_CTS, TIOCM_CD, TIOCM_DSR, TIOCM_RI}
[static]
```

Serial signals

Referenced by gtkterm_window_set_signals().

6.22.3.4 win_entries

```
GActionEntry win_entries[]  [static]
```

Initial value:

```
= {
  { "about", on_gtkterm_about, NULL, NULL, NULL }
}
```

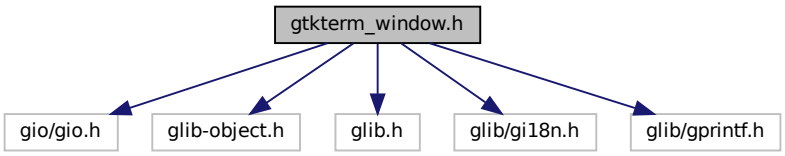
GtkTermWindow definitions and callbacks

Referenced by gtkterm_window_init().

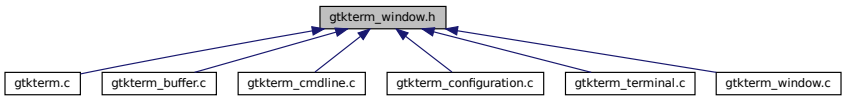
6.23 gtkterm_window.h File Reference

```
#include <gio/gio.h>
#include <glib-object.h>
#include <glib.h>
#include <glib/gi18n.h>
#include <glib/gprintf.h>
```

Include dependency graph for gtkterm_window.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GTKTERM_TYPE_GTKTERM_WINDOW gtkterm_window_get_type()

Typedefs

- typedef struct _GtkTermWindow GtkTermWindow

Functions

- void create_window (GApplication *, GtkTermWindow *)
- void gtkterm_show_infobar (GtkTermWindow *, char *, int)

  Shows a message into the Infobar.

6.23.1 Macro Definition Documentation

6.23.1.1 GTKTERM_TYPE_GTKTERM_WINDOW

```
#define GTKTERM_TYPE_GTKTERM_WINDOW gtkterm_window_get_type()
```

6.23.2 Typedef Documentation

6.23.2.1 GtkTermWindow

```
typedef struct _GtkTermWindow GtkTermWindow
```

6.23.3 Function Documentation

6.23.3.1 create_window()

```
void create_window (
            GApplication * app,
            GtkTermWindow * window )
```

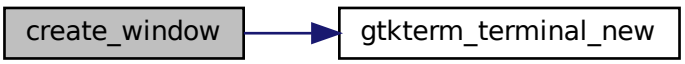
Todo remove and set it with properties.

Create a new terminal window and send section and keyfile as parameter GTKTERM_TERMINAL then can load the right section.

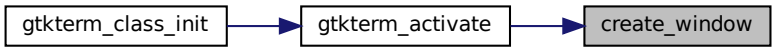
Make the VTE window scrollable

Referenced by gtkterm_activate().

Here is the call graph for this function:



Here is the caller graph for this function:



6.23.3.2 gtkterm_show_infobar()

```
void gtkterm_show_infobar (
            GtkTermWindow * window,
            char * message,
            int message_type )
```

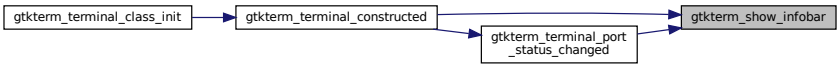
Shows a message into the Infobar.

Parameters

window	The window with the infobar property.
message	The message we want to show
message_type	The type of message GTK_MESSAGE_∗

Referenced by gtkterm_terminal_constructed(), and gtkterm_terminal_port_status_changed().

Here is the caller graph for this function:



6.24 gtkterm_window.h

Go to the documentation of this file.

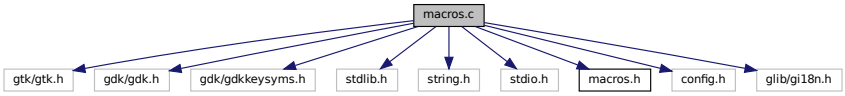
```
1 #ifndef GTKTERM_WINDOW_H
2 #define GTKTERM_WINDOW_H
3
4 #include <gio/gio.h>
5 #include <glib-object.h>
6 #include <glib.h>
7 #include <glib/gi18n.h>
8 #include <glib/gprintf.h>
9
10
11 G_BEGIN_DECLS
12
13 #define GTKTERM_TYPE_GTKTERM_WINDOW gtkterm_window_get_type()
14 G_DECLARE_FINAL_TYPE (GtkTermWindow, gtkterm_window, GTKTERM, WINDOW, GtkApplicationWindow)
15 typedef struct _GtkTermWindow GtkTermWindow;
16
17 void create_window (GApplication *, GtkTermWindow *);
18 void gtkterm_show_infobar (GtkTermWindow *, char *, int);
19
20 G_END_DECLS
21
22 #endif // GTKTERM_WINDOW_H
```

6.25 macros.c File Reference

```
#include <gtk/gtk.h>
#include <gdk/gdk.h>
#include <gdk/gdkkeysyms.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "macros.h"
#include <config.h>
#include <glib/gi18n.h>
```

Include dependency graph for macros.c:



Enumerations

- enum { COLUMN_SHORTCUT , COLUMN_ACTION , NUM_COLUMNS }

Functions

- int macro_count ()
- void convert_string_to_macros (char **string_list, int size)

  Convert the array of strings to macros.
- int convert_macros_to_string (char **string_list)
- static void macros_destroy (void)
- macro_t * get_shortcuts (int *size)
- void remove_shortcuts (void)

  Remove shortcuts from accel_group and free memory.

Variables

- macro_t * macros = NULL
- int nr_of_macros = 0

6.25.1 Enumeration Type Documentation

6.25.1.1 anonymous enum

```
anonymous enum
```

Todo : Migrate to GObject

Enumerator

COLUMN_SHORTCUT	
COLUMN_ACTION	
NUM_COLUMNS	

6.25.2 Function Documentation

6.25.2.1 convert_macros_to_string()

```
int convert_macros_to_string (
            char ** string_list )
```

Convert the in memory macros to an array of strings for storage in file Must be NULL terminated

Number of strings is 2x the macros (shortcut and action)

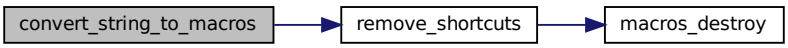
6.25.2.2 convert_string_to_macros()

```
void convert_string_to_macros (
            char ** string_list,
            int size )
```

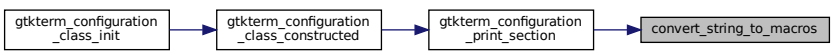
Convert the array of strings to macros.

Referenced by gtkterm_configuration_print_section().

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.2.3 get_shortcuts()

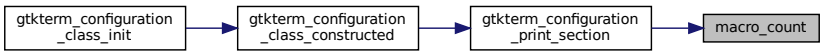
```
macro_t * get_shortcuts (
            int * size )
```

6.25.2.4 macro_count()

```
int macro_count ( )
```

Referenced by gtkterm_configuration_print_section().

Here is the caller graph for this function:



6.25.2.5 macros_destroy()

```
static void macros_destroy (
            void ) [static]
```

Free all macro-member memory

Referenced by remove_shortcuts().

Here is the caller graph for this function:

6.25.2.6 remove_shortcuts()

```
void remove_shortcuts (
            void  )
```

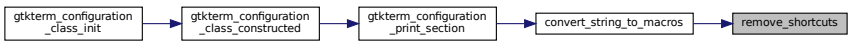
Remove shortcuts from accel_group and free memory.

Clean up all macros

Referenced by convert_string_to_macros().

Here is the call graph for this function:

Here is the caller graph for this function:



6.25.3 Variable Documentation

6.25.3.1 macros

```
macro_t* macros = NULL
```

Referenced by convert_macros_to_string(), convert_string_to_macros(), get_shortcuts(), gtkterm_configuration_print_section(), macros_destroy(), and remove_shortcuts().

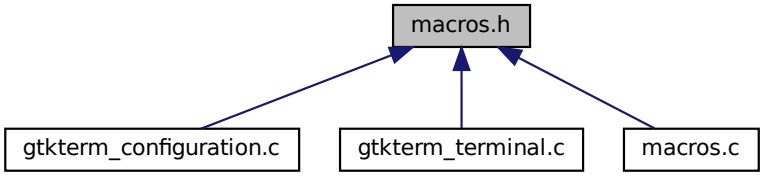
6.25.3.2 nr_of_macros

```
int nr_of_macros = 0
```

Referenced by convert_macros_to_string(), convert_string_to_macros(), and macro_count().

6.26 macros.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct macro_t

  Define macro structure type.

Functions

- void remove_shortcuts (void)

  Remove shortcuts from accel_group and free memory.
- void add_shortcuts (void)
- macro_t * get_shortcuts (gint *)
- void convert_string_to_macros (char **, int)

  Convert the array of strings to macros.
- int convert_macros_to_string (char **)
- int macro_count ()

Variables

- macro_t * macros

6.26.1 Function Documentation

6.26.1.1 add_shortcuts()

```
void add_shortcuts (
            void  )
```

6.26.1.2 convert_macros_to_string()

```
int convert_macros_to_string (
            char ** string_list )
```

Convert the in memory macros to an array of strings for storage in file Must be NULL terminated

Number of strings is 2x the macros (shortcut and action)

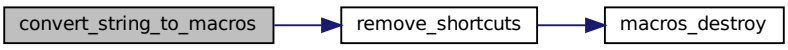
6.26.1.3 convert_string_to_macros()

```
void convert_string_to_macros (
            char ** string_list,
            int size )
```

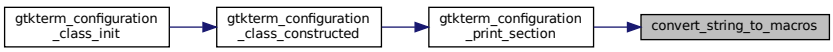
Convert the array of strings to macros.

Referenced by gtkterm_configuration_print_section().

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.1.4 get_shortcuts()

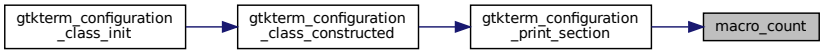
```
macro_t * get_shortcuts (
            gint *  )
```

6.26.1.5 macro_count()

```
int macro_count ( )
```

Referenced by gtkterm_configuration_print_section().

Here is the caller graph for this function:



6.26.1.6 remove_shortcuts()

```
void remove_shortcuts (
            void  )
```

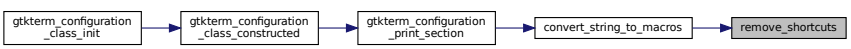
Remove shortcuts from accel_group and free memory.

Clean up all macros

Referenced by convert_string_to_macros().

Here is the call graph for this function:

Here is the caller graph for this function:



6.26.2 Variable Documentation

6.26.2.1 macros

```
macro_t* macros  [extern]
```

Referenced by convert_macros_to_string(), convert_string_to_macros(), get_shortcuts(), gtkterm_configuration_print_section(), macros_destroy(), and remove_shortcuts().

6.27 macros.h

Go to the documentation of this file.

```
1 /*****************************************************************************
2 * macros.h
3 * --------
4 *           GTKTerm Software
5 *                     (c) Julien Schmitt
6 *
7 * ---------------------------------------------------------------------------
8 *
9 * \brief Purpose
10 *        Functions for the management of the macros
11 *        - Header file -
12 *
13 *****************************************************************************/
14
15 #ifndef MACROS_H_
16 #define MACROS_H_
17
21 typedef struct
22 {
23     char *shortcut;
24     char *action;
25     GClosure *closure;
26 }
27 macro_t;
28
29 //void config_macros(GtkAction *action, gpointer data);
30 void remove_shortcuts(void);
31 void add_shortcuts(void);
32 macro_t *get_shortcuts(gint *);
33
34 void convert_string_to_macros (char **, int);
35 int convert_macros_to_string (char **);
36
37 int macro_count ();
38
39 extern macro_t *macros;
40
41 #endif
```

Index