

# Passive-source Seismic-processing (PsSp)

0.1.0

Generated by Doxygen 1.9.8



<b>1 Passive-source Seismic-Processing</b>	<b>1</b>
1.1 Summary of Purpose . . . . .	1
1.2 Introduction . . . . .	1
<b>2 Todo List</b>	<b>3</b>
<b>3 Namespace Index</b>	<b>5</b>
3.1 Namespace List . . . . .	5
<b>4 Hierarchical Index</b>	<b>7</b>
4.1 Class Hierarchy . . . . .	7
<b>5 Class Index</b>	<b>9</b>
5.1 Class List . . . . .	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 pssp Namespace Reference . . . . .	11
6.1.1 Detailed Description . . . . .	12
6.1.2 Typedef Documentation . . . . .	13
6.1.2.1 ConsoleSink_mt . . . . .	13
6.1.2.2 ConsoleSink_st . . . . .	13
6.1.3 Enumeration Type Documentation . . . . .	13
6.1.3.1 Field . . . . .	13
6.1.3.2 Type . . . . .	17
6.1.4 Variable Documentation . . . . .	17
6.1.4.1 field_info . . . . .	17
6.1.4.2 field_num . . . . .	19
6.1.4.3 type_names . . . . .	21
6.2 pssp::about Namespace Reference . . . . .	21
6.2.1 Detailed Description . . . . .	21
6.2.2 Variable Documentation . . . . .	22
6.2.2.1 button_height . . . . .	22
6.2.2.2 button_width . . . . .	22
6.2.2.3 height . . . . .	22
6.2.2.4 text_height . . . . .	22
6.2.2.5 text_width . . . . .	22
6.2.2.6 width . . . . .	22
6.3 pssp::constants Namespace Reference . . . . .	23
6.3.1 Variable Documentation . . . . .	23
6.3.1.1 sac_bool . . . . .	23
6.3.1.2 sac_data . . . . .	23
6.3.1.3 sac_double . . . . .	23
6.3.1.4 sac_float . . . . .	23
6.3.1.5 sac_int . . . . .	24

6.3.1.6 sac_string . . . . .	24
6.4 pssp::datasheet Namespace Reference . . . . .	24
6.4.1 Detailed Description . . . . .	24
6.4.2 Variable Documentation . . . . .	24
6.4.2.1 cell_buffer . . . . .	24
6.4.2.2 edit_chars . . . . .	25
6.4.2.3 font_size . . . . .	25
6.4.2.4 max_chars . . . . .	25
6.5 pssp::mw Namespace Reference . . . . .	25
6.5.1 Detailed Description . . . . .	25
6.5.2 Variable Documentation . . . . .	25
6.5.2.1 menu_height . . . . .	25
6.5.2.2 minimum_x . . . . .	26
6.5.2.3 minimum_y . . . . .	26
6.6 pssp::structs Namespace Reference . . . . .	26
6.7 pssp::welcome Namespace Reference . . . . .	26
6.7.1 Detailed Description . . . . .	26
6.7.2 Variable Documentation . . . . .	27
6.7.2.1 button_height . . . . .	27
6.7.2.2 button_width . . . . .	27
6.7.2.3 height . . . . .	27
6.7.2.4 text_height . . . . .	27
6.7.2.5 text_width . . . . .	27
6.7.2.6 width . . . . .	27
<b>7 Class Documentation . . . . .</b>	<b>29</b>
7.1 pssp::AboutWindow Class Reference . . . . .	29
7.1.1 Detailed Description . . . . .	31
7.1.2 Constructor & Destructor Documentation . . . . .	31
7.1.2.1 AboutWindow() . . . . .	31
7.1.3 Member Function Documentation . . . . .	31
7.1.3.1 okay_cb() . . . . .	31
7.1.4 Member Data Documentation . . . . .	32
7.1.4.1 message . . . . .	32
7.1.4.2 message_ . . . . .	32
7.1.4.3 okay_button . . . . .	32
7.2 pssp::Application Class Reference . . . . .	33
7.2.1 Detailed Description . . . . .	33
7.2.2 Constructor & Destructor Documentation . . . . .	33
7.2.2.1 Application() . . . . .	33
7.2.3 Member Data Documentation . . . . .	34
7.2.3.1 main_window . . . . .	34

7.2.3.2 welcome_window . . . . .	34
7.3 pssp::datasheet::Cell Struct Reference . . . . .	34
7.3.1 Detailed Description . . . . .	36
7.3.2 Member Data Documentation . . . . .	36
7.3.2.1 alignment . . . . .	36
7.3.2.2 box_color . . . . .	36
7.3.2.3 box_type . . . . .	36
7.3.2.4 font . . . . .	36
7.3.2.5 full_box . . . . .	36
7.3.2.6 text . . . . .	36
7.3.2.7 text_box . . . . .	37
7.3.2.8 text_color . . . . .	37
7.4 pssp::ConsoleSink< Mutex > Class Template Reference . . . . .	37
7.4.1 Detailed Description . . . . .	38
7.4.2 Constructor & Destructor Documentation . . . . .	39
7.4.2.1 ConsoleSink() . . . . .	39
7.4.3 Member Function Documentation . . . . .	39
7.4.3.1 flush_() . . . . .	39
7.4.3.2 sink_it_() . . . . .	39
7.4.4 Member Data Documentation . . . . .	39
7.4.4.1 tty_ . . . . .	39
7.5 pssp::Datasheet Class Reference . . . . .	40
7.5.1 Constructor & Destructor Documentation . . . . .	43
7.5.1.1 Datasheet() . . . . .	43
7.5.2 Member Function Documentation . . . . .	44
7.5.2.1 done_editing() . . . . .	44
7.5.2.2 draw_cell() . . . . .	45
7.5.2.3 draw_generic_cell() . . . . .	46
7.5.2.4 draw_header_cell() . . . . .	46
7.5.2.5 event_callback() . . . . .	47
7.5.2.6 event_callback2() . . . . .	47
7.5.2.7 set_value_hide() . . . . .	48
7.5.2.8 start_editing() . . . . .	49
7.5.3 Member Data Documentation . . . . .	50
7.5.3.1 check_button . . . . .	50
7.5.3.2 edit_col . . . . .	50
7.5.3.3 edit_row . . . . .	50
7.5.3.4 input_manager . . . . .	50
7.5.3.5 max_col . . . . .	50
7.5.3.6 max_row . . . . .	50
7.5.3.7 sheet_manager . . . . .	51
7.6 pssp::structs::Geometry Struct Reference . . . . .	51

7.6.1 Detailed Description . . . . .	51
7.6.2 Member Data Documentation . . . . .	52
7.6.2.1 height . . . . .	52
7.6.2.2 width . . . . .	52
7.6.2.3 x_pos . . . . .	52
7.6.2.4 y_pos . . . . .	52
7.7 pssp::structs::Grid Struct Reference . . . . .	52
7.7.1 Detailed Description . . . . .	53
7.7.2 Member Data Documentation . . . . .	53
7.7.2.1 col . . . . .	53
7.7.2.2 col_span . . . . .	53
7.7.2.3 row . . . . .	53
7.7.2.4 row_span . . . . .	53
7.8 pssp::InputManager Class Reference . . . . .	54
7.8.1 Detailed Description . . . . .	55
7.8.2 Constructor & Destructor Documentation . . . . .	55
7.8.2.1 InputManager() . . . . .	55
7.8.3 Member Function Documentation . . . . .	56
7.8.3.1 cleanup() . . . . .	56
7.8.3.2 clear() . . . . .	56
7.8.3.3 done_editing() . . . . .	57
7.8.3.4 hide() . . . . .	57
7.8.3.5 input_cb() . . . . .	57
7.8.3.6 start_editing() . . . . .	58
7.8.3.7 value() . . . . .	58
7.8.3.8 visible() . . . . .	59
7.8.4 Member Data Documentation . . . . .	59
7.8.4.1 input_float . . . . .	59
7.8.4.2 input_int . . . . .	59
7.8.4.3 input_string . . . . .	59
7.8.4.4 modified . . . . .	59
7.9 pssp::MainWindow Class Reference . . . . .	59
7.9.1 Detailed Description . . . . .	62
7.9.2 Constructor & Destructor Documentation . . . . .	63
7.9.2.1 MainWindow() . . . . .	63
7.9.3 Member Function Documentation . . . . .	64
7.9.3.1 about_cb() . . . . .	64
7.9.3.2 append_tty() . . . . .	64
7.9.3.3 make_menu() . . . . .	64
7.9.3.4 make_tty() . . . . .	66
7.9.3.5 prevent_escape() . . . . .	66
7.9.3.6 quit_cb() . . . . .	67

7.9.3.7 show_about()	67
7.9.4 Member Data Documentation	67
7.9.4.1 about_window_	67
7.9.4.2 datasheet_	68
7.9.4.3 debug_tty	68
7.9.4.4 gridspace_	68
7.9.4.5 list_	68
7.9.4.6 logger	68
7.9.4.7 menu	68
7.9.4.8 name_	68
7.9.4.9 sink	69
7.9.4.10 status_bar_	69
7.10 pssp::SheetManager Class Reference	69
7.10.1 Constructor & Destructor Documentation	71
7.10.1.1 SheetManager()	71
7.10.2 Member Function Documentation	71
7.10.2.1 cols()	71
7.10.2.2 get()	72
7.10.2.3 get_bool()	72
7.10.2.4 get_double()	72
7.10.2.5 get_float()	72
7.10.2.6 get_int()	73
7.10.2.7 get_string()	73
7.10.2.8 resize_data()	73
7.10.2.9 rows()	74
7.10.2.10 set() [1/5]	74
7.10.2.11 set() [2/5]	74
7.10.2.12 set() [3/5]	74
7.10.2.13 set() [4/5]	74
7.10.2.14 set() [5/5]	75
7.10.3 Member Data Documentation	75
7.10.3.1 bools	75
7.10.3.2 doubles	75
7.10.3.3 floats	75
7.10.3.4 ints	75
7.10.3.5 strings	75
7.11 pssp::datasheet::Spec Struct Reference	76
7.11.1 Detailed Description	76
7.11.2 Member Data Documentation	76
7.11.2.1 header_height	76
7.11.2.2 header_width	77
7.11.2.3 height	77

---

7.11.2.4 width . . . . .	77
7.12 pssp::StatusBar Class Reference . . . . .	77
7.12.1 Constructor & Destructor Documentation . . . . .	78
7.12.1.1 StatusBar() . . . . .	78
7.12.2 Member Data Documentation . . . . .	79
7.12.2.1 left_box_ . . . . .	79
7.12.2.2 middle_box_ . . . . .	79
7.12.2.3 right_box_ . . . . .	79
7.13 pssp::trace_info Struct Reference . . . . .	79
7.13.1 Detailed Description . . . . .	80
7.13.2 Member Data Documentation . . . . .	80
7.13.2.1 array_col . . . . .	80
7.13.2.2 col . . . . .	80
7.13.2.3 name . . . . .	80
7.13.2.4 type . . . . .	80
7.14 pssp::WelcomeWindow Class Reference . . . . .	81
7.14.1 Detailed Description . . . . .	83
7.14.2 Constructor & Destructor Documentation . . . . .	83
7.14.2.1 WelcomeWindow() . . . . .	83
7.14.3 Member Function Documentation . . . . .	84
7.14.3.1 continue_cb() . . . . .	84
7.14.4 Member Data Documentation . . . . .	84
7.14.4.1 continue_button . . . . .	84
7.14.4.2 message . . . . .	84
7.14.4.3 message_ . . . . .	84
<b>Index . . . . .</b>	<b>85</b>



# Chapter 1

## Passive-source Seismic-Processing

Passive-source Seismic-processing (PsSp) aims to provide an OS-independent, graphically driven, and free seismic processing application targeted at passive-source seismologists.

### 1.1 Summary of Purpose

The purpose of this project is to **extend the productivity suite** of the passive-source seismologist. Great tools exist for writing manuscripts (such as MS Word, LibreOffice Write, LaTeX, and so on). Great tools exist for creating presentations (e.g. MS Powerpoint, Impress Persentation, and so on). Great tools exist for communicating with each other across the world (e.g. MS Outlook, Thunderbird, Zoom, MS Teams, and so on). What tools exist for actually doing the seismic analysis? Far too often it is whatever the analyst manages to kludge together. PsSp aims to fill this gap with a modern graphical-interface, fast computation, and some much needed quality of life functionality (undo/redo, notes, checkpoints, and so on).

### 1.2 Introduction

Despite the numerous seismological tools that exist (SAC, Seismic Unix, Computer Programs in Seismology, Obs↔Py, and so on), and by the nature of their design, the typical seismologist will **most likely** need to code their own tool(s) and workflow(s). Often, this takes the form of scripts/macros to stitch together the output from one program to the input of another—taking into account any necessary intermediate data transformations. Having the ability to do this is awesome, needing to do this is not. This leads to poorly written, designed, documented, and tested codes. Even mature programs suffer from these problems, placing the onus on the user to make up for the mistakes of the creator.

It gets worse. Scientists often choose a language out of convenience: Fortran because everyone uses it (often using archaic programming conventions that were best lost to their decade of origin); or Python because it's easy and a ton of fun when it breaks every few months after a library gets updated, commands get deprecated, or during the ugly transition from 2.x to 3.x; Matlab because it provides all the fun of code-breakage from Python **and** its language features are stuck behind a paywall—like playing a modern video game; or whatever other language is in vogue as the next greatest innovation in developing barely functional code quickly.

For an analyst pushing the envelop to develop entirely novel analysis approaches, programming will always be a necessity. For the numerous others that are focused exclusively on **using** already developed analysis methods, programming should not be necessary. PsSp will fill this gap as a modern software solution.



# Chapter 2

## Todo List

### File `Constants.hpp`

So far these are only related to SAC records and are used to prototype the interface. In the future, they'll be supplied by the sac-format library and not needed to be defined here.

### File `Datasheet.hpp`

Add sorting functional.

Add ability to drag and drop columns/rows.

Redo/Undo functionality.

Boolean cells use checkboxes (or switches).

### File `Enums.hpp`

Non-enums (constants) belong in `PSP/Utility/Constants.hpp`

### Namespace `pssp`

Move structs from other files to this file.

### Namespace `pssp::about`

Move this to `PSP/Utility/Constants.hpp`

### Class `pssp::ConsoleSink` < `Mutex` >

At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

### Namespace `pssp::datasheet`

Move constants to `PSP/Utility/Constants.hpp`

Move structs to `PSP/Utility/Structs.hpp`

### Member `pssp::Field`

This is for prototyping SAC-records, in the future this will be supplied by the sac-format library (once we're ready to read in SAC-files).

### Member `pssp::field_info`

Merge `field_num` into this.

### Member `pssp::field_num`

Merge into `field_Info`

### Class `pssp::InputManager`

### Class `pssp::MainWindow`

Work on record-organization sidebar object.

**Member `pssp::MainWindow::make_menu ()`**

Fix shallow menus that do not display on macOS (all menus must have depth).

**Member `pssp::MainWindow::quit_cb (Fl_Widget *menu, void *junk)`**

Request if the user wants to save first (if unsaved work).

Doesn't display on macOS when CMD+Q is hit (just closes).

BugFix: Doesn't display when keyboard input is captured by [Datasheet](#).

**Namespace `pssp::mw`**

Move this to PsSp/Utility/Constants.hpp

**Struct `pssp::trace_info`**

Move to PsSp/Utility/Structs.hpp

**Member `pssp::type_names`**

Move to PsSp/Utility/Constants.hpp

**Namespace `pssp::welcome`**

Move this to PsSp/Utility/Constants.hpp

**Class `pssp::WelcomeWindow`**

Auto-size window to size of message.

"Do not show again" checkbox.

## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">pssp</a> . . . . .	11
<a href="#">pssp::about</a> . . . . .	21
<a href="#">pssp::constants</a> . . . . .	23
<a href="#">pssp::datasheet</a> . . . . .	24
<a href="#">pssp::mw</a> . . . . .	25
<a href="#">pssp::structs</a> . . . . .	26
<a href="#">pssp::welcome</a> . . . . .	26



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

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

pssp::Application . . . . .	33
spdlog::sinks::base_sink	
pssp::ConsoleSink< Mutex > . . . . .	37
pssp::datasheet::Cell . . . . .	34
FI_Double_Window	
pssp::MainWindow . . . . .	59
FI_Grid	
pssp::StatusBar . . . . .	77
FI_Table	
pssp::Datasheet . . . . .	40
FI_Window	
pssp::AboutWindow . . . . .	29
pssp::WelcomeWindow . . . . .	81
pssp::structs::Geometry . . . . .	51
pssp::structs::Grid . . . . .	52
pssp::InputManager . . . . .	54
pssp::SheetManager . . . . .	69
pssp::datasheet::Spec . . . . .	76
pssp::trace_info . . . . .	79





## Chapter 5

# Class Index

### 5.1 Class List

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

<a href="#">pssp::AboutWindow</a>	
Class to provide the About Window . . . . .	29
<a href="#">pssp::Application</a>	
Main application class . . . . .	33
<a href="#">pssp::datasheet::Cell</a>	
Specify a datasheet cell. This includes placement, size, font, color, box-type, alignment, border-type, and content of a datasheet cell . . . . .	34
<a href="#">pssp::ConsoleSink&lt; Mutex &gt;</a>	
Sink (receiver) of log messages for PsSp console . . . . .	37
<a href="#">pssp::Datasheet</a>	40
<a href="#">pssp::structs::Geometry</a>	
FLTK <a href="#">Geometry</a> handling struct . . . . .	51
<a href="#">pssp::structs::Grid</a>	
FLTK <a href="#">Grid</a> definition struct . . . . .	52
<a href="#">pssp::InputManager</a>	
Manager of user-input . . . . .	54
<a href="#">pssp::MainWindow</a>	
Class to provide the Main Window . . . . .	59
<a href="#">pssp::SheetManager</a>	69
<a href="#">pssp::datasheet::Spec</a>	
Used to specify the size of <a href="#">Datasheet</a> cells . . . . .	76
<a href="#">pssp::StatusBar</a>	77
<a href="#">pssp::trace_info</a>	
Information for use in the <a href="#">Datasheet</a> . . . . .	79
<a href="#">pssp::WelcomeWindow</a>	
Class to provide a Welcome Window . . . . .	81



## Chapter 6

# Namespace Documentation

### 6.1 pssp Namespace Reference

#### Namespaces

- namespace [about](#)
- namespace [constants](#)
- namespace [datasheet](#)
- namespace [mw](#)
- namespace [structs](#)
- namespace [welcome](#)

#### Classes

- class [AboutWindow](#)  
*Class to provide the About Window.*
- class [Application](#)  
*Main application class.*
- class [ConsoleSink](#)  
*Sink (receiver) of log messages for PsSp console.*
- class [Datasheet](#)
- class [InputManager](#)  
*Manager of user-input.*
- class [MainWindow](#)  
*Class to provide the Main Window.*
- class [SheetManager](#)
- class [StatusBar](#)
- struct [trace\\_info](#)  
*Information for use in the [Datasheet](#).*
- class [WelcomeWindow](#)  
*Class to provide a Welcome Window.*

#### Typedefs

- using [ConsoleSink\\_mt](#) = [ConsoleSink](#)< std::mutex >  
*Multi-thread safe Console\_Sink.*
- using [ConsoleSink\\_st](#) = [ConsoleSink](#)< spdlog::details::null\_mutex >  
*Single-thread Console\_Sink.*

## Enumerations

- enum class [Type](#) {  
[string\\_](#), [int\\_](#), [float\\_](#), [double\\_](#),  
[bool\\_](#) }  
*Data-type enumeration.*
- enum class [Field](#) {  
[depmin](#), [depmax](#), [odelta](#), [resp0](#),  
[resp1](#), [resp2](#), [resp3](#), [resp4](#),  
[resp5](#), [resp6](#), [resp7](#), [resp8](#),  
[resp9](#), [stel](#), [stdp](#), [evel](#),  
[evdp](#), [mag](#), [user0](#), [user1](#),  
[user2](#), [user3](#), [user4](#), [user5](#),  
[user6](#), [user7](#), [user8](#), [user9](#),  
[dist](#), [az](#), [baz](#), [gcarc](#),  
[depmen](#), [cmpaz](#), [cmpinc](#), [xminimum](#),  
[xmaximum](#), [yminimum](#), [ymaximum](#), [delta](#),  
[b](#), [e](#), [o](#), [a](#),  
[t0](#), [t1](#), [t2](#), [t3](#),  
[t4](#), [t5](#), [t6](#), [t7](#),  
[t8](#), [t9](#), [f](#), [stla](#),  
[stlo](#), [evla](#), [evlo](#), [sb](#),  
[sdelta](#), [nzyear](#), [nzjday](#), [nzhour](#),  
[nzmin](#), [nzsec](#), [nzmsec](#), [nvhdr](#),  
[norid](#), [nevid](#), [npts](#), [nsnpts](#),  
[nwfid](#), [nxsize](#), [nysize](#), [iftype](#),  
[idep](#), [iztype](#), [iinst](#), [istreg](#),  
[ievreg](#), [ievtyp](#), [igual](#), [isynth](#),  
[imagtyp](#), [imagsrc](#), [ibody](#), [leven](#),  
[lpspol](#), [lovrok](#), [lcalda](#), [kstnm](#),  
[kevnrm](#), [khole](#), [ko](#), [ka](#),  
[kt0](#), [kt1](#), [kt2](#), [kt3](#),  
[kt4](#), [kt5](#), [kt6](#), [kt7](#),  
[kt8](#), [kt9](#), [kf](#), [kuser0](#),  
[kuser1](#), [kuser2](#), [kcmpnm](#), [knetwk](#),  
[kdatrd](#), [kinst](#), [data1](#), [data2](#) }  
*SAC-header/footer field enumeration.*

## Variables

- const std::unordered\_map< [Type](#), const std::string > [type\\_names](#)  
*Map Type to string-name.*
- const std::unordered\_map< size\_t, [Field](#) > [field\\_num](#)  
*Map of column number ([Datashheet](#)) to Field.*
- const std::unordered\_map< [Field](#), [trace\\_info](#) > [field\\_info](#)  
*Map Field to [trace\\_info](#).*

### 6.1.1 Detailed Description

:structs

Namespace for holding universal PsSp structs.

**Todo** Move structs from other files to this file.

## 6.1.2 Typedef Documentation

### 6.1.2.1 ConsoleSink\_mt

```
using pssp::ConsoleSink_mt = typedef ConsoleSink<std::mutex>
```

Multi-thread safe Console\_Sink.

### 6.1.2.2 ConsoleSink\_st

```
using pssp::ConsoleSink_st = typedef ConsoleSink<spdlog::details::null_mutex>
```

Single-thread Console\_Sink.

## 6.1.3 Enumeration Type Documentation

### 6.1.3.1 Field

```
enum class pssp::Field [strong]
```

SAC-header/footer field enumeration.

**Todo** This is for prototyping SAC-records, in the future this will be supplied by the sac-format library (once we're ready to read in SAC-files).

#### Enumerator

depmin	
depmax	
odelta	
resp0	
resp1	
resp2	
resp3	
resp4	
resp5	
resp6	
resp7	
resp8	
resp9	
stel	
stdp	
evel	
evdp	
mag	
user0	
user1	
user2	

## Enumerator

user3	
user4	
user5	
user6	
user7	
user8	
user9	
dist	
az	
baz	
gcarc	
depmen	
cmpaz	
cmpinc	
xminimum	
xmaximum	
yminimum	
ymaximum	
delta	
b	
e	
o	
a	
t0	
t1	
t2	
t3	
t4	
t5	
t6	
t7	
t8	
t9	
f	
stla	
stlo	
evla	
evlo	
sb	
sdelta	
nzyear	
nzjday	
nzhour	
nzmin	
nzsec	
nzmsec	
nvhdr	
norid	
nevid	
npts	
nsnpts	

## Enumerator

nwfid	
nxsize	
nysize	
iftype	
idep	
iztype	
iinst	
istreg	
ievreg	
ievtyp	
igual	
isynt	
imagtyp	
imagsrc	
ibody	
leven	
lpspol	
lovrok	
lcalda	
kstnm	
kevn	
khole	
ko	
ka	
kt0	
kt1	
kt2	
kt3	
kt4	
kt5	
kt6	
kt7	
kt8	
kt9	
kf	
kuser0	
kuser1	
kuser2	
kcmpnm	
knetwk	
kdatrd	
kinst	
data1	
data2	

```

00082 {
00083     depmin,
00084     depmax,
00085     odelta,
00086     resp0,
00087     resp1,
00088     resp2,
00089     resp3,

```

```
00090     resp4,  
00091     resp5,  
00092     resp6,  
00093     resp7,  
00094     resp8,  
00095     resp9,  
00096     stel,  
00097     stdp,  
00098     evel,  
00099     evdp,  
00100     mag,  
00101     user0,  
00102     user1,  
00103     user2,  
00104     user3,  
00105     user4,  
00106     user5,  
00107     user6,  
00108     user7,  
00109     user8,  
00110     user9,  
00111     dist,  
00112     az,  
00113     baz,  
00114     gcarc,  
00115     depmen,  
00116     cmpaz,  
00117     cmpinc,  
00118     xminimum,  
00119     xmaximum,  
00120     yminimum,  
00121     ymaximum,  
00122     delta,  
00123     b,  
00124     e,  
00125     o,  
00126     a,  
00127     t0,  
00128     t1,  
00129     t2,  
00130     t3,  
00131     t4,  
00132     t5,  
00133     t6,  
00134     t7,  
00135     t8,  
00136     t9,  
00137     f,  
00138     stla,  
00139     stlo,  
00140     evla,  
00141     evlo,  
00142     sb,  
00143     sdelta,  
00144     nzyear,  
00145     nzjday,  
00146     nzhour,  
00147     nzmin,  
00148     nzsec,  
00149     nzmsec,  
00150     nvhdr,  
00151     norid,  
00152     nevid,  
00153     npts,  
00154     nsnpts,  
00155     nwfid,  
00156     nxsize,  
00157     nysize,  
00158     iftype,  
00159     idep,  
00160     iztype,  
00161     iinst,  
00162     istreg,  
00163     ievreg,  
00164     ievtyp,  
00165     igual,  
00166     isynth,  
00167     imagtyp,  
00168     imagsrc,  
00169     ibody,  
00170     leven,  
00171     lspol,  
00172     lovrok,  
00173     lcalda,  
00174     kstnm,  
00175     kevnrm,  
00176     khole,
```



```

00177     ko,
00178     ka,
00179     kt0,
00180     kt1,
00181     kt2,
00182     kt3,
00183     kt4,
00184     kt5,
00185     kt6,
00186     kt7,
00187     kt8,
00188     kt9,
00189     kf,
00190     kuser0,
00191     kuser1,
00192     kuser2,
00193     kcmpnm,
00194     knetwk,
00195     kdatrd,
00196     kinst,
00197     data1,
00198     data2
00199 };

```

### 6.1.3.2 Type

```
enum class pssp::Type [strong]
```

Data-type enumeration.

Allows maintaining the type of data (string, integer, float, double, bool) for an object since this isn't supported by default in C++.

Enumerator

string↔ —	String data-type.
int_ —	Integer data-type.
float_ —	Float data-type.
double↔ —	Double data-type.
bool_ —	Boolean data-type.

```

00032     {
00033     string_,
00034     int_,
00035     float_,
00036     double_,
00037     bool_,
00038 };

```

## 6.1.4 Variable Documentation

### 6.1.4.1 field\_info

```
const std::unordered_map<Field, trace_info> pssp::field_info
```

Map Field to [trace\\_info](#).

Given a field, get its [trace\\_info](#) (column, array-colun, type-name, and Type).

This is needed for interacting with the [Datashheet](#).

**Todo** Merge field\_num into this.

```

00340                                     {
00341     // Floats
00342     {Field::depmin, {0, 0, "DepMin", Type::float_}},
00343     {Field::depmax, {1, 1, "DepMax", Type::float_}},
00344     {Field::odelta, {2, 2, "ODelta", Type::float_}},
00345     {Field::resp0, {3, 3, "Resp0", Type::float_}},
00346     {Field::resp1, {4, 4, "Resp1", Type::float_}},
00347     {Field::resp2, {5, 5, "Resp2", Type::float_}},
00348     {Field::resp3, {6, 6, "Resp3", Type::float_}},
00349     {Field::resp4, {7, 7, "Resp4", Type::float_}},
00350     {Field::resp5, {8, 8, "Resp5", Type::float_}},
00351     {Field::resp6, {9, 9, "Resp6", Type::float_}},
00352     {Field::resp7, {10, 10, "Resp7", Type::float_}},
00353     {Field::resp8, {11, 11, "Resp8", Type::float_}},
00354     {Field::resp9, {12, 12, "Resp9", Type::float_}},
00355     {Field::stel, {13, 13, "StEl", Type::float_}},
00356     {Field::stdp, {14, 14, "StDp", Type::float_}},
00357     {Field::evel, {15, 15, "EvEl", Type::float_}},
00358     {Field::evdp, {16, 16, "EvDp", Type::float_}},
00359     {Field::mag, {17, 17, "Mag", Type::float_}},
00360     {Field::user0, {18, 18, "User0", Type::float_}},
00361     {Field::user1, {19, 19, "User1", Type::float_}},
00362     {Field::user2, {20, 20, "User2", Type::float_}},
00363     {Field::user3, {21, 21, "User3", Type::float_}},
00364     {Field::user4, {21, 22, "User4", Type::float_}},
00365     {Field::user5, {23, 23, "User5", Type::float_}},
00366     {Field::user6, {24, 24, "User6", Type::float_}},
00367     {Field::user7, {25, 25, "User7", Type::float_}},
00368     {Field::user8, {26, 26, "User8", Type::float_}},
00369     {Field::user9, {27, 27, "User9", Type::float_}},
00370     {Field::dist, {28, 28, "Dist", Type::float_}},
00371     {Field::az, {29, 29, "Az", Type::float_}},
00372     {Field::baz, {30, 30, "BAz", Type::float_}},
00373     {Field::gcarc, {31, 31, "GCArc", Type::float_}},
00374     {Field::depmen, {32, 32, "DepMen", Type::float_}},
00375     {Field::cmpaz, {33, 33, "CmpAz", Type::float_}},
00376     {Field::cmpinc, {34, 34, "CmpInc", Type::float_}},
00377     {Field::xminimum, {35, 35, "XMinimum", Type::float_}},
00378     {Field::xmaximum, {36, 36, "XMaximum", Type::float_}},
00379     {Field::yminimum, {37, 37, "YMinimum", Type::float_}},
00380     {Field::ymaximum, {38, 38, "YMaximum", Type::float_}},
00381     // Doubles
00382     {Field::delta, {39, 0, "Delta", Type::double_}},
00383     {Field::b, {40, 1, "B", Type::double_}},
00384     {Field::e, {41, 2, "E", Type::double_}},
00385     {Field::o, {42, 3, "O", Type::double_}},
00386     {Field::a, {43, 4, "A", Type::double_}},
00387     {Field::t0, {44, 5, "T0", Type::double_}},
00388     {Field::t1, {45, 6, "T1", Type::double_}},
00389     {Field::t2, {46, 7, "T2", Type::double_}},
00390     {Field::t3, {47, 8, "T3", Type::double_}},
00391     {Field::t4, {48, 9, "T4", Type::double_}},
00392     {Field::t5, {49, 10, "T5", Type::double_}},
00393     {Field::t6, {50, 11, "T6", Type::double_}},
00394     {Field::t7, {51, 12, "T7", Type::double_}},
00395     {Field::t8, {52, 13, "T8", Type::double_}},
00396     {Field::t9, {53, 14, "T9", Type::double_}},
00397     {Field::f, {54, 15, "F", Type::double_}},
00398     {Field::stla, {55, 16, "StLa", Type::double_}},
00399     {Field::stlo, {56, 17, "StLo", Type::double_}},
00400     {Field::evla, {57, 18, "EvLa", Type::double_}},
00401     {Field::evlo, {58, 19, "EvLo", Type::double_}},
00402     {Field::sb, {59, 20, "sB", Type::double_}},
00403     {Field::sdelta, {60, 21, "sDelta", Type::double_}},
00404     // Ints
00405     {Field::nzyear, {61, 0, "nzYear", Type::int_}},
00406     {Field::nzjday, {62, 1, "nzJDay", Type::int_}},
00407     {Field::nzhour, {63, 2, "nzHour", Type::int_}},
00408     {Field::nzmin, {64, 3, "nzMin", Type::int_}},
00409     {Field::nzsec, {65, 4, "nzSec", Type::int_}},
00410     {Field::nz msec, {66, 5, "nzMSec", Type::int_}},
00411     {Field::nvhdr, {67, 6, "nVHdr", Type::int_}},
00412     {Field::norid, {68, 7, "nOrID", Type::int_}},
00413     {Field::nevid, {69, 8, "nEvID", Type::int_}},
00414     {Field::npts, {70, 9, "nPts", Type::int_}},
00415     {Field::nsnpts, {71, 10, "nsnPts", Type::int_}},
00416     {Field::nwfid, {72, 11, "nWfID", Type::int_}},
00417     {Field::nxsize, {73, 12, "nXSize", Type::int_}},
00418     {Field::ny size, {74, 13, "nYSize", Type::int_}},
00419     {Field::ifttype, {75, 14, "iFType", Type::int_}},
00420     {Field::idep, {76, 15, "iDep", Type::int_}},
00421     {Field::iztype, {77, 16, "iZType", Type::int_}},
00422     {Field::iinst, {78, 17, "iInst", Type::int_}},
00423     {Field::istreg, {79, 18, "iStReg", Type::int_}},
00424     {Field::ievreg, {80, 19, "iEvReg", Type::int_}},

```

```

00425     {Field::ievtyp, {81, 20, "iEvTyp", Type::int_}},
00426     {Field::iqual, {82, 21, "iQual", Type::int_}},
00427     {Field::isynth, {83, 22, "iSynth", Type::int_}},
00428     {Field::imagtyp, {84, 23, "iMagTyp", Type::int_}},
00429     {Field::imagsrc, {85, 24, "iMagSrc", Type::int_}},
00430     {Field::ibody, {86, 25, "iBody", Type::int_}},
00431     // Bools
00432     {Field::leven, {87, 0, "lEven", Type::bool_}},
00433     {Field::lpspol, {88, 1, "lPsPol", Type::bool_}},
00434     {Field::lovrOK, {89, 2, "lOvrOK", Type::bool_}},
00435     {Field::lcalda, {90, 3, "lCalDA", Type::bool_}},
00436     // Strings
00437     {Field::kstnm, {91, 0, "kStNm", Type::string_}},
00438     {Field::kevn, {92, 1, "kEvNm", Type::string_}},
00439     {Field::khole, {93, 2, "kHole", Type::string_}},
00440     {Field::ko, {94, 3, "kO", Type::string_}},
00441     {Field::ka, {95, 4, "kA", Type::string_}},
00442     {Field::kt0, {96, 5, "kT0", Type::string_}},
00443     {Field::kt1, {97, 6, "kT1", Type::string_}},
00444     {Field::kt2, {98, 7, "kT2", Type::string_}},
00445     {Field::kt3, {99, 8, "kT3", Type::string_}},
00446     {Field::kt4, {100, 9, "kT4", Type::string_}},
00447     {Field::kt5, {101, 10, "kT5", Type::string_}},
00448     {Field::kt6, {102, 11, "kT6", Type::string_}},
00449     {Field::kt7, {103, 12, "kT7", Type::string_}},
00450     {Field::kt8, {104, 13, "kT8", Type::string_}},
00451     {Field::kt9, {105, 14, "kT9", Type::string_}},
00452     {Field::kf, {106, 15, "kF", Type::string_}},
00453     {Field::kuser0, {107, 16, "kUser0", Type::string_}},
00454     {Field::kuser1, {108, 17, "kUser1", Type::string_}},
00455     {Field::kuser2, {109, 18, "kUser2", Type::string_}},
00456     {Field::kcmpnm, {110, 19, "kCmpNm", Type::string_}},
00457     {Field::knetwk, {111, 20, "kNetwk", Type::string_}},
00458     {Field::kdatrd, {112, 21, "kDatRd", Type::string_}},
00459     {Field::kinst, {113, 22, "kInst", Type::string_}},
00460     // Data
00461     {Field::data1, {114, 0, "Data1", Type::int_}},
00462     {Field::data2, {115, 1, "Data2", Type::int_}};

```

#### 6.1.4.2 field\_num

```
const std::unordered_map<size_t, Field> pssp::field_num
```

Map of column number ([Datasheet](#)) to Field.

Given a column in the [Datasheet](#), get the Field (used as a key in another map).

**Todo** Merge into field\_Info

```

00208                                     { // Floats
00209                                     {0, Field::depmin},
00210                                     {1, Field::depmax},
00211                                     {2, Field::odelta},
00212                                     {3, Field::resp0},
00213                                     {4, Field::resp1},
00214                                     {5, Field::resp2},
00215                                     {6, Field::resp3},
00216                                     {7, Field::resp4},
00217                                     {8, Field::resp5},
00218                                     {9, Field::resp6},
00219                                     {10, Field::resp7},
00220                                     {11, Field::resp8},
00221                                     {12, Field::resp9},
00222                                     {13, Field::stel},
00223                                     {14, Field::stdp},
00224                                     {15, Field::evel},
00225                                     {16, Field::evdp},
00226                                     {17, Field::mag},
00227                                     {18, Field::user0},
00228                                     {19, Field::user1},
00229                                     {20, Field::user2},
00230                                     {21, Field::user3},
00231                                     {22, Field::user4},
00232                                     {23, Field::user5},
00233                                     {24, Field::user6},
00234                                     {25, Field::user7},
00235                                     {26, Field::user8},
00236                                     {27, Field::user9},

```

```

00237         {28, Field::dist},
00238         {29, Field::az},
00239         {30, Field::baz},
00240         {31, Field::gcarc},
00241         {32, Field::depmen},
00242         {33, Field::cmpaz},
00243         {34, Field::cmpinc},
00244         {35, Field::xminimum},
00245         {36, Field::xmaximum},
00246         {37, Field::yminimum},
00247         {38, Field::ymaximum},
00248         // Doubles
00249         {39, Field::delta},
00250         {40, Field::b},
00251         {41, Field::e},
00252         {42, Field::o},
00253         {43, Field::a},
00254         {44, Field::t0},
00255         {45, Field::t1},
00256         {46, Field::t2},
00257         {47, Field::t3},
00258         {48, Field::t4},
00259         {49, Field::t5},
00260         {50, Field::t6},
00261         {51, Field::t7},
00262         {52, Field::t8},
00263         {53, Field::t9},
00264         {54, Field::f},
00265         {55, Field::stla},
00266         {56, Field::stlo},
00267         {57, Field::evla},
00268         {58, Field::evlo},
00269         {59, Field::sb},
00270         {60, Field::sdelta},
00271         // Ints
00272         {61, Field::nzyear},
00273         {62, Field::nzjday},
00274         {63, Field::nzhour},
00275         {64, Field::nzmin},
00276         {65, Field::nzsec},
00277         {66, Field::nzmsec},
00278         {67, Field::nvhdr},
00279         {68, Field::norid},
00280         {69, Field::nevid},
00281         {70, Field::npts},
00282         {71, Field::nsnpts},
00283         {72, Field::nwfid},
00284         {73, Field::nxsize},
00285         {74, Field::nysize},
00286         {75, Field::iftype},
00287         {76, Field::idep},
00288         {77, Field::iztype},
00289         {78, Field::iinst},
00290         {79, Field::istreg},
00291         {80, Field::ievreg},
00292         {81, Field::ievtyp},
00293         {82, Field::igual},
00294         {83, Field::isynth},
00295         {84, Field::imagtyp},
00296         {85, Field::imgsrc},
00297         {86, Field::ibody},
00298         // Bools
00299         {87, Field::leven},
00300         {88, Field::lpspol},
00301         {89, Field::lovrok},
00302         {90, Field::lcald},
00303         // Strings
00304         {91, Field::kstnm},
00305         {92, Field::kevn},
00306         {93, Field::khole},
00307         {94, Field::ko},
00308         {95, Field::ka},
00309         {96, Field::kt0},
00310         {97, Field::kt1},
00311         {98, Field::kt2},
00312         {99, Field::kt3},
00313         {100, Field::kt4},
00314         {101, Field::kt5},
00315         {102, Field::kt6},
00316         {103, Field::kt7},
00317         {104, Field::kt8},
00318         {105, Field::kt9},
00319         {106, Field::kf},
00320         {107, Field::kuser0},
00321         {108, Field::kuser1},
00322         {109, Field::kuser2},
00323         {110, Field::kcmpnm},

```

```

00324                                     {111, Field::knetwk},
00325                                     {112, Field::kdatrd},
00326                                     {113, Field::kinst},
00327                                     // Data
00328                                     {114, Field::data1},
00329                                     {115, Field::data2}};

```

### 6.1.4.3 type\_names

```
const std::unordered_map<Type, const std::string> pssp::type_names
```

#### Initial value:

```

{
    {Type::string_, "string"},
    {Type::int_, "int"},
    {Type::float_, "float"},
    {Type::double_, "double"},
    {Type::bool_, "bool"}}

```

Map Type to string-name.

Used to provide labels for the [trace\\_info](#) struct.

**Todo** Move to PsSp/Utility/Constants.hpp

```

00047                                     {
00048     {Type::string_, "string"},
00049     {Type::int_, "int"},
00050     {Type::float_, "float"},
00051     {Type::double_, "double"},
00052     {Type::bool_, "bool"}};

```

## 6.2 pssp::about Namespace Reference

### Variables

- constexpr int [button\\_width](#) {75}  
*Width (pixels) of the [AboutWindow.okay\\_button](#) object.*
- constexpr int [button\\_height](#) {25}  
*Height (pixels) of the [AboutWindow.okay\\_button](#) object.*
- constexpr int [text\\_height](#) {90}  
*Height (pixels) of the [AboutWindow.message](#) object.*
- constexpr int [height](#) {[text\\_height](#) + [button\\_height](#) + 10}  
*Height (pixels) of the [AboutWindow](#).*
- constexpr int [text\\_width](#) {330}  
*Width (pixels) of the [AboutWindow.message](#) object.*
- constexpr int [width](#) {[text\\_width](#) + 50}  
*Width (pixels) of the [AboutWindow](#).*

### 6.2.1 Detailed Description

Constants specific to the [AboutWindow](#).

**Todo** Move this to PsSp/Utility/Constants.hpp

## 6.2.2 Variable Documentation

### 6.2.2.1 button\_height

```
constexpr int pssp::about::button_height {25} [constexpr]
```

Height (pixels) of the [AboutWindow.okay\\_button](#) object.  
00039 {25};

### 6.2.2.2 button\_width

```
constexpr int pssp::about::button_width {75} [constexpr]
```

Width (pixels) of the [AboutWindow.okay\\_button](#) object.  
00037 {75};

### 6.2.2.3 height

```
constexpr int pssp::about::height {text_height + button_height + 10} [constexpr]
```

Height (pixels) of the [AboutWindow](#).  
00043 {text\_height + button\_height + 10};

### 6.2.2.4 text\_height

```
constexpr int pssp::about::text_height {90} [constexpr]
```

Height (pixels) of the [AboutWindow.message](#) object.  
00041 {90};

### 6.2.2.5 text\_width

```
constexpr int pssp::about::text_width {330} [constexpr]
```

Width (pixels) of the [AboutWindow.message](#) object.  
00045 {330};

### 6.2.2.6 width

```
constexpr int pssp::about::width {text_width + 50} [constexpr]
```

Width (pixels) of the [AboutWindow](#).  
00047 {text\_width + 50};

## 6.3 pssp::constants Namespace Reference

### Variables

- constexpr int [sac\\_float](#) {39}  
*Number of float columns for SAC records.*
- constexpr int [sac\\_double](#) {22}  
*Number of double columns for SAC records.*
- constexpr int [sac\\_int](#) {26}  
*Number of integer columns for SAC records.*
- constexpr int [sac\\_bool](#) {4}  
*Number of boolean columns for SAC records.*
- constexpr int [sac\\_string](#) {22 + 1}  
*Number of string columns for SAC records.*
- constexpr int [sac\\_data](#) {2}  
*Number of possible data vectors for a SAC record.*

### 6.3.1 Variable Documentation

#### 6.3.1.1 sac\_bool

```
constexpr int pssp::constants::sac_bool {4} [constexpr]
```

Number of boolean columns for SAC records.

```
00024 {4};
```

#### 6.3.1.2 sac\_data

```
constexpr int pssp::constants::sac_data {2} [constexpr]
```

Number of possible data vectors for a SAC record.

```
00028 {2};
```

#### 6.3.1.3 sac\_double

```
constexpr int pssp::constants::sac_double {22} [constexpr]
```

Number of double columns for SAC records.

```
00020 {22};
```

#### 6.3.1.4 sac\_float

```
constexpr int pssp::constants::sac_float {39} [constexpr]
```

Number of float columns for SAC records.

```
00018 {39};
```

### 6.3.1.5 sac\_int

```
constexpr int pssp::constants::sac_int {26} [constexpr]
```

Number of integer columns for SAC records.

```
00022 {26};
```

### 6.3.1.6 sac\_string

```
constexpr int pssp::constants::sac_string {22 + 1} [constexpr]
```

Number of string columns for SAC records.

```
00026 {22 + 1};
```

## 6.4 pssp::datasheet Namespace Reference

### Classes

- struct [Cell](#)  
*Specify a datasheet cell. This includes placement, size, font, color, box-type, alignment, border-type, and content of a datasheet cell.*
- struct [Spec](#)  
*Used to specify the size of [Datasheet](#) cells.*

### Variables

- constexpr int [font\\_size](#) {14}  
*Font-size in cells.*
- constexpr int [cell\\_buffer](#) {3}  
*Buffer between cell contents region and cell edge (pixels).*
- constexpr int [max\\_chars](#) {10}  
*Maximum number of characters allow in a cell.*
- const std::string [edit\\_chars](#) {"0123456789+-\r\n"}  
*Keys that trigger cell editing.*

### 6.4.1 Detailed Description

Constants and structs specific to the [Datasheet](#).

**Todo** Move constants to PsSp/Utility/Constants.hpp

Move structs to PsSp/Utility/Structs.hpp

### 6.4.2 Variable Documentation

#### 6.4.2.1 cell\_buffer

```
constexpr int pssp::datasheet::cell_buffer {3} [constexpr]
```

Buffer between cell contents region and cell edge (pixels).

```
00086 {3};
```



### 6.4.2.2 edit\_chars

```
const std::string pssp::datasheet::edit_chars {"0123456789+-\r\n"}
```

Keys that trigger cell editing.

```
00090 {"0123456789+-\r\n"};
```

### 6.4.2.3 font\_size

```
constexpr int pssp::datasheet::font_size {14} [constexpr]
```

Font-size in cells.

```
00084 {14};
```

### 6.4.2.4 max\_chars

```
constexpr int pssp::datasheet::max_chars {10} [constexpr]
```

Maximum number of characters allow in a cell.

```
00088 {10};
```

## 6.5 pssp::mw Namespace Reference

### Variables

- constexpr int [minimum\\_x](#) {300}  
*Minimum width of the [MainWindow](#).*
- constexpr int [minimum\\_y](#) {300}  
*Minimum height of the [MainWindow](#).*
- constexpr int [menu\\_height](#) {25}  
*Height of the menubar (Linux/Windows only).*

### 6.5.1 Detailed Description

Constants specific to the [MainWindow](#).

**Todo** Move this to PsSp/Utility/Constants.hpp

### 6.5.2 Variable Documentation

#### 6.5.2.1 menu\_height

```
constexpr int pssp::mw::menu_height {25} [constexpr]
```

Height of the menubar (Linux/Windows only).

```
00054 {25};
```

### 6.5.2.2 minimum\_x

```
constexpr int pssp::mw::minimum_x {300} [constexpr]
```

Minimum width of the [MainWindow](#).

```
00050 {300};
```

### 6.5.2.3 minimum\_y

```
constexpr int pssp::mw::minimum_y {300} [constexpr]
```

Minimum height of the [MainWindow](#).

```
00052 {300};
```

## 6.6 pssp::structs Namespace Reference

### Classes

- struct [Geometry](#)  
*FLTK [Geometry](#) handling struct.*
- struct [Grid](#)  
*FLTK [Grid](#) definition struct.*

## 6.7 pssp::welcome Namespace Reference

### Variables

- constexpr int [button\\_width](#) {125}  
*Width of [WelcomeWindow.continue\\_button](#) (pixels).*
- constexpr int [button\\_height](#) {25}  
*Height of [WelcomeWindow.continue\\_button](#) (pixels).*
- constexpr int [text\\_height](#) {50}  
*Height of [WelcomeWindow.message](#) box (pixels).*
- constexpr int [height](#) {[text\\_height](#) + [button\\_height](#) + 10}  
*Height of [WelcomeWindow](#) (pixels).*
- constexpr int [text\\_width](#) {380}  
*Width of [WelcomeWindow.message](#) box (pixels).*
- constexpr int [width](#) {[text\\_width](#) + 20}  
*Width of [WelcomeWindow](#) (pixels).*

### 6.7.1 Detailed Description

Constants specific to the [WelcomeWindow](#).

**Todo** Move this to PsSp/Utility/Constants.hpp

## 6.7.2 Variable Documentation

### 6.7.2.1 button\_height

```
constexpr int pssp::welcome::button_height {25} [constexpr]
```

Height of [WelcomeWindow.continue\\_button](#) (pixels).  
00038 {25};

### 6.7.2.2 button\_width

```
constexpr int pssp::welcome::button_width {125} [constexpr]
```

Width of [WelcomeWindow.continue\\_button](#) (pixels).  
00036 {125};

### 6.7.2.3 height

```
constexpr int pssp::welcome::height {text_height + button_height + 10} [constexpr]
```

Height of [WelcomeWindow](#) (pixels).  
00042 {text\_height + button\_height + 10};

### 6.7.2.4 text\_height

```
constexpr int pssp::welcome::text_height {50} [constexpr]
```

Height of [WelcomeWindow.message](#) box (pixels).  
00040 {50};

### 6.7.2.5 text\_width

```
constexpr int pssp::welcome::text_width {380} [constexpr]
```

Width of [WelcomeWindow.message](#) box (pixels).  
00044 {380};

### 6.7.2.6 width

```
constexpr int pssp::welcome::width {text_width + 20} [constexpr]
```

Width of [WelcomeWindow](#) (pixels).  
00046 {text\_width + 20};



## Chapter 7

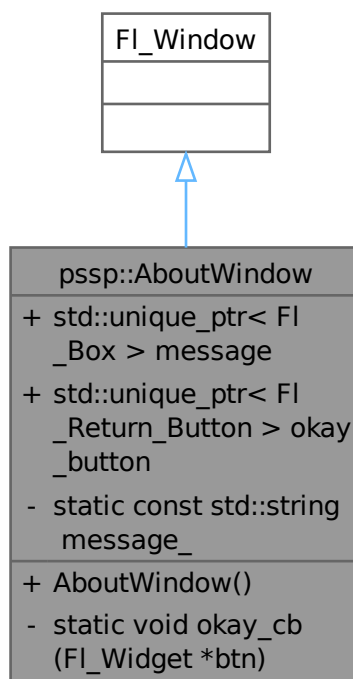
# Class Documentation

### 7.1 pssp::AboutWindow Class Reference

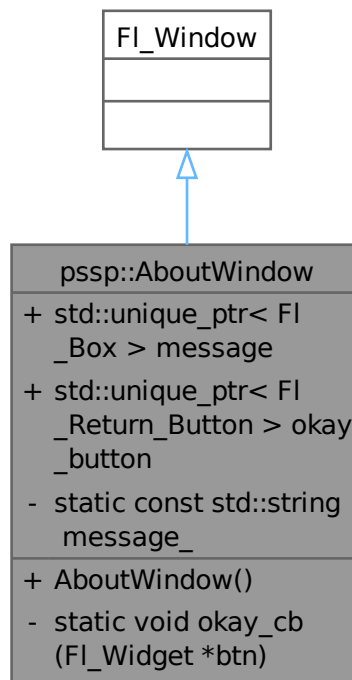
Class to provide the About Window.

```
#include <About.hpp>
```

Inheritance diagram for pssp::AboutWindow:



Collaboration diagram for pssp::AboutWindow:



### Public Member Functions

- [AboutWindow](#) ()  
*AboutWindow* constructor.

### Public Attributes

- `std::unique_ptr< FI_Box >` [message](#) {}
- `std::unique_ptr< FI_Return_Button >` [okay\\_button](#) {}

### Static Private Member Functions

- `static void` [okay\\_cb](#) (FI\_Widget \*btn)  
*AboutWindow.okay\_button* callback.

### Static Private Attributes

- `static const std::string` [message\\_](#)  
*Message to display in the about window.*

### 7.1.1 Detailed Description

Class to provide the About Window.

This provides the about window for the PsSp program.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 AboutWindow()

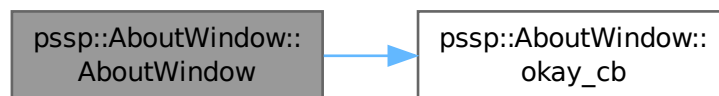
```
pssp::AboutWindow::AboutWindow ( )
```

[AboutWindow](#) constructor.

This creates the [AboutWindow](#) object with all the specified sizes from the [pssp::about](#) namespaces and centers the window.

```
00019             : Fl_Window(0, 0, 0, 0, "About") {
00020   this->begin();
00021   structs::Geometry geo{};
00022   Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
00023   geo.x_pos = ((geo.width - about::width) / 2);
00024   geo.y_pos = ((geo.height - about::height) / 2);
00025   this->resize(geo.x_pos, geo.y_pos, about::width, about::height);
00026   this->box(FL_BORDER_BOX);
00027   set_modal();
00028   message = std::make_unique<Fl_Box>(about::width - about::text_width, 0,
00029                                     about::text_width, about::text_height);
00030   okay_button = std::make_unique<Fl_Return_Button>(
00031       (about::width - about::button_width) / 2, about::text_height,
00032       about::button_width, about::button_height, "Okay");
00033   message->label(message_.c_str());
00034   message->align(FL_ALIGN_CENTER);
00035   okay_button->callback(okay_cb);
00036   this->end();
00037 }
```

Here is the call graph for this function:



### 7.1.3 Member Function Documentation

#### 7.1.3.1 okay\_cb()

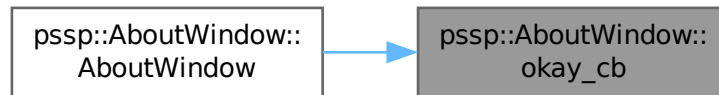
```
void pssp::AboutWindow::okay_cb (
    Fl_Widget * btn ) [static], [private]
```

[AboutWindow.okay\\_button](#) callback.

When the user choses to close the About window, the button tells the parent to hide (how FLTK handles closing a window).

```
00045 { btn->parent()->hide(); }
```

Here is the caller graph for this function:



## 7.1.4 Member Data Documentation

### 7.1.4.1 message

```
std::unique_ptr<Fl_Box> pssp::AboutWindow::message {}
00059 {};
```

### 7.1.4.2 message\_

```
const std::string pssp::AboutWindow::message_ [inline], [static], [private]
```

#### Initial value:

```
{ "Website: https://arbCoding.github.io/PsSp/\n"
  "GitHub: https://arbCoding.github.com/PsSp\n"
  "Developer: Alexander R. Blanchette <arbCoding@gmail.com>"
  "License: MIT" }
```

Message to display in the about window.

```
00067 { "Website: https://arbCoding.github.io/PsSp/\n"
00068   "GitHub: https://arbCoding.github.com/PsSp\n"
00069   "Developer: Alexander R. Blanchette <arbCoding@gmail.com>"
00070   "License: MIT" };
```

### 7.1.4.3 okay\_button

```
std::unique_ptr<Fl_Return_Button> pssp::AboutWindow::okay_button {}
00060 {};
```

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

- include/PsSp/Windows/About.hpp
- src/Windows/About.cpp

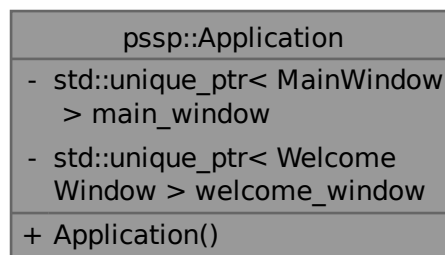


## 7.2 pssp::Application Class Reference

Main application class.

```
#include <Application.hpp>
```

Collaboration diagram for pssp::Application:



### Public Member Functions

- [Application \(\)](#)  
*Application constructor.*

### Private Attributes

- std::unique\_ptr< [MainWindow](#) > [main\\_window](#) {}  
*Unique Pointer to the Main\_Window object.*
- std::unique\_ptr< [WelcomeWindow](#) > [welcome\\_window](#) {}  
*Unique Pointer to the Welcome\_Window object.*

### 7.2.1 Detailed Description

Main application class.

This manages the application (created in main()).

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 Application()

```
pssp::Application::Application ( )
```

[Application](#) constructor.

Creates the main\_window object and the welcome\_window object.

Logs status after creation.

```
00020         {
00021     main_window = std::make_unique<MainWindow>();
00022     main_window->show();
00023     welcome_window = std::make_unique<WelcomeWindow>();
00024     welcome_window->show();
00025     spdlog::trace("Application ready.");
00026 }
```

## 7.2.3 Member Data Documentation

### 7.2.3.1 main\_window

```
std::unique_ptr<MainWindow> pssp::Application::main_window {} [private]
```

Unique Pointer to the Main\_Window object.  
00038 {};

### 7.2.3.2 welcome\_window

```
std::unique_ptr<WelcomeWindow> pssp::Application::welcome_window {} [private]
```

Unique Pointer to the Welcome\_Window object.  
00040 {};

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

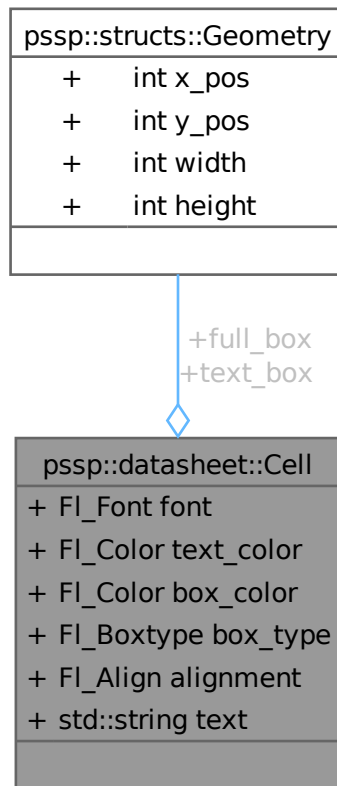
- include/PsSp/Application/Application.hpp
- src/Application/Application.cpp

## 7.3 pssp::datasheet::Cell Struct Reference

Specify a datasheet cell. This includes placement, size, font, color, box-type, alignment, border-type, and content of a datasheet cell.

```
#include <Datasheet.hpp>
```

Collaboration diagram for pssp::datasheet::Cell:



### Public Attributes

- [structs::Geometry full\\_box](#) {}  
*Geometry of [Cell](#) edges.*
- [structs::Geometry text\\_box](#) {}  
*Geometry of [Cell](#) content (internal to full\_box).*
- FI\_Font [font](#) {FL\_HELVETICA}  
*Font used for [Cell](#) content.*
- FI\_Color [text\\_color](#) {FL\_BLACK}  
*Color of [Cell](#) text.*
- FI\_Color [box\\_color](#) {FL\_GRAY}  
*Color of cell background.*
- FI\_Boxtype [box\\_type](#) {FL\_THIN\_UP\_BOX}  
*Type of cell drawing.*
- FI\_Align [alignment](#) {FL\_ALIGN\_CENTER}  
*Alignment of cell contents.*
- std::string [text](#) {}  
*String of cell contents.*

### 7.3.1 Detailed Description

Specify a datasheet cell. This includes placement, size, font, color, box-type, alignment, border-type, and content of a datasheet cell.

### 7.3.2 Member Data Documentation

#### 7.3.2.1 alignment

```
Fl_Align pssp::datasheet::Cell::alignment {FL_ALIGN_CENTER}
```

Alignment of cell contents.

```
00107 {FL_ALIGN_CENTER};
```

#### 7.3.2.2 box\_color

```
Fl_Color pssp::datasheet::Cell::box_color {FL_GRAY}
```

Color of cell background.

```
00105 {FL_GRAY};
```

#### 7.3.2.3 box\_type

```
Fl_Boxtype pssp::datasheet::Cell::box_type {FL_THIN_UP_BOX}
```

Type of cell drawing.

```
00106 {FL_THIN_UP_BOX};
```

#### 7.3.2.4 font

```
Fl_Font pssp::datasheet::Cell::font {FL_HELVETICA}
```

Font used for [Cell](#) content.

```
00103 {FL_HELVETICA};
```

#### 7.3.2.5 full\_box

```
structs::Geometry pssp::datasheet::Cell::full_box {}
```

Geometry of [Cell](#) edges.

```
00099 {};
```

#### 7.3.2.6 text

```
std::string pssp::datasheet::Cell::text {}
```

String of cell contents.

```
00109 {};
```

## 7.3.2.7 text\_box

```
structs::Geometry pssp::datasheet::Cell::text_box {}
```

Geometry of [Cell](#) content (internal to full\_box).

```
00102 {};
```

## 7.3.2.8 text\_color

```
Fl_Color pssp::datasheet::Cell::text_color {FL_BLACK}
```

Color of [Cell](#) text.

```
00104 {FL_BLACK};
```

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

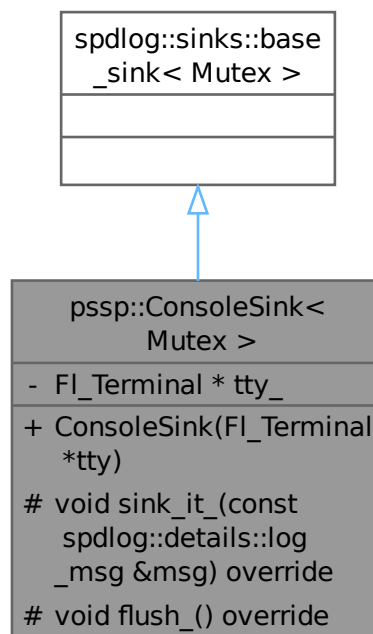
- include/PsSp/Widgets/Datasheet.hpp

## 7.4 pssp::ConsoleSink&lt; Mutex &gt; Class Template Reference

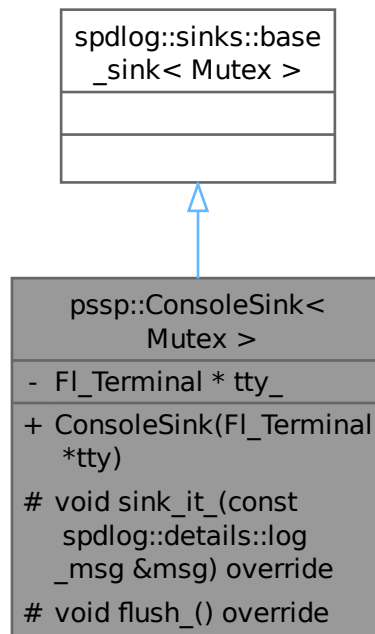
Sink (receiver) of log messages for PsSp console.

```
#include <ConsoleSink.hpp>
```

Inheritance diagram for pssp::ConsoleSink< Mutex >:



Collaboration diagram for `pssp::ConsoleSink< Mutex >`:



### Public Member Functions

- `ConsoleSink` (`FI_Terminal *tty`)  
*Default constructor.*

### Protected Member Functions

- `void sink_it_` (`const spdlog::details::log_msg &msg`) override  
*Receives message from spdlog and then passes message to display console.*
- `void flush_` () override  
*Clear (flush) FI\_Terminal.*

### Private Attributes

- `FI_Terminal * tty_` {}  
*Message receiver (console/terminal/tty).*

## 7.4.1 Detailed Description

```
template<typename Mutex>
class pssp::ConsoleSink< Mutex >
```

Sink (receiver) of log messages for PsSp console.

This class receiver logs from spdlog and passes them on to a FLTK terminal (`FL_Terminal`) object for presentation.

**Todo** At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

## 7.4.2 Constructor & Destructor Documentation

### 7.4.2.1 ConsoleSink()

```
template<typename Mutex >
pssp::ConsoleSink< Mutex >::ConsoleSink (
    Fl_Terminal * tty ) [inline], [explicit]
```

Default constructor.

#### Parameters

in	tty	Fl_Terminal* FLTK Terminal widget that will display the logs.
----	-----	---------------------------------------------------------------

```
00053 { tty_ = tty; }
```

## 7.4.3 Member Function Documentation

### 7.4.3.1 flush\_()

```
template<typename Mutex >
void pssp::ConsoleSink< Mutex >::flush_ ( ) [inline], [override], [protected]
```

Clear (flush) Fl\_Terminal.

```
00072 { tty_>clear(); }
```

### 7.4.3.2 sink\_it\_()

```
template<typename Mutex >
void pssp::ConsoleSink< Mutex >::sink_it_ (
    const spdlog::details::log_msg & msg ) [inline], [override], [protected]
```

Receives message from spdlog and then passes message to display console.

#### Parameters

in	msg	spdlog::details::log_msg& Message to format and pass.
----	-----	-------------------------------------------------------

```
00062 {
00063     // log_msg is a struct containing the log entry info like level, timestamp,
00064     // msg.raw contains the pre-formatted log
00065
00066     // If needed (very likely, but not madatory), the sink formats the message
00067     spdlog::memory_buf_t formatted{};
00068     spdlog::sinks::base_sink<Mutex>::formatter->format(msg, formatted);
00069     tty_>append(fmt::to_string(formatted).c_str());
00070 }
```

## 7.4.4 Member Data Documentation

### 7.4.4.1 tty\_

```
template<typename Mutex >
Fl_Terminal* pssp::ConsoleSink< Mutex >::tty_ {} [private]
```

Message receiver (console/terminal/tty).

```
00075 {};
```

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

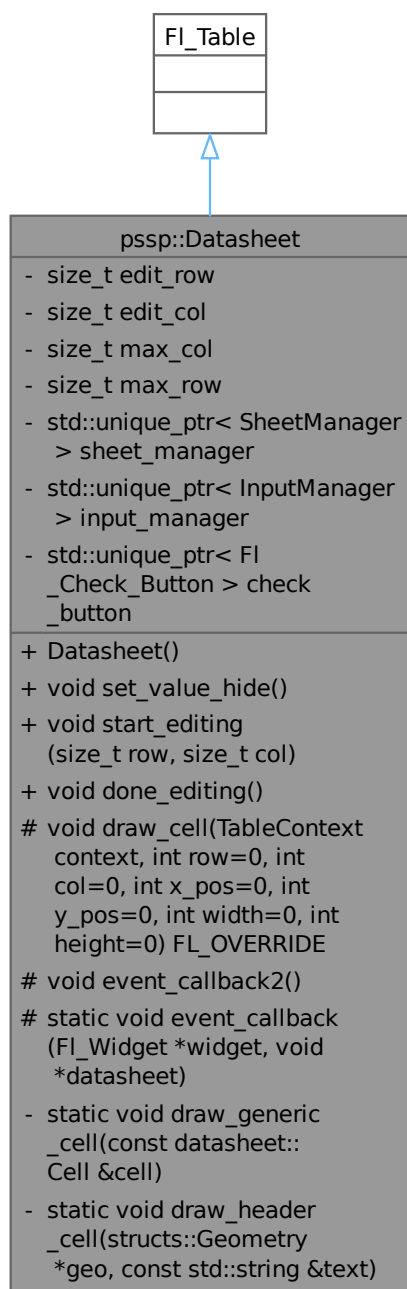
- include/PsSp/Logging/ConsoleSink.hpp

## 7.5 pssp::Datasheet Class Reference

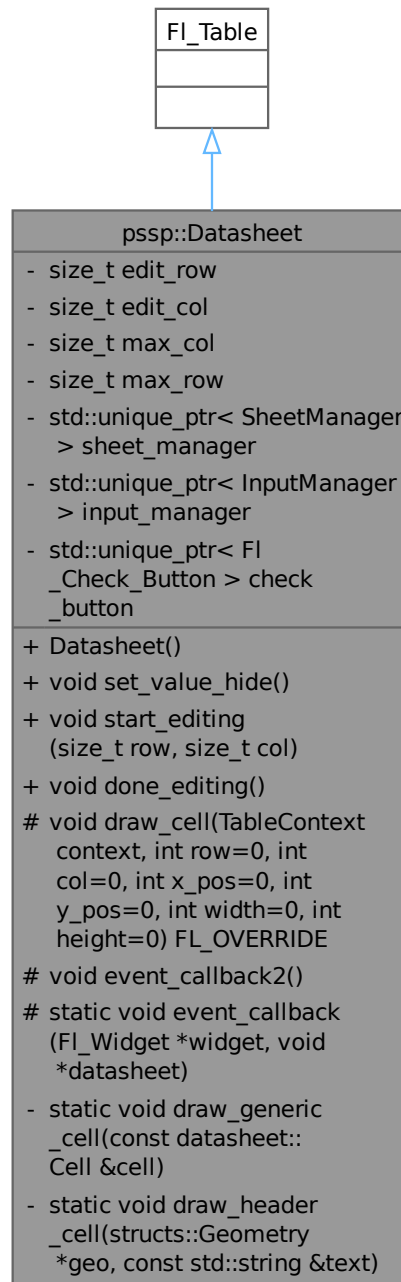
```
#include <Datasheet.hpp>
```



Inheritance diagram for pssp::Datasheet:



Collaboration diagram for pssp::Datasheet:



## Public Member Functions

- [Datasheet \(\)](#)  
*Datasheet constructor.*
- void [set\\_value\\_hide \(\)](#)
- void [start\\_editing](#) (size\_t row, size\_t col)
- void [done\\_editing \(\)](#)

### Protected Member Functions

- void [draw\\_cell](#) (TableContext context, int row=0, int col=0, int x\_pos=0, int y\_pos=0, int width=0, int height=0) FL\_OVERRIDE
- void [event\\_callback2](#) ()

### Static Protected Member Functions

- static void [event\\_callback](#) (Fl\_Widget \*widget, void \*datasheet)

### Static Private Member Functions

- static void [draw\\_generic\\_cell](#) (const [datasheet::Cell](#) &cell)
- static void [draw\\_header\\_cell](#) ([structs::Geometry](#) \*geo, const std::string &text)

### Private Attributes

- size\_t [edit\\_row](#) {0}  
*Row of most recently edited cell.*
- size\_t [edit\\_col](#) {0}  
*Column of most recently edited cell.*
- size\_t [max\\_col](#) {0}  
*Maximum number of columns in the [Datasheet](#).*
- size\_t [max\\_row](#) {0}  
*Maximum number of rows in the [Datasheet](#).*
- std::unique\_ptr< [SheetManager](#) > [sheet\\_manager](#) {}  
*[SheetManager](#).*
- std::unique\_ptr< [InputManager](#) > [input\\_manager](#) {}  
*[InputManager](#).*
- std::unique\_ptr< Fl\_Check\_Button > [check\\_button](#) {}  
*Boolean toggle (not implemented).*

## 7.5.1 Constructor & Destructor Documentation

### 7.5.1.1 Datasheet()

pssp::Datasheet::Datasheet ( )

[Datasheet](#) constructor.

Builds the datasheet using the constants from the [pssp::datasheet](#) namespace.

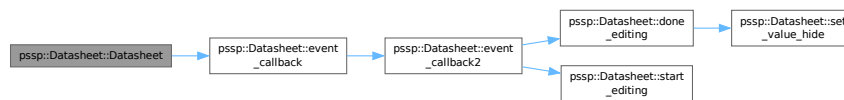
```
00018         : Fl_Table(0, 0, 0, 0) {
00019     spdlog::trace("Making \033[1mDatasheet\033[0m.");
00020     // trick to use event_callback2
00021     callback(&event_callback, reinterpret_cast<void *>(this));
00022     this->begin();
00023     this->when(static_cast<uchar>(FL_WHEN_NOT_CHANGED | this->when()));
00024     input_manager = std::make_unique<InputManager>();
00025     this->tab_cell_nav(1); // enable tab navigation
00026     tooltip("Use keyboard to navigate cells:\n"
00027             "Arrow keys or Tab/Shift-Tab");
00028     sheet_manager = std::make_unique<SheetManager>();
00029     check_button = std::make_unique<Fl_Check_Button>(0, 0, 0, 0);
00030     check_button->hide();
00031     max_col = static_cast<size_t>(sheet_manager->cols());
```

```

00032     max_row = static_cast<size_t>(sheet_manager->rows());
00033     constexpr datasheet::Spec spec{25, 25, 25, 70};
00034     row_header(1);
00035     row_header_width(spec.header_width);
00036     row_height_all(spec.height);
00037     rows(static_cast<int>(max_row));
00038     col_header(1);
00039     col_header_height(spec.header_height);
00040     col_width_all(spec.width);
00041     cols(static_cast<int>(max_col));
00042     row_resize(1);
00043     col_resize(1);
00044     set_selection(0, 0, 0, 0);
00045     this->end();
00046     spdlog::trace("Done making \033[1mDdatasheet\033[0m.");
00047 }

```

Here is the call graph for this function:



## 7.5.2 Member Function Documentation

### 7.5.2.1 done\_editing()

```

void pssp::Datasheet::done_editing ( )
00132 {
00133     if (input_manager->visible() || input_manager->modified) {
00134         set_value_hide();
00135         edit_row = 0;
00136         edit_col = 0;
00137     }
00138 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



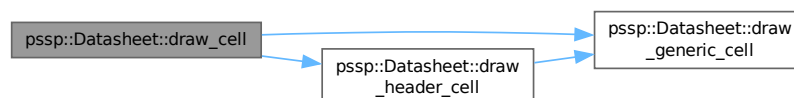
## 7.5.2.2 draw\_cell()

```

void pssp::Datasheet::draw_cell (
    TableContext context,
    int row = 0,
    int col = 0,
    int x_pos = 0,
    int y_pos = 0,
    int width = 0,
    int height = 0 ) [protected]
{
00171
00172     // NOLINTEND(bugprone-easily-swappable-parameters)
00173     switch (context) {
00174     case CONTEXT_COL_HEADER: {
00175         structs::Geometry geo{x_pos, y_pos, width, height};
00176         draw_header_cell(
00177             &geo, field_info.at(field_num.at(static_cast<size_t>(col))).name);
00178     } break;
00179     case CONTEXT_ROW_HEADER: {
00180         structs::Geometry geo{x_pos, y_pos, width, height};
00181         draw_header_cell(&geo, std::to_string(row + 1));
00182     } break;
00183     case CONTEXT_CELL: {
00184         // This needs to be refactored
00185         datasheet::Cell cell{};
00186         cell.full_box = {x_pos, y_pos, width, height};
00187         cell.text_box = {x_pos + datasheet::cell_buffer,
00188             y_pos + datasheet::cell_buffer,
00189             width - (2 * datasheet::cell_buffer),
00190             height - (2 * datasheet::cell_buffer)};
00191         const Field &field{field_num.at(static_cast<size_t>(col))};
00192         const trace_info &info{field_info.at(field)};
00193         if (info.type == Type::string_) {
00194             cell.text = sheet_manager->get_string(static_cast<size_t>(row), field);
00195         } else if (info.type == Type::int_) {
00196             std::ostringstream oss{};
00197             oss << sheet_manager->get_int(static_cast<size_t>(row), field);
00198             cell.text = oss.str();
00199         } else if (info.type == Type::float_) {
00200             std::ostringstream oss{};
00201             oss << sheet_manager->get_float(static_cast<size_t>(row), field);
00202             cell.text = oss.str();
00203         } else if (info.type == Type::double_) {
00204             std::ostringstream oss{};
00205             oss << sheet_manager->get_double(static_cast<size_t>(row), field);
00206             cell.text = oss.str();
00207         } else if (info.type == Type::bool_) {
00208             std::ostringstream oss{};
00209             oss << sheet_manager->get_bool(static_cast<size_t>(row), field);
00210             cell.text = oss.str();
00211         }
00212         cell.box_color = ((is_selected(row, col) != 0) ? FL_YELLOW : FL_WHITE);
00213         cell.alignment = FL_ALIGN_RIGHT;
00214         draw_generic_cell(cell);
00215     } break;
00216     default:
00217         return;
00218     }
00219 }

```

Here is the call graph for this function:



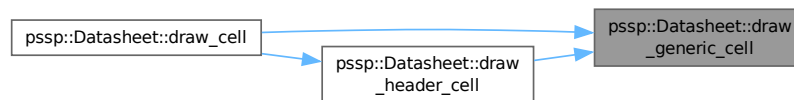
### 7.5.2.3 draw\_generic\_cell()

```

void pssp::Datasheet::draw_generic_cell (
    const datasheet::Cell & cell ) [static], [private]
{
00140     fl_font(cell.font, datasheet::font_size);
00141     fl_draw_box(cell.box_type, cell.full_box.x_pos, cell.full_box.y_pos,
00142               cell.full_box.width, cell.full_box.height, cell.box_color);
00143     fl_push_clip(cell.text_box.x_pos, cell.text_box.y_pos, cell.text_box.width,
00144               cell.text_box.height);
00145     fl_color(cell.text_color);
00146     fl_draw(cell.text.c_str(), cell.text_box.x_pos, cell.text_box.y_pos,
00147             cell.text_box.width, cell.text_box.height, cell.alignment);
00148     fl_pop_clip();
00149 }
00150

```

Here is the caller graph for this function:



### 7.5.2.4 draw\_header\_cell()

```

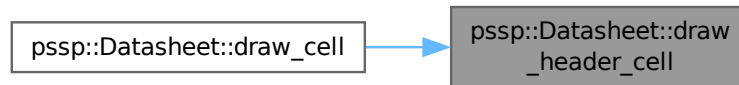
void pssp::Datasheet::draw_header_cell (
    structs::Geometry * geo,
    const std::string & text ) [static], [private]
{
00153     datasheet::Cell cell{};
00154     cell.full_box = *geo;
00155     cell.text_box = cell.full_box;
00156     cell.font = FL_HELVETICA | FL_BOLD;
00157     cell.text = text;
00158     draw_generic_cell(cell);
00159 }
00160

```

Here is the call graph for this function:



Here is the caller graph for this function:

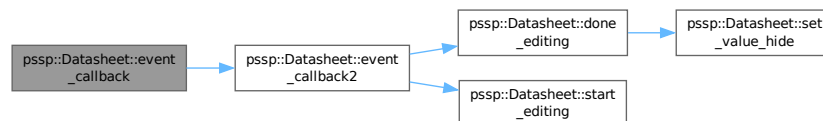


### 7.5.2.5 event\_callback()

```

static void pssp::Datasheet::event_callback (
    Fl_Widget * widget,
    void * datasheet ) [inline], [static], [protected]
00145 {
00146     (void)widget;
00147     reinterpret_cast<Datasheet *>(datasheet)->event_callback2();
00148 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.5.2.6 event\_callback2()

```

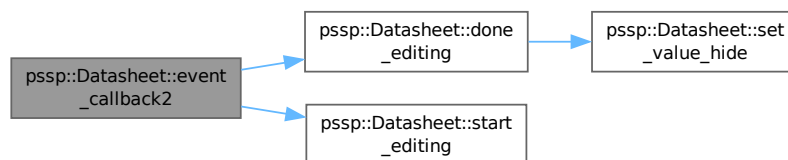
void pssp::Datasheet::event_callback2 ( ) [protected]
00221 {
00222     int row{callback_row()};
00223     int col{callback_col()};
00224     TableContext context{callback_context()};
00225     switch (context) {
00226     case CONTEXT_CELL: {
  
```

```

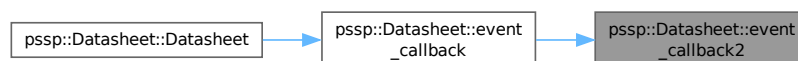
00227     switch (Fl::event()) {
00228     case FL_PUSH:
00229         start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
00230         break;
00231     case FL_KEYBOARD:
00232         done_editing();
00233         if (Fl::event_state() == FL_COMMAND) {
00234             parent()->take_focus();
00235         } else if (datasheet::edit_chars.find(Fl::e_text[0]) !=
00236             std::string::npos) {
00237             start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
00238         }
00239         break;
00240     default:
00241         break;
00242     }
00243 } break;
00244 case CONTEXT_TABLE:
00245 case CONTEXT_ROW_HEADER:
00246 case CONTEXT_COL_HEADER:
00247     done_editing();
00248     break;
00249 default:
00250     return;
00251 }
00252 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.5.2.7 set\_value\_hide()

```

void pssp::Datasheet::set_value_hide ( )
00057 {
00058     const Field &field{field_num.at(edit_col)};
00059     const trace_info &info{field_info.at(field)};
00060     switch (info.type) {
00061     case Type::string_:
00062         sheet_manager->set(edit_row, field, input_manager->value());
00063         break;
00064     case Type::int_:
00065         if (!input_manager->value().empty()) {
00066             sheet_manager->set(edit_row, field, std::stoi(input_manager->value()));
00067         } else {
00068             sheet_manager->set(edit_row, field, 0);
00069         }
00070         break;
00071     case Type::float_:

```

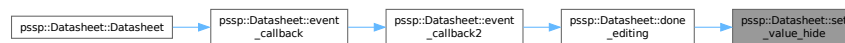


```

00072     if (!input_manager->value().empty()) {
00073         sheet_manager->set(edit_row, field, std::stof(input_manager->value()));
00074     } else {
00075         sheet_manager->set(edit_row, field, 0.0F);
00076     }
00077     break;
00078 case Type::double_:
00079     if (!input_manager->value().empty()) {
00080         sheet_manager->set(edit_row, field, std::stod(input_manager->value()));
00081     } else {
00082         sheet_manager->set(edit_row, field, 0.0);
00083     }
00084     break;
00085 case Type::bool_:
00086     // This is just junk for prototyping
00087     sheet_manager->set(edit_row, field, !input_manager->value().empty());
00088     break;
00089 default:
00090     break;
00091 }
00092 input_manager->cleanup();
00093 input_manager->modified = false;
00094 window()->cursor(FL_CURSOR_DEFAULT); // deals with disappearing cursor
00095 }

```

Here is the caller graph for this function:



### 7.5.2.8 start\_editing()

```

void pssp::Datasheet::start_editing (
    size_t row,
    size_t col )
{
00099
00100     edit_row = row;
00101     edit_col = col;
00102     set_selection(static_cast<int>(row), static_cast<int>(col),
00103                 static_cast<int>(row), static_cast<int>(col));
00104     structs::Geometry geo{};
00105     find_cell(CONTEXT_CELL, static_cast<int>(row), static_cast<int>(col),
00106             geo.x_pos, geo.y_pos, geo.width, geo.height);
00107     // Need to refactor
00108     const Field &field{field_num.at(col)};
00109     const trace_info &info{field_info.at(field)};
00110     if (info.type == Type::string_) {
00111         input_manager->start_editing(info, geo,
00112             sheet_manager->get_string(row, field));
00113     } else if (info.type == Type::int_) {
00114         std::ostringstream oss{};
00115         oss << sheet_manager->get_int(row, field);
00116         input_manager->start_editing(info, geo, oss.str());
00117     } else if (info.type == Type::float_) {
00118         std::ostringstream oss{};
00119         oss << sheet_manager->get_float(row, field);
00120         input_manager->start_editing(info, geo, oss.str());
00121     } else if (info.type == Type::double_) {
00122         std::ostringstream oss{};
00123         oss << sheet_manager->get_double(row, field);
00124         input_manager->start_editing(info, geo, oss.str());
00125     } else if (info.type == Type::bool_) {
00126         std::ostringstream oss{};
00127         oss << sheet_manager->get_bool(row, field);
00128         input_manager->start_editing(info, geo, oss.str());
00129     }
00130 }

```

Here is the caller graph for this function:



## 7.5.3 Member Data Documentation

### 7.5.3.1 check\_button

```
std::unique_ptr<Fl_Check_Button> pssp::Datasheet::check_button {} [private]
```

Boolean toggle (not implemented).

```
00162 {};
```

### 7.5.3.2 edit\_col

```
size_t pssp::Datasheet::edit_col {0} [private]
```

Column of most recently edited cell.

```
00154 {};
```

### 7.5.3.3 edit\_row

```
size_t pssp::Datasheet::edit_row {0} [private]
```

Row of most recently edited cell.

```
00152 {};
```

### 7.5.3.4 input\_manager

```
std::unique_ptr<InputManager> pssp::Datasheet::input_manager {} [private]
```

[InputManager](#).

```
00160 {};
```

### 7.5.3.5 max\_col

```
size_t pssp::Datasheet::max_col {0} [private]
```

Maximum number of columns in the [Datasheet](#).

```
00156 {};
```

### 7.5.3.6 max\_row

```
size_t pssp::Datasheet::max_row {0} [private]
```

Maximum number of rows in the [Datasheet](#).

```
00158 {};
```

### 7.5.3.7 sheet\_manager

```
std::unique_ptr<SheetManager> pssp::Datasheet::sheet_manager {} [private]
```

[SheetManager](#).

```
00159 {};
```

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

- include/PsSp/Widgets/Datasheet.hpp
- src/Widgets/Datasheet.cpp

## 7.6 pssp::structs::Geometry Struct Reference

FLTK [Geometry](#) handling struct.

```
#include <Structs.hpp>
```

Collaboration diagram for pssp::structs::Geometry:

pssp::structs::Geometry	
+	int x_pos
+	int y_pos
+	int width
+	int height

### Public Attributes

- int [x\\_pos](#) {0}  
*Left-most position of FLTK object.*
- int [y\\_pos](#) {0}  
*Upper-most position of FLTK object.*
- int [width](#) {0}  
*Width of FLTK object.*
- int [height](#) {0}  
*Height of FLTK object.*

### 7.6.1 Detailed Description

FLTK [Geometry](#) handling struct.

This struct simplifies passing parameters to FLTK drawing functions (instead of passing four loose integers).

## 7.6.2 Member Data Documentation

### 7.6.2.1 height

```
int pssp::structs::Geometry::height {0}
```

Height of FLTK object.

```
00037 {0};
```

### 7.6.2.2 width

```
int pssp::structs::Geometry::width {0}
```

Width of FLTK object.

```
00035 {0};
```

### 7.6.2.3 x\_pos

```
int pssp::structs::Geometry::x_pos {0}
```

Left-most position of FLTK object.

```
00031 {0};
```

### 7.6.2.4 y\_pos

```
int pssp::structs::Geometry::y_pos {0}
```

Upper-most position of FLTK object.

```
00033 {0};
```

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

- include/PsSp/Utility/Structs.hpp

## 7.7 pssp::structs::Grid Struct Reference

FLTK [Grid](#) definition struct.

```
#include <Structs.hpp>
```

Collaboration diagram for pssp::structs::Grid:

pssp::structs::Grid
<ul style="list-style-type: none"><li>+ int row</li><li>+ int col</li><li>+ int row_span</li><li>+ int col_span</li></ul>

## Public Attributes

- int [row](#) {0}  
*First row (top-most row) of grid position.*
- int [col](#) {0}  
*First column (left-most column) of grid position.*
- int [row\\_span](#) {0}  
*Width (in rows) of object.*
- int [col\\_span](#) {0}  
*Height (in columns) of object.*

### 7.7.1 Detailed Description

FLTK [Grid](#) definition struct.

This struct makes it easy to define objects in an FLTK grid (FI\_Grid). Used in Windows/Main.cpp to define the layout of the [MainWindow](#).

### 7.7.2 Member Data Documentation

#### 7.7.2.1 col

```
int pssp::structs::Grid::col {0}
```

First column (left-most column) of grid position.  
00052 {0};

#### 7.7.2.2 col\_span

```
int pssp::structs::Grid::col_span {0}
```

Height (in columns) of object.  
00056 {0};

#### 7.7.2.3 row

```
int pssp::structs::Grid::row {0}
```

First row (top-most row) of grid position.  
00050 {0};

#### 7.7.2.4 row\_span

```
int pssp::structs::Grid::row_span {0}
```

Width (in rows) of object.  
00054 {0};

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

- include/PsSp/Utility/Structs.hpp

## 7.8 pssp::InputManager Class Reference

Manager of user-input.

```
#include <InputManager.hpp>
```

Collaboration diagram for pssp::InputManager:

pssp::InputManager
<ul style="list-style-type: none"> <li>+ bool modified</li> <li>- std::unique_ptr&lt; FI _Input &gt; input_string</li> <li>- std::unique_ptr&lt; FI _Int_Input &gt; input_int</li> <li>- std::unique_ptr&lt; FI _Float_Input &gt; input_float</li> </ul>
<ul style="list-style-type: none"> <li>+ InputManager()</li> <li>+ std::string value()</li> <li>+ void start_editing (const trace_info &amp;info, const structs::Geometry &amp;geo, const std::string &amp;input)</li> <li>+ void done_editing()</li> <li>+ bool visible() const</li> <li>+ void hide()</li> <li>+ void cleanup()</li> <li>+ static void input_cb (FI_Widget *widget, void *input_manager)</li> <li>- void clear()</li> </ul>

### Public Member Functions

- [InputManager](#) ()
- [std::string value](#) ()
- void [start\\_editing](#) (const [trace\\_info](#) &info, const [structs::Geometry](#) &geo, const std::string &input)
- void [done\\_editing](#) ()
- bool [visible](#) () const
- void [hide](#) ()
- void [cleanup](#) ()

### Static Public Member Functions

- static void [input\\_cb](#) (Fl\_Widget \*widget, void \*input\_manager)

### Public Attributes

- bool [modified](#) {false}

### Private Member Functions

- void [clear](#) ()

### Private Attributes

- std::unique\_ptr< Fl\_Input > [input\\_string](#) {}
- std::unique\_ptr< Fl\_Int\_Input > [input\\_int](#) {}
- std::unique\_ptr< Fl\_Float\_Input > [input\\_float](#) {}

## 7.8.1 Detailed Description

Manager of user-input.

This class handles taking input from the user (text/numerical) that is destined to enter the [Datasheet](#) spreadsheet display (and the underlying data-arrays).

It is designed to handle generic string input, integer input, and float input.

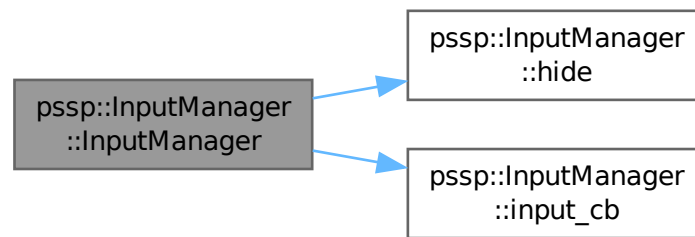
### Todo

## 7.8.2 Constructor & Destructor Documentation

### 7.8.2.1 InputManager()

```
pssp::InputManager::InputManager ( )
00006     {
00007     input\_string = std::make_unique<Fl_Input>(0, 0, 0, 0);
00008     input\_int = std::make_unique<Fl_Int_Input>(0, 0, 0, 0);
00009     input\_float = std::make_unique<Fl_Float_Input>(0, 0, 0, 0);
00010     hide();
00011     input\_string->callback(input\_cb, reinterpret\_cast<void *>(this));
00012     input\_int->callback(input\_cb, reinterpret\_cast<void *>(this));
00013     input\_float->callback(input\_cb, reinterpret\_cast<void *>(this));
00014     input\_string->when(FL_WHEN_ENTER_KEY_ALWAYS);
00015     input\_int->when(FL_WHEN_ENTER_KEY_ALWAYS);
00016     input\_float->when(FL_WHEN_ENTER_KEY_ALWAYS);
00017     input\_string->maximum_size(40);
00018     input\_int->maximum_size(40);
00019     input\_float->maximum_size(40);
00020     input\_string->color(FL_YELLOW);
00021     input\_int->color(FL_RED);
00022     input\_float->color(FL_GREEN);
00023 }
```

Here is the call graph for this function:



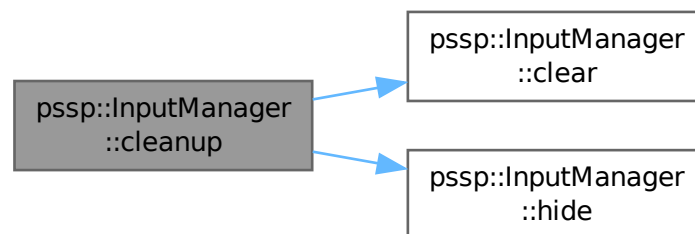
## 7.8.3 Member Function Documentation

### 7.8.3.1 `cleanup()`

```

void pssp::InputManager::cleanup ( )
00030     {
00031     clear();
00032     hide();
00033 }
  
```

Here is the call graph for this function:



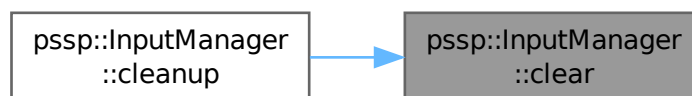
### 7.8.3.2 `clear()`

```

void pssp::InputManager::clear ( ) [private]
00041     {
00042     input_string->value("");
00043     input_int->value("");
00044     input_float->value("");
00045 }
  
```



Here is the caller graph for this function:



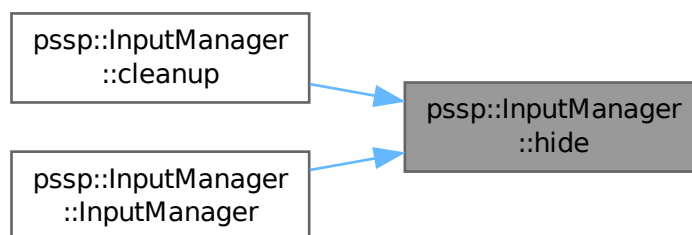
### 7.8.3.3 done\_editing()

```
void pssp::InputManager::done_editing ( )
```

### 7.8.3.4 hide()

```
void pssp::InputManager::hide ( )
00035 {
00036     input_string->hide();
00037     input_int->hide();
00038     input_float->hide();
00039 }
```

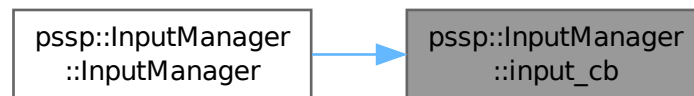
Here is the caller graph for this function:



### 7.8.3.5 input\_cb()

```
static void pssp::InputManager::input_cb (
    Fl_Widget * widget,
    void * input_manager ) [inline], [static]
00053 {
00054     (void)widget;
00055     reinterpret_cast<InputManager*>(input_manager)->modified = true;
00056 }
```

Here is the caller graph for this function:



### 7.8.3.6 start\_editing()

```

void pssp::InputManager::start_editing (
    const trace_info & info,
    const structs::Geometry & geo,
    const std::string & input )
{
00063
00064     if (info.type == Type::string_) {
00065         input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00066         input_string->value(input.c_str());
00067         input_string->insert_position(0, static_cast<int>(input.size()));
00068         input_string->show();
00069         input_string->take_focus();
00070     } else if (info.type == Type::int_) {
00071         input_int->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00072         input_int->value(input.c_str());
00073         input_int->insert_position(0, static_cast<int>(input.size()));
00074         input_int->show();
00075         input_int->take_focus();
00076     } else if (info.type == Type::float_) {
00077         input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00078         input_float->value(input.c_str());
00079         input_float->insert_position(0, static_cast<int>(input.size()));
00080         input_float->show();
00081         input_float->take_focus();
00082     } else if (info.type == Type::double_) {
00083         input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00084         input_float->value(input.c_str());
00085         input_float->insert_position(0, static_cast<int>(input.size()));
00086         input_float->show();
00087         input_float->take_focus();
00088     } else if (info.type == Type::bool_) {
00089         input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00090         input_string->value(input.c_str());
00091         input_string->insert_position(0, static_cast<int>(input.size()));
00092         input_string->show();
00093         input_string->take_focus();
00094     }
00095 }
  
```

### 7.8.3.7 value()

```

std::string pssp::InputManager::value ( )
00047 {
00048     std::string result{};
00049     // Which one is being used? They're empty after cleanup
00050     // so only the used one is full
00051     if (!std::string(input_string->value()).empty()) {
00052         result = input_string->value();
00053     } else if (!std::string(input_int->value()).empty()) {
00054         result = input_int->value();
00055     } else {
00056         result = input_float->value();
00057     }
00058     return result;
00059 }
  
```

### 7.8.3.8 visible()

```
bool pssp::InputManager::visible ( ) const
00025     {
00026     return ((input_string->visible() != 0) || (input_int->visible() != 0) ||
00027            (input_float->visible() != 0));
00028 }
```

## 7.8.4 Member Data Documentation

### 7.8.4.1 input\_float

```
std::unique_ptr<Fl_Float_Input> pssp::InputManager::input_float {} [private]
00068 {};
```

### 7.8.4.2 input\_int

```
std::unique_ptr<Fl_Int_Input> pssp::InputManager::input_int {} [private]
00067 {};
```

### 7.8.4.3 input\_string

```
std::unique_ptr<Fl_Input> pssp::InputManager::input_string {} [private]
00066 {};
```

### 7.8.4.4 modified

```
bool pssp::InputManager::modified {false}
00062 {false};
```

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

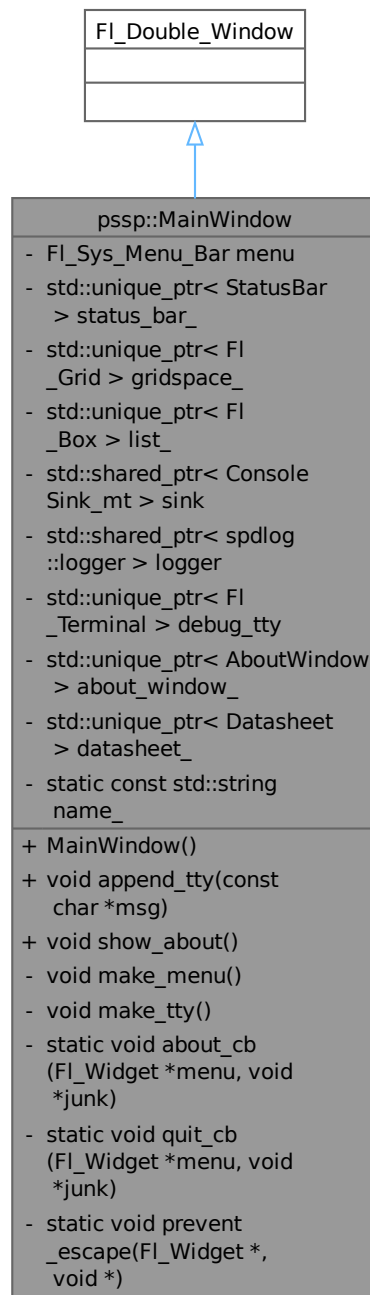
- include/PsSp/Managers/InputManager.hpp
- src/Managers/InputManager.cpp

## 7.9 pssp::MainWindow Class Reference

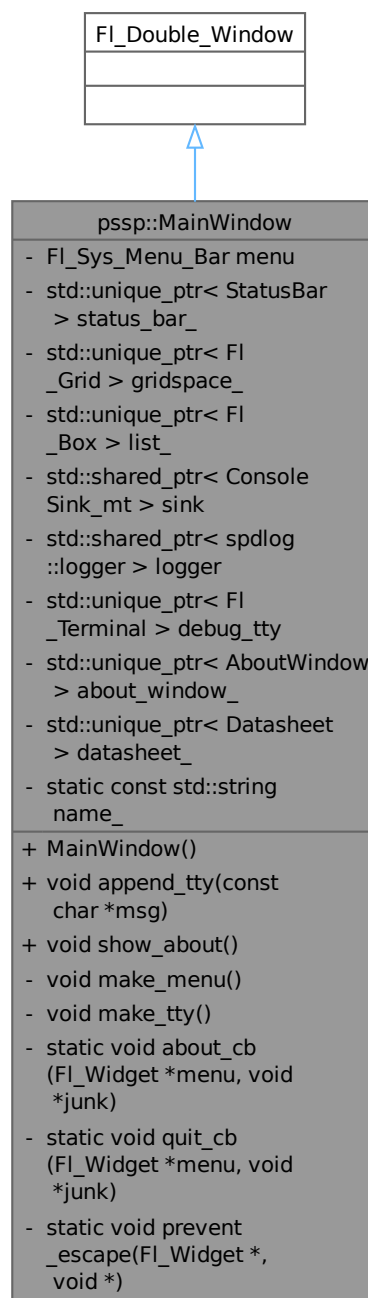
Class to provide the Main Window.

```
#include <Main.hpp>
```

Inheritance diagram for pssp::MainWindow:



Collaboration diagram for pssp::MainWindow:



## Public Member Functions

- [MainWindow \(\)](#)  
*MainWindow constructor.*
- `void append\_tty (const char *msg)`  
*Add a message to the end of the console log.*
- `void show\_about ()`  
*Show the [AboutWindow](#).*

### Private Member Functions

- void [make\\_menu](#) ()  
*Construct and initialize the Menu-bar at the top.*
- void [make\\_tty](#) ()  
*Construct and initialize the FL\_Terminal log display.*

### Static Private Member Functions

- static void [about\\_cb](#) (FI\_Widget \*menu, void \*junk)  
*Callback function to show the [AboutWindow](#).*
- static void [quit\\_cb](#) (FI\_Widget \*menu, void \*junk)  
*Menu/Hotkey Quit callback command.*
- static void [prevent\\_escape](#) (FI\_Widget \*, void \*)

### Private Attributes

- FI\_Sys\_Menu\_Bar [menu](#) {0, 0, 0, [mw::menu\\_height](#), nullptr}  
*The menubar (Window/Linux) or systembar (macOS).*
- std::unique\_ptr< [StatusBar](#) > [status\\_bar\\_](#) {}  
*PsSp [StatusBar](#).*
- std::unique\_ptr< FI\_Grid > [gridspace\\_](#) {}  
*Grid to layout window components.*
- std::unique\_ptr< FI\_Box > [list\\_](#) {}  
*Record-organization sidebar object (prototype).*
- std::shared\_ptr< [ConsoleSink\\_mt](#) > [sink](#) {}  
*[ConsoleSink](#) debug log sink.*
- std::shared\_ptr< spdlog::logger > [logger](#) {}  
*spdlog log source*
- std::unique\_ptr< FI\_Terminal > [debug\\_tty](#) {}  
*Terminal to display [ConsoleSink](#) formatted logs.*
- std::unique\_ptr< [AboutWindow](#) > [about\\_window\\_](#) {}  
*The [AboutWindow](#).*
- std::unique\_ptr< [Datasheet](#) > [datasheet\\_](#) {}  
*The [Datasheet](#) to display (spreadsheet of records).*

### Static Private Attributes

- static const std::string [name\\_](#) {"PsSp - Passive-source Seismic-processing"}  
*Program name.*

## 7.9.1 Detailed Description

Class to provide the Main Window.

This provides the main window for the PsSp program.

**Todo** Work on record-organization sidebar object.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 MainWindow()

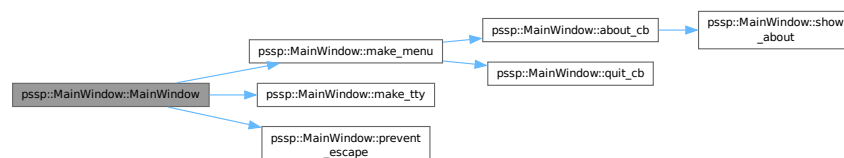
pssp::MainWindow::MainWindow ( )

[MainWindow](#) constructor.

This creates the [MainWindow](#) object with all the specified sizes from the [pssp::mw](#) namespace and maximizes the window.

```
00020             : Fl_Double_Window(0, 0, name_.c_str()) {
00021   this->callback(prevent_escape);
00022   make_tty();
00023   spdlog::trace("Building \033[1mMainWindow\033[0m.");
00024   this->begin();
00025   resizable(this);
00026   // Minimum window size width/height
00027   this->size_range(mw::minimum_x, mw::minimum_y);
00028   structs::Geometry geo{};
00029   Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
00030   this->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00031   make_menu();
00032   menu.resize(0, 0, geo.width, menu.h());
00033   status_bar_ = std::make_unique<StatusBar>(this->h(), this->w(), menu.h());
00034   #if defined(__APPLE__)
00035     const int menu_shift{0};
00036   #else
00037     const int menu_shift{menu.h()};
00038   #endif
00039   gridspace_ = std::make_unique<Fl_Grid>(0, menu_shift, this->w(),
00040                                           this->h() - menu_shift - menu.h());
00041   gridspace_->begin();
00042   gridspace_->add(debug_tty.get());
00043   gridspace_->show_grid(0); // 1 to show guide lines
00044   constexpr structs::Grid layout{10, 10, 1, 1};
00045   gridspace_->layout(layout.row, layout.col, layout.row_span, layout.col_span);
00046   list_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "List");
00047   list_->box(FL_BORDER_BOX);
00048   list_->color(FL_WHITE);
00049   datasheet_ = std::make_unique<Datasheet>();
00050   constexpr structs::Grid tty_grid{7, 0, 3, 10};
00051   gridspace_->widget(debug_tty.get(), tty_grid.row, tty_grid.col,
00052                     tty_grid.row_span, tty_grid.col_span);
00053   constexpr structs::Grid list_grid{0, 0, 7, 2};
00054   gridspace_->widget(list_.get(), list_grid.row, list_grid.col,
00055                     list_grid.row_span, list_grid.col_span);
00056   constexpr structs::Grid ds_grid{0, 2, 7, 8};
00057   gridspace_->widget(datasheet_.get(), ds_grid.row, ds_grid.col,
00058                     ds_grid.row_span, ds_grid.col_span);
00059   gridspace_->end();
00060   this->end();
00061   this->resizable(status_bar_.get());
00062   this->resizable(datasheet_.get());
00063   this->resizable(gridspace_.get());
00064   about_window_ = std::make_unique<AboutWindow>();
00065   about_window_->hide();
00066   spdlog::trace("Done making \033[1mMainWindow\033[0m.");
00067 }
```

Here is the call graph for this function:



## 7.9.3 Member Function Documentation

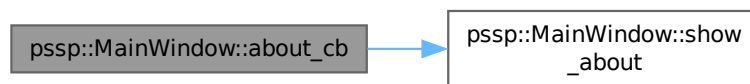
### 7.9.3.1 about\_cb()

```
void pssp::MainWindow::about_cb (
    Fl_Widget * menu,
    void * junk ) [static], [private]
```

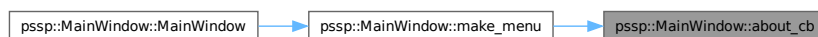
Callback function to show the [AboutWindow](#).

```
00183                                     {
00184     (void)junk;
00185     auto *window = reinterpret_cast<MainWindow *>(menu->parent()->as_window());
00186     window->show_about();
00187 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.9.3.2 append\_tty()

```
void pssp::MainWindow::append_tty (
    const char * msg )
```

Add a message to the end of the console log.

```
00157 { debug_tty->append(msg); }
```

### 7.9.3.3 make\_menu()

```
void pssp::MainWindow::make_menu ( ) [private]
```

Construct and initialize the Menu-bar at the top.

On Linux/Windows this is a standard Top-menu bar that takes up some window space.

On macOS this is a system-menu bar that does not take up any window space.

This also links all menu options to their respective call\_backs (or nullptr if just a placeholder).



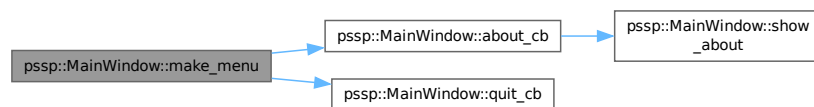
**Todo** Fix shallow menus that do not display on macOS (all menus must have depth).

```

00107         {
00108     spdlog::trace("Making \033[1mMenu\033[0m.");
00109     // Program
00110     menu.add("&Program/&Quit", FL_COMMAND + 'q', quit_cb, this);
00111     // Project
00112     menu.add("&Project/&New", FL_COMMAND + 'n', nullptr, this, FL_MENU_INACTIVE);
00113     menu.add("&Project/&Load", FL_COMMAND + 'o', nullptr, this, FL_MENU_INACTIVE);
00114     menu.add("&Project/&Close", FL_COMMAND + 'c', nullptr, this,
00115             FL_MENU_INACTIVE);
00116     menu.add("&Project/&Bookmark", FL_COMMAND + 'b', nullptr, this,
00117             FL_MENU_INACTIVE);
00118     // Data
00119     menu.add("&Data/&Add File", 0, nullptr, this, FL_MENU_INACTIVE);
00120     menu.add("&Data/&Add Directory", 0, nullptr, this, FL_MENU_INACTIVE);
00121     menu.add("&Data/&Download Data", 0, nullptr, this, FL_MENU_INACTIVE);
00122     // Processing
00123     menu.add("&Processing/&Filters/&Butterworth/&Lowpass", 0, nullptr, this,
00124             FL_MENU_INACTIVE);
00125     menu.add("&Processing/&Filters/&Butterworth/&Highpass", 0, nullptr, this,
00126             FL_MENU_INACTIVE);
00127     menu.add("&Processing/&Filters/&Butterworth/&Bandpass", 0, nullptr, this,
00128             FL_MENU_INACTIVE);
00129     // Plotting
00130     menu.add("&Plot/&Single Component/&Time-series", 0, nullptr, this,
00131             FL_MENU_INACTIVE);
00132     menu.add("&Plot/&Single Component/&Spectrum/&Real-Imaginary", 0, nullptr,
00133             this, FL_MENU_INACTIVE);
00134     menu.add("&Plot/&Single Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
00135             this, FL_MENU_INACTIVE);
00136     menu.add("&Plot/&Single Component/&Spectrogram", 0, nullptr, this,
00137             FL_MENU_INACTIVE);
00138     menu.add("&Plot/&Three Component/&Time-series", 0, nullptr, this,
00139             FL_MENU_INACTIVE);
00140     menu.add("&Plot/&Three Component/&Spectrum/&Real-Imaginary", 0, nullptr, this,
00141             FL_MENU_INACTIVE);
00142     menu.add("&Plot/&Three Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
00143             this, FL_MENU_INACTIVE);
00144     menu.add("&Plot/&Three Component/&Spectrogram", 0, nullptr, this,
00145             FL_MENU_INACTIVE);
00146     menu.add("&Plot/&Profile", 0, nullptr, this, FL_MENU_INACTIVE);
00147     // Settings
00148     menu.add("&Settings", 0, nullptr, this, FL_MENU_INACTIVE);
00149     // Help
00150     menu.add("&Help", 0, nullptr, this, FL_MENU_INACTIVE);
00151     // About
00152     menu.add("&About", 0, about_cb, this);
00153     spdlog::trace("Done making \033[1mMenu\033[0m.");
00154 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.9.3.4 make\_tty()

```
void pssp::MainWindow::make_tty ( ) [private]
```

Construct and initialize the FL\_Terminal log display.

```
00072     {
00073     // Debug terminal
00074     debug_tty = std::make_unique<Fl_Terminal>(0, 0, 0, 0);
00075     sink = std::make_shared<ConsoleSink_mt>(debug_tty.get());
00076     logger = std::make_shared<spdlog::logger>("tty logger", sink);
00077     spdlog::set_default_logger(logger);
00078     // levels are critical, error, warn, info, debug, trace
00079     spdlog::set_level(spdlog::level::trace);
00080     spdlog::set_pattern(
00081         "\33[1m\33[32m[%Y-%m-%d %T]\33[33m[%l]\33[36m[thread %t]\33[0m %v");
00082     debug_tty->begin();
00083     constexpr int font_size{14};
00084     debug_tty->textsize(font_size);
00085     debug_tty->redraw_style(Fl_Terminal::NO_REDRAW);
00086     constexpr int num_columns{80};
00087     debug_tty->display_columns(num_columns);
00088     spdlog::trace("Logger started.");
00089     debug_tty->end();
00090     resizable();
00091 }
```

Here is the caller graph for this function:



### 7.9.3.5 prevent\_escape()

```
void pssp::MainWindow::prevent_escape (
    Fl_Widget * caller,
    void * data ) [static], [private]
```

FLTK has the odd-behavior of having a built-in auto-close callback upon either the Escape key or the Q key being hit. Just immediate closure with no questions asked.

This is silly, both keys are useful for programs and having a program suddenly close due to your pinky pressing down on either key is rather jarring.

This disables that nonsense for the [MainWindow](#) so that we can have some sane functionality instead.

```
00200     {
00201     (void) caller;
00202     (void) data;
00203     if ((Fl::event() == FL_SHORTCUT) && (Fl::event_key() == FL_Escape)) {
00204         return; // ignore Escape
00205     }
00206     exit(0);
00207 }
```

Here is the caller graph for this function:



### 7.9.3.6 quit\_cb()

```
void pssp::MainWindow::quit_cb (
    Fl_Widget * menu,
    void * junk ) [static], [private]
```

Menu/Hotkey Quit callback command.

When the user chooses to close the program, pop-up a confirmation window before annihilating the window (wouldn't you like to save first?).

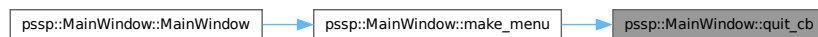
**Todo** Request if the user wants to save first (if unsaved work).

Doesn't display on macOS when CMD+Q is hit (just closes).

BugFix: Doesn't display when keyboard input is captured by [Datasheet](#).

```
00169                                     {
00170     (void) junk;
00171     // reinterpret_cast is unnecessary, but I wanted to figure it out
00172     auto *window = reinterpret_cast<MainWindow *>(menu->parent()->as_window());
00173     if (fl_choice("Are you sure you want to quit?", "cancel", "quit", nullptr) !=
00174         0) {
00175         window->hide();
00176     }
00177 }
```

Here is the caller graph for this function:



### 7.9.3.7 show\_about()

```
void pssp::MainWindow::show_about ( )
```

Show the [AboutWindow](#).

```
00180 { about_window_->show(); }
```

Here is the caller graph for this function:



## 7.9.4 Member Data Documentation

### 7.9.4.1 about\_window\_

```
std::unique_ptr<AboutWindow> pssp::MainWindow::about_window_ {} [private]
```

The [AboutWindow](#).

```
00089 {};
```

#### 7.9.4.2 datasheet\_

```
std::unique_ptr<Datasheet> pssp::MainWindow::datasheet_ {} [private]
```

The [Datasheet](#) to display (spreadsheet of records).

```
00091 {};
```

#### 7.9.4.3 debug\_tty

```
std::unique_ptr<Fl_Terminal> pssp::MainWindow::debug_tty {} [private]
```

Terminal to display [ConsoleSink](#) formatted logs.

```
00087 {};
```

#### 7.9.4.4 gridspace\_

```
std::unique_ptr<Fl_Grid> pssp::MainWindow::gridspace_ {} [private]
```

Grid to layout window components.

```
00079 {};
```

#### 7.9.4.5 list\_

```
std::unique_ptr<Fl_Box> pssp::MainWindow::list_ {} [private]
```

Record-organization sidebar object (prototype).

```
00081 {};
```

#### 7.9.4.6 logger

```
std::shared_ptr<spdlog::logger> pssp::MainWindow::logger {} [private]
```

spdlog log source

```
00085 {};
```

#### 7.9.4.7 menu

```
Fl_Sys_Menu_Bar pssp::MainWindow::menu {0, 0, 0, mw::menu_height, nullptr} [private]
```

The menubar (Window/Linux) or systembar (macOS).

```
00073 {0, 0, 0, mw::menu_height, nullptr};
```

#### 7.9.4.8 name\_

```
const std::string pssp::MainWindow::name_ {"PsSp - Passive-source Seismic-processing"} [inline],  
[static], [private]
```

Program name.

```
00097 {"PsSp - Passive-source Seismic-processing"};
```

#### 7.9.4.9 sink

```
std::shared_ptr<ConsoleSink_mt> pssp::MainWindow::sink {} [private]
```

[ConsoleSink](#) debug log sink.

```
00083 {};
```

#### 7.9.4.10 status\_bar\_

```
std::unique_ptr<StatusBar> pssp::MainWindow::status_bar_ {} [private]
```

PsSp [StatusBar](#).

```
00077 {};
```

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

- include/PsSp/Windows/Main.hpp
- src/Windows/Main.cpp

## 7.10 pssp::SheetManager Class Reference

```
#include <SheetManager.hpp>
```

Collaboration diagram for pssp::SheetManager:

pssp::SheetManager
<ul style="list-style-type: none"> <li>- std::vector&lt; std::array&lt; std::string, constants::sac_string &gt; &gt; strings</li> <li>- std::vector&lt; std::array&lt; int, constants::sac_int &gt; &gt; ints</li> <li>- std::vector&lt; std::array&lt; float, constants::sac_float &gt; &gt; floats</li> <li>- std::vector&lt; std::array&lt; double, constants::sac_double &gt; &gt; doubles</li> <li>- std::vector&lt; std::array&lt; bool, constants::sac_bool &gt; &gt; bools</li> </ul>
<ul style="list-style-type: none"> <li>+ SheetManager()</li> <li>+ void resize_data(size_t size)</li> <li>+ int rows() const</li> <li>+ int cols() const</li> <li>+ void set(size_t row, const Field &amp;field, const std::string &amp;input)</li> <li>+ void set(size_t row, const Field &amp;field, int input)</li> <li>+ void set(size_t row, const Field &amp;field, float input)</li> <li>+ void set(size_t row, const Field &amp;field, double input)</li> <li>+ void set(size_t row, const Field &amp;field, bool input)</li> <li>+ std::string get(size_t row, const Field &amp;field)</li> <li>+ std::string get_string(size_t row, const Field &amp;field)</li> <li>+ int get_int(size_t row, const Field &amp;field)</li> <li>+ float get_float(size_t row, const Field &amp;field)</li> <li>+ double get_double(size_t row, const Field &amp;field)</li> <li>+ bool get_bool(size_t row, const Field &amp;field)</li> </ul>

## Public Member Functions

- [SheetManager](#) ()
- void [resize\\_data](#) (size\_t size)
- int [rows](#) () const
- int [cols](#) () const
- void [set](#) (size\_t row, const [Field](#) &field, const std::string &input)

- void [set](#) (size\_t row, const [Field](#) &field, int input)
- void [set](#) (size\_t row, const [Field](#) &field, float input)
- void [set](#) (size\_t row, const [Field](#) &field, double input)
- void [set](#) (size\_t row, const [Field](#) &field, bool input)
- std::string [get](#) (size\_t row, const [Field](#) &field)
- std::string [get\\_string](#) (size\_t row, const [Field](#) &field)
- int [get\\_int](#) (size\_t row, const [Field](#) &field)
- float [get\\_float](#) (size\_t row, const [Field](#) &field)
- double [get\\_double](#) (size\_t row, const [Field](#) &field)
- bool [get\\_bool](#) (size\_t row, const [Field](#) &field)

### Private Attributes

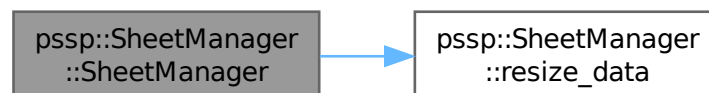
- std::vector< std::array< std::string, [constants::sac\\_string](#) > > [strings](#) {}
- std::vector< std::array< int, [constants::sac\\_int](#) > > [ints](#) {}
- std::vector< std::array< float, [constants::sac\\_float](#) > > [floats](#) {}
- std::vector< std::array< double, [constants::sac\\_double](#) > > [doubles](#) {}
- std::vector< std::array< bool, [constants::sac\\_bool](#) > > [bools](#) {}

## 7.10.1 Constructor & Destructor Documentation

### 7.10.1.1 SheetManager()

```
pssp::SheetManager::SheetManager ( )
00006 { resize\_data(100); }
```

Here is the call graph for this function:



## 7.10.2 Member Function Documentation

### 7.10.2.1 cols()

```
int pssp::SheetManager::cols ( ) const
00018 {
00019     size_t num_cols{strings[0].size() + ints[0].size() + floats[0].size() +
00020                   doubles[0].size() + bools[0].size()};
00021     return static_cast<int>(num_cols);
00022 }
```

### 7.10.2.2 get()

```
std::string pssp::SheetManager::get (
    size_t row,
    const Field & field )
{
00077
00078     std::string result{};
00079     const trace_info &info{field_info.at(field)};
00080     switch (info.type) {
00081     case Type::string_:
00082         break;
00083     case Type::int_:
00084         result = std::to_string(ints[row][info.array_col]);
00085         break;
00086     case Type::float_:
00087         result = std::to_string(floats[row][info.array_col]);
00088         break;
00089     case Type::double_:
00090         result = std::to_string(doubles[row][info.array_col]);
00091         break;
00092     case Type::bool_:
00093         result = std::to_string(static_cast<int>(bools[row][info.array_col]));
00094         break;
00095     default:
00096         break;
00097     }
00098     return result;
00099 }
```

### 7.10.2.3 get\_bool()

```
bool pssp::SheetManager::get_bool (
    size_t row,
    const Field & field )
{
00149
00150     bool result{};
00151     const trace_info &info{field_info.at(field)};
00152     if (info.type == Type::bool_) {
00153         result = bools[row][info.array_col];
00154     } else {
00155         spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00156                     type_names.at(info.type));
00157     }
00158     return result;
00159 }
```

### 7.10.2.4 get\_double()

```
double pssp::SheetManager::get_double (
    size_t row,
    const Field & field )
{
00137
00138     double result{};
00139     const trace_info &info{field_info.at(field)};
00140     if (info.type == Type::double_) {
00141         result = doubles[row][info.array_col];
00142     } else {
00143         spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00144                     type_names.at(info.type));
00145     }
00146     return result;
00147 }
```

### 7.10.2.5 get\_float()

```
float pssp::SheetManager::get_float (
    size_t row,
    const Field & field )
```



```

00125                                     {
00126     float result{};
00127     const trace_info &info{field_info.at(field)};
00128     if (info.type == Type::float_) {
00129         result = floats[row][info.array_col];
00130     } else {
00131         spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00132                       type_names.at(info.type));
00133     }
00134     return result;
00135 }

```

### 7.10.2.6 get\_int()

```

int pssp::SheetManager::get_int (
    size_t row,
    const Field & field )
00113                                     {
00114     int result{};
00115     const trace_info &info{field_info.at(field)};
00116     if (info.type == Type::int_) {
00117         result = ints[row][info.array_col];
00118     } else {
00119         spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00120                       type_names.at(info.type));
00121     }
00122     return result;
00123 }

```

### 7.10.2.7 get\_string()

```

std::string pssp::SheetManager::get_string (
    size_t row,
    const Field & field )
00101                                     {
00102     std::string result{};
00103     const trace_info &info{field_info.at(field)};
00104     if (info.type == Type::string_) {
00105         result = strings[row][info.array_col];
00106     } else {
00107         spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00108                       type_names.at(info.type));
00109     }
00110     return result;
00111 }

```

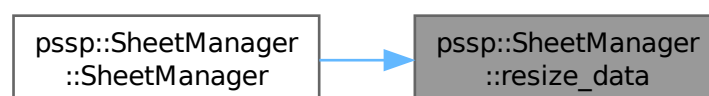
### 7.10.2.8 resize\_data()

```

void pssp::SheetManager::resize_data (
    size_t size )
00008                                     {
00009     strings.resize(size);
00010     ints.resize(size);
00011     floats.resize(size);
00012     doubles.resize(size);
00013     bools.resize(size);
00014 }

```

Here is the caller graph for this function:



### 7.10.2.9 rows()

```
int pssp::SheetManager::rows ( ) const
00016 { return static_cast<int>(bools.size()); }
```

### 7.10.2.10 set() [1/5]

```
void pssp::SheetManager::set (
    size_t row,
    const Field & field,
    bool input )
00067 {
00068     const trace_info &info{field_info.at(field)};
00069     if (info.type == Type::bool_) {
00070         bools[row][info.array_col] = input;
00071     } else {
00072         spdlog::error("Wrong type {0} inserted into field {1}.",
00073             type_names.at(info.type), info.name);
00074     }
00075 }
```

### 7.10.2.11 set() [2/5]

```
void pssp::SheetManager::set (
    size_t row,
    const Field & field,
    const std::string & input )
00025 {
00026     const trace_info &info{field_info.at(field)};
00027     if (info.type == Type::string_) {
00028         strings[row][info.array_col] = input;
00029     } else {
00030         spdlog::error("Wrong type {0} inserted into field {1}.",
00031             type_names.at(info.type), info.name);
00032     }
00033 }
```

### 7.10.2.12 set() [3/5]

```
void pssp::SheetManager::set (
    size_t row,
    const Field & field,
    double input )
00057 {
00058     const trace_info &info{field_info.at(field)};
00059     if (info.type == Type::double_) {
00060         doubles[row][info.array_col] = input;
00061     } else {
00062         spdlog::error("Wrong type {0} inserted into field {1}.",
00063             type_names.at(info.type), info.name);
00064     }
00065 }
```

### 7.10.2.13 set() [4/5]

```
void pssp::SheetManager::set (
    size_t row,
    const Field & field,
    float input )
00046 {
00047     const trace_info &info{field_info.at(field)};
00048     if (info.type == Type::float_) {
00049         floats[row][info.array_col] = input;
00050     } else {
00051         spdlog::error("Wrong type {0} inserted into field {1}.",
00052             type_names.at(info.type), info.name);
00053     }
00054 }
```

### 7.10.2.14 set() [5/5]

```

void pssp::SheetManager::set (
    size_t row,
    const Field & field,
    int input )
00035                                     {
00036     const trace_info &info{field_info.at(field)};
00037     if (info.type == Type::int_) {
00038         ints[row][info.array_col] = input;
00039     } else {
00040         spdlog::error("Wrong type {0} inserted into field {1}.",
00041                     type_names.at(info.type), info.name);
00042     }
00043 }

```

## 7.10.3 Member Data Documentation

### 7.10.3.1 bools

```

std::vector<std::array<bool, constants::sac_bool> > pssp::SheetManager::bools {} [private]
00057 {};

```

### 7.10.3.2 doubles

```

std::vector<std::array<double, constants::sac_double> > pssp::SheetManager::doubles {} [private]
00055 {};

```

### 7.10.3.3 floats

```

std::vector<std::array<float, constants::sac_float> > pssp::SheetManager::floats {} [private]
00053 {};

```

### 7.10.3.4 ints

```

std::vector<std::array<int, constants::sac_int> > pssp::SheetManager::ints {} [private]
00051 {};

```

### 7.10.3.5 strings

```

std::vector<std::array<std::string, constants::sac_string> > pssp::SheetManager::strings {}
[private]
00049 {};

```

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

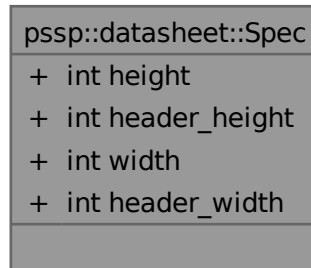
- include/PsSp/Managers/SheetManager.hpp
- src/Managers/SheetManager.cpp

## 7.11 pssp::datasheet::Spec Struct Reference

Used to specify the size of [Datasheet](#) cells.

```
#include <Datasheet.hpp>
```

Collaboration diagram for pssp::datasheet::Spec:



### Public Attributes

- int [height](#) {0}  
*Cell height (pixels).*
- int [header\\_height](#) {0}  
*Header-cell height (pixels).*
- int [width](#) {0}  
*Cell width (pixels).*
- int [header\\_width](#) {0}  
*Header-cell width (pixels).*

### 7.11.1 Detailed Description

Used to specify the size of [Datasheet](#) cells.

### 7.11.2 Member Data Documentation

#### 7.11.2.1 header\_height

```
int pssp::datasheet::Spec::header_height {0}
```

Header-cell height (pixels).  
00077 {0};

### 7.11.2.2 header\_width

```
int pssp::datasheet::Spec::header_width {0}
```

Header-cell width (pixels).

```
00081 {0};
```

### 7.11.2.3 height

```
int pssp::datasheet::Spec::height {0}
```

Cell height (pixels).

```
00075 {0};
```

### 7.11.2.4 width

```
int pssp::datasheet::Spec::width {0}
```

Cell width (pixels).

```
00079 {0};
```

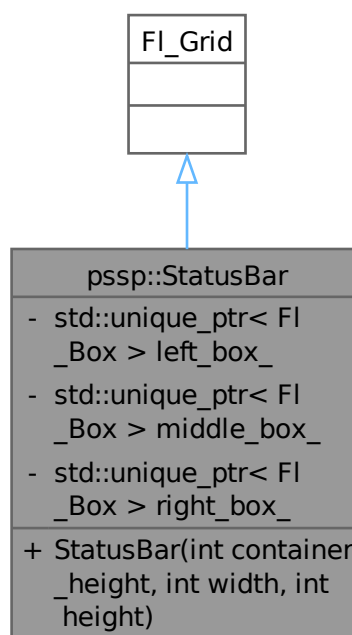
The documentation for this struct was generated from the following file:

- include/PsSp/Widgets/Datasheet.hpp

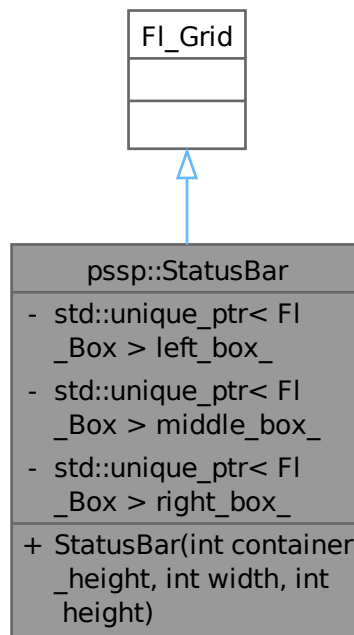
## 7.12 pssp::StatusBar Class Reference

```
#include <StatusBar.hpp>
```

Inheritance diagram for pssp::StatusBar:



Collaboration diagram for pssp::StatusBar:



## Public Member Functions

- [StatusBar](#) (int container\_height, int width, int height)

## Private Attributes

- std::unique\_ptr< Fl\_Box > [left\\_box\\_](#) {}
- std::unique\_ptr< Fl\_Box > [middle\\_box\\_](#) {}
- std::unique\_ptr< Fl\_Box > [right\\_box\\_](#) {}

## 7.12.1 Constructor & Destructor Documentation

### 7.12.1.1 StatusBar()

```

pssp::StatusBar::StatusBar (
    int container_height,
    int width,
    int height )
00007 : Fl_Grid(0, container_height - height, width, height) {
00008     spdlog::trace("Making \033[1mStatusBar\033[0m.");
00009     this->begin();
00010     constexpr structs::Grid layout{1, 10, 1, 1};
00011     this->layout(layout.row, layout.col, layout.row_span, layout.col_span);
00012     left_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Left Box");
00013     left_box_->box(FL_BORDER_BOX);
  
```

```

00014 constexpr structs::Grid left{0, 0, 1, 2};
00015 this->widget(left_box_.get(), left.row, left.col, left.row_span,
00016             left.col_span);
00017 middle_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Middle Box");
00018 middle_box_->box(FL_BORDER_BOX);
00019 constexpr structs::Grid middle{0, 2, 1, 6};
00020 this->widget(middle_box_.get(), middle.row, middle.col, middle.row_span,
00021             middle.col_span);
00022 right_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Right Box");
00023 right_box_->box(FL_BORDER_BOX);
00024 constexpr structs::Grid right{0, 8, 1, 2};
00025 this->widget(right_box_.get(), right.row, right.col, right.row_span,
00026             right.col_span);
00027 this->end();
00028 spdlog::trace("Done making \033[1mStatus_Bar\033[0m.");
00029 }

```

## 7.12.2 Member Data Documentation

### 7.12.2.1 left\_box\_

```

std::unique_ptr<Fl_Box> pssp::StatusBar::left_box_ {} [private]
00024 {};

```

### 7.12.2.2 middle\_box\_

```

std::unique_ptr<Fl_Box> pssp::StatusBar::middle_box_ {} [private]
00025 {};

```

### 7.12.2.3 right\_box\_

```

std::unique_ptr<Fl_Box> pssp::StatusBar::right_box_ {} [private]
00026 {};

```

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

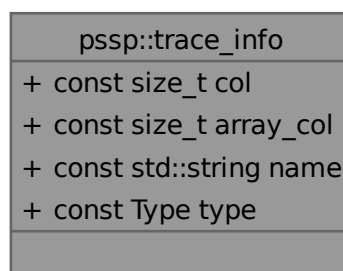
- include/PsSp/Widgets/StatusBar.hpp
- src/Widgets/StatusBar.cpp

## 7.13 pssp::trace\_info Struct Reference

Information for use in the [Datasheet](#).

```
#include <Enums.hpp>
```

Collaboration diagram for pssp::trace\_info:



## Public Attributes

- `const size_t col {0}`  
*Location of value in [Datasheet](#).*
- `const size_t array_col {0}`  
*Location of value in internal storage array.*
- `const std::string name {}`  
*Data Type name derived from [type\\_names](#).*
- `const Type type {}`  
*Data Type.*

### 7.13.1 Detailed Description

Information for use in the [Datasheet](#).

This is information for a specific column (row each row) in teh [Datasheet](#).

**Todo** Move to PsSp/Utility/Structs.hpp

### 7.13.2 Member Data Documentation

#### 7.13.2.1 array\_col

```
const size_t pssp::trace_info::array_col {0}
```

Location of value in internal storage array.  
00067 {};

#### 7.13.2.2 col

```
const size_t pssp::trace_info::col {0}
```

Location of value in [Datasheet](#).  
00065 {};

#### 7.13.2.3 name

```
const std::string pssp::trace_info::name {}
```

Data Type name derived from [type\\_names](#).  
00069 {};

#### 7.13.2.4 type

```
const Type pssp::trace_info::type {}
```

Data Type.  
00071 {};

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

- `include/PsSp/Utility/Enums.hpp`

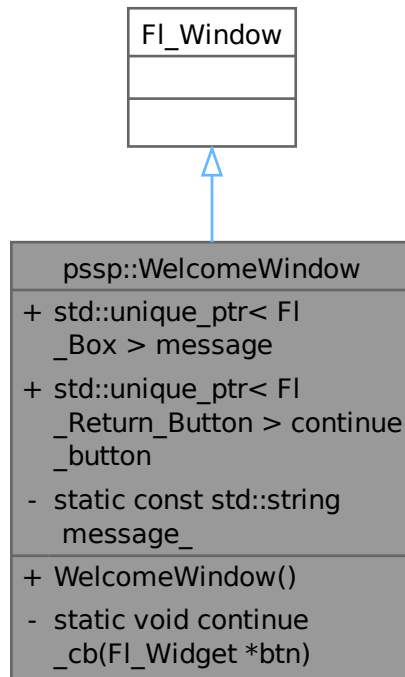


## 7.14 pssp::WelcomeWindow Class Reference

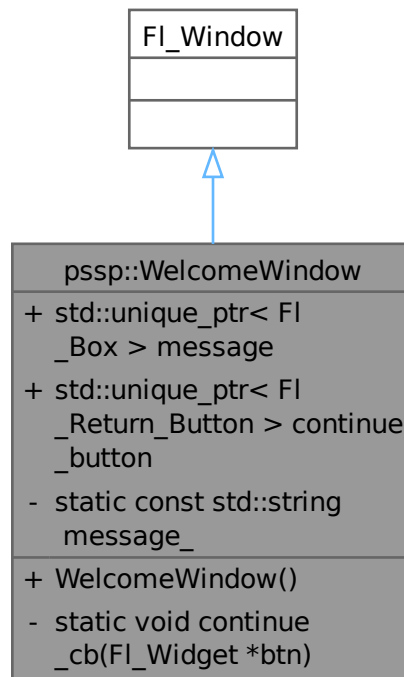
Class to provide a Welcome Window.

```
#include <Welcome.hpp>
```

Inheritance diagram for pssp::WelcomeWindow:



Collaboration diagram for pssp::WelcomeWindow:



### Public Member Functions

- [WelcomeWindow](#) ()  
*WelcomeWindow constructor.*

### Public Attributes

- `std::unique_ptr< FI_Box > message` {}  
*Box to contain message.*
- `std::unique_ptr< FI_Return_Button > continue\_button` {}  
*Button to close window.*

### Static Private Member Functions

- `static void continue\_cb (FI_Widget *btn)`  
*Continue button callback function.*

### Static Private Attributes

- `static const std::string message\_`  
*Message to display in the welcome window.*

### 7.14.1 Detailed Description

Class to provide a Welcome Window.

This provides a welcome window that is open on program startup.

**Todo** Auto-size window to size of message.

"Do not show again" checkbox.

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 WelcomeWindow()

```
pssp::WelcomeWindow::WelcomeWindow ( )
```

[WelcomeWindow](#) constructor.

This creates a [WelcomeWindow](#) object with all specified sizes from the welcome namespace and centers it on the screen.

```
00019                                     : Fl_Window(0, 0, 0, 0, "Welcome!") {
00020     this->begin();
00021     int x_start{};
00022     int y_start{};
00023     int width{};
00024     int height{};
00025     Fl::screen_work_area(x_start, y_start, width, height);
00026     x_start = ((width - welcome::width) / 2);
00027     y_start = ((height - welcome::height) / 2);
00028     this->resize(x_start, y_start, welcome::width, welcome::height);
00029     this->box(FL_BORDER_BOX);
00030     set_modal();
00031     message =
00032         std::make_unique<Fl_Box>((welcome::width - welcome::text_width) / 2, 0,
00033                                 welcome::text_width, welcome::text_height);
00034     continue_button = std::make_unique<Fl_Return_Button>(
00035         (welcome::width - welcome::button_width) / 2, welcome::text_height,
00036         welcome::button_width, welcome::button_height, "Continue");
00037     message->label(message_.c_str());
00038     message->align(FL_ALIGN_CENTER);
00039     continue_button->callback(continue_cb);
00040     this->end();
00041 }
```

Here is the call graph for this function:



### 7.14.3 Member Function Documentation

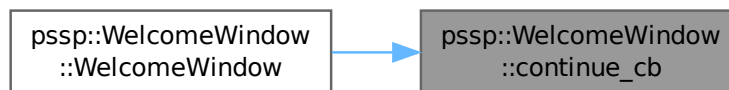
#### 7.14.3.1 continue\_cb()

```
void pssp::WelcomeWindow::continue_cb (
    Fl_Widget * btn ) [static], [private]
```

Continue button callback function.

```
00048 { btn->parent()->hide(); }
```

Here is the caller graph for this function:



### 7.14.4 Member Data Documentation

#### 7.14.4.1 continue\_button

```
std::unique_ptr<Fl_Return_Button> pssp::WelcomeWindow::continue_button {}
```

Button to close window.

```
00064 {};
```

#### 7.14.4.2 message

```
std::unique_ptr<Fl_Box> pssp::WelcomeWindow::message {}
```

Box to contain message.

```
00062 {};
```

#### 7.14.4.3 message\_

```
const std::string pssp::WelcomeWindow::message_ [inline], [static], [private]
```

**Initial value:**

```
{"Welcome to Passive-source Seismic-processing (PsSp)!\n"
  "This program is very early in development..."}
```

Message to display in the welcome window.

```
00072 {"Welcome to Passive-source Seismic-processing (PsSp)!\n"
00073 "This program is very early in development..."};
```

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

- include/PsSp/Windows/Welcome.hpp
- src/Windows/Welcome.cpp

# Index

a

pssp, [14](#)

about\_cb

pssp::MainWindow, [64](#)

about\_window\_

pssp::MainWindow, [67](#)

AboutWindow

pssp::AboutWindow, [31](#)

alignment

pssp::datasheet::Cell, [36](#)

append\_tty

pssp::MainWindow, [64](#)

Application

pssp::Application, [33](#)

array\_col

pssp::trace\_info, [80](#)

az

pssp, [14](#)

b

pssp, [14](#)

baz

pssp, [14](#)

bool\_

pssp, [17](#)

bools

pssp::SheetManager, [75](#)

box\_color

pssp::datasheet::Cell, [36](#)

box\_type

pssp::datasheet::Cell, [36](#)

button\_height

pssp::about, [22](#)

pssp::welcome, [27](#)

button\_width

pssp::about, [22](#)

pssp::welcome, [27](#)

cell\_buffer

pssp::datasheet, [24](#)

check\_button

pssp::Datasheet, [50](#)

cleanup

pssp::InputManager, [56](#)

clear

pssp::InputManager, [56](#)

cmpaz

pssp, [14](#)

cmpinc

pssp, [14](#)

col

pssp::structs::Grid, [53](#)

pssp::trace\_info, [80](#)

col\_span

pssp::structs::Grid, [53](#)

cols

pssp::SheetManager, [71](#)

ConsoleSink

pssp::ConsoleSink< Mutex >, [39](#)

ConsoleSink\_mt

pssp, [13](#)

ConsoleSink\_st

pssp, [13](#)

continue\_button

pssp::WelcomeWindow, [84](#)

continue\_cb

pssp::WelcomeWindow, [84](#)

data1

pssp, [15](#)

data2

pssp, [15](#)

Datasheet

pssp::Datasheet, [43](#)

datasheet\_

pssp::MainWindow, [67](#)

debug\_tty

pssp::MainWindow, [68](#)

delta

pssp, [14](#)

depmax

pssp, [13](#)

depmen

pssp, [14](#)

depmin

pssp, [13](#)

dist

pssp, [14](#)

done\_editing

pssp::Datasheet, [44](#)

pssp::InputManager, [57](#)

double\_

pssp, [17](#)

doubles

pssp::SheetManager, [75](#)

draw\_cell

pssp::Datasheet, [44](#)

draw\_generic\_cell

pssp::Datasheet, [45](#)

draw\_header\_cell

- pssp::Datasheet, [46](#)
- e
  - pssp, [14](#)
- edit\_chars
  - pssp::datasheet, [24](#)
- edit\_col
  - pssp::Datasheet, [50](#)
- edit\_row
  - pssp::Datasheet, [50](#)
- evdp
  - pssp, [13](#)
- evel
  - pssp, [13](#)
- event\_callback
  - pssp::Datasheet, [47](#)
- event\_callback2
  - pssp::Datasheet, [47](#)
- evla
  - pssp, [14](#)
- evlo
  - pssp, [14](#)
- f
  - pssp, [14](#)
- Field
  - pssp, [13](#)
- field\_info
  - pssp, [17](#)
- field\_num
  - pssp, [19](#)
- float\_
  - pssp, [17](#)
- floats
  - pssp::SheetManager, [75](#)
- flush\_
  - pssp::ConsoleSink< Mutex >, [39](#)
- font
  - pssp::datasheet::Cell, [36](#)
- font\_size
  - pssp::datasheet, [25](#)
- full\_box
  - pssp::datasheet::Cell, [36](#)
- gcarc
  - pssp, [14](#)
- get
  - pssp::SheetManager, [71](#)
- get\_bool
  - pssp::SheetManager, [72](#)
- get\_double
  - pssp::SheetManager, [72](#)
- get\_float
  - pssp::SheetManager, [72](#)
- get\_int
  - pssp::SheetManager, [73](#)
- get\_string
  - pssp::SheetManager, [73](#)
- gridspace\_
  - pssp::MainWindow, [68](#)
- header\_height
  - pssp::datasheet::Spec, [76](#)
- header\_width
  - pssp::datasheet::Spec, [76](#)
- height
  - pssp::about, [22](#)
  - pssp::datasheet::Spec, [77](#)
  - pssp::structs::Geometry, [52](#)
  - pssp::welcome, [27](#)
- hide
  - pssp::InputManager, [57](#)
- ibody
  - pssp, [15](#)
- idep
  - pssp, [15](#)
- ievreg
  - pssp, [15](#)
- ievtyp
  - pssp, [15](#)
- iftyp
  - pssp, [15](#)
- iinst
  - pssp, [15](#)
- imagsrc
  - pssp, [15](#)
- imagtyp
  - pssp, [15](#)
- input\_cb
  - pssp::InputManager, [57](#)
- input\_float
  - pssp::InputManager, [59](#)
- input\_int
  - pssp::InputManager, [59](#)
- input\_manager
  - pssp::Datasheet, [50](#)
- input\_string
  - pssp::InputManager, [59](#)
- InputManager
  - pssp::InputManager, [55](#)
- int\_
  - pssp, [17](#)
- ints
  - pssp::SheetManager, [75](#)
- igual
  - pssp, [15](#)
- istreg
  - pssp, [15](#)
- isynth
  - pssp, [15](#)
- iztype
  - pssp, [15](#)
- ka
  - pssp, [15](#)
- kcmpnm
  - pssp, [15](#)

kdatrd  
    pssp, 15  
kevnrm  
    pssp, 15  
kf  
    pssp, 15  
khole  
    pssp, 15  
kinst  
    pssp, 15  
knetwk  
    pssp, 15  
ko  
    pssp, 15  
kstnm  
    pssp, 15  
kt0  
    pssp, 15  
kt1  
    pssp, 15  
kt2  
    pssp, 15  
kt3  
    pssp, 15  
kt4  
    pssp, 15  
kt5  
    pssp, 15  
kt6  
    pssp, 15  
kt7  
    pssp, 15  
kt8  
    pssp, 15  
kt9  
    pssp, 15  
kuser0  
    pssp, 15  
kuser1  
    pssp, 15  
kuser2  
    pssp, 15  
  
lcalda  
    pssp, 15  
left\_box\_  
    pssp::StatusBar, 79  
leven  
    pssp, 15  
list\_  
    pssp::MainWindow, 68  
logger  
    pssp::MainWindow, 68  
lovrok  
    pssp, 15  
lpspol  
    pssp, 15  
mag  
    pssp, 13  
main\_window  
    pssp::Application, 34  
MainWindow  
    pssp::MainWindow, 63  
make\_menu  
    pssp::MainWindow, 64  
make\_tty  
    pssp::MainWindow, 65  
max\_chars  
    pssp::datasheet, 25  
max\_col  
    pssp::Datasheet, 50  
max\_row  
    pssp::Datasheet, 50  
menu  
    pssp::MainWindow, 68  
menu\_height  
    pssp::mw, 25  
message  
    pssp::AboutWindow, 32  
    pssp::WelcomeWindow, 84  
message\_  
    pssp::AboutWindow, 32  
    pssp::WelcomeWindow, 84  
middle\_box\_  
    pssp::StatusBar, 79  
minimum\_x  
    pssp::mw, 25  
minimum\_y  
    pssp::mw, 26  
modified  
    pssp::InputManager, 59  
  
name  
    pssp::trace\_info, 80  
name\_  
    pssp::MainWindow, 68  
nevid  
    pssp, 14  
norid  
    pssp, 14  
npts  
    pssp, 14  
nsnpts  
    pssp, 14  
nvhdr  
    pssp, 14  
nwfid  
    pssp, 15  
nxsize  
    pssp, 15  
nysize  
    pssp, 15  
nzhour  
    pssp, 14  
nzjday  
    pssp, 14  
nzmin

- pssp, 14
- nzmsec
  - pssp, 14
- nzsec
  - pssp, 14
- nzyear
  - pssp, 14
- o
  - pssp, 14
- odelta
  - pssp, 13
- okay\_button
  - pssp::AboutWindow, 32
- okay\_cb
  - pssp::AboutWindow, 31
- Passive-source Seismic-Processing, 1
- prevent\_escape
  - pssp::MainWindow, 66
- pssp, 11
  - a, 14
  - az, 14
  - b, 14
  - baz, 14
  - bool\_, 17
  - cmpaz, 14
  - cmpinc, 14
  - ConsoleSink\_mt, 13
  - ConsoleSink\_st, 13
  - data1, 15
  - data2, 15
  - delta, 14
  - depmax, 13
  - depmen, 14
  - depmin, 13
  - dist, 14
  - double\_, 17
  - e, 14
  - evdp, 13
  - evel, 13
  - evla, 14
  - evlo, 14
  - f, 14
  - Field, 13
  - field\_info, 17
  - field\_num, 19
  - float\_, 17
  - gcarc, 14
  - ibody, 15
  - idep, 15
  - ievreg, 15
  - ievtyp, 15
  - iftyp, 15
  - iinst, 15
  - imagsrc, 15
  - imagtyp, 15
  - int\_, 17
  - igual, 15
  - istreg, 15
  - isynth, 15
  - iztype, 15
  - ka, 15
  - kcmpnm, 15
  - kdatrd, 15
  - kevm, 15
  - kf, 15
  - khole, 15
  - kinst, 15
  - knetwk, 15
  - ko, 15
  - kstnm, 15
  - kt0, 15
  - kt1, 15
  - kt2, 15
  - kt3, 15
  - kt4, 15
  - kt5, 15
  - kt6, 15
  - kt7, 15
  - kt8, 15
  - kt9, 15
  - kuser0, 15
  - kuser1, 15
  - kuser2, 15
  - lcalda, 15
  - leven, 15
  - lovrok, 15
  - lpspol, 15
  - mag, 13
  - nevid, 14
  - norid, 14
  - npts, 14
  - nsnpts, 14
  - nvhdr, 14
  - nwfid, 15
  - nxsize, 15
  - nysize, 15
  - nzhour, 14
  - nzjday, 14
  - nzmin, 14
  - nzmsec, 14
  - nzsec, 14
  - nzyear, 14
  - o, 14
  - odelta, 13
  - resp0, 13
  - resp1, 13
  - resp2, 13
  - resp3, 13
  - resp4, 13
  - resp5, 13
  - resp6, 13
  - resp7, 13
  - resp8, 13
  - resp9, 13
  - sb, 14



- sdelta, 14
- stdp, 13
- stel, 13
- stla, 14
- stlo, 14
- string\_, 17
- t0, 14
- t1, 14
- t2, 14
- t3, 14
- t4, 14
- t5, 14
- t6, 14
- t7, 14
- t8, 14
- t9, 14
- Type, 17
- type\_names, 21
- user0, 13
- user1, 13
- user2, 13
- user3, 14
- user4, 14
- user5, 14
- user6, 14
- user7, 14
- user8, 14
- user9, 14
- xmaximum, 14
- xminimum, 14
- ymaximum, 14
- yminimum, 14
- pssp::about, 21
  - button\_height, 22
  - button\_width, 22
  - height, 22
  - text\_height, 22
  - text\_width, 22
  - width, 22
- pssp::AboutWindow, 29
  - AboutWindow, 31
  - message, 32
  - message\_, 32
  - okay\_button, 32
  - okay\_cb, 31
- pssp::Application, 33
  - Application, 33
  - main\_window, 34
  - welcome\_window, 34
- pssp::ConsoleSink< Mutex >, 37
  - ConsoleSink, 39
  - flush\_, 39
  - sink\_it\_, 39
  - tty\_, 39
- pssp::constants, 23
  - sac\_bool, 23
  - sac\_data, 23
  - sac\_double, 23
  - sac\_float, 23
  - sac\_int, 23
  - sac\_string, 24
- pssp::Datasheet, 40
  - check\_button, 50
  - Datasheet, 43
  - done\_editing, 44
  - draw\_cell, 44
  - draw\_generic\_cell, 45
  - draw\_header\_cell, 46
  - edit\_col, 50
  - edit\_row, 50
  - event\_callback, 47
  - event\_callback2, 47
  - input\_manager, 50
  - max\_col, 50
  - max\_row, 50
  - set\_value\_hide, 48
  - sheet\_manager, 50
  - start\_editing, 49
- pssp::datasheet, 24
  - cell\_buffer, 24
  - edit\_chars, 24
  - font\_size, 25
  - max\_chars, 25
- pssp::datasheet::Cell, 34
  - alignment, 36
  - box\_color, 36
  - box\_type, 36
  - font, 36
  - full\_box, 36
  - text, 36
  - text\_box, 36
  - text\_color, 37
- pssp::datasheet::Spec, 76
  - header\_height, 76
  - header\_width, 76
  - height, 77
  - width, 77
- pssp::InputManager, 54
  - cleanup, 56
  - clear, 56
  - done\_editing, 57
  - hide, 57
  - input\_cb, 57
  - input\_float, 59
  - input\_int, 59
  - input\_string, 59
  - InputManager, 55
  - modified, 59
  - start\_editing, 58
  - value, 58
  - visible, 58
- pssp::MainWindow, 59
  - about\_cb, 64
  - about\_window\_, 67
  - append\_tty, 64
  - datasheet\_, 67

- debug\_tty, 68
- gridspace\_, 68
- list\_, 68
- logger, 68
- MainWindow, 63
- make\_menu, 64
- make\_tty, 65
- menu, 68
- name\_, 68
- prevent\_escape, 66
- quit\_cb, 66
- show\_about, 67
- sink, 68
- status\_bar\_, 69
- pssp::mw, 25
  - menu\_height, 25
  - minimum\_x, 25
  - minimum\_y, 26
- pssp::SheetManager, 69
  - bools, 75
  - cols, 71
  - doubles, 75
  - floats, 75
  - get, 71
  - get\_bool, 72
  - get\_double, 72
  - get\_float, 72
  - get\_int, 73
  - get\_string, 73
  - ints, 75
  - resize\_data, 73
  - rows, 74
  - set, 74
  - SheetManager, 71
  - strings, 75
- pssp::StatusBar, 77
  - left\_box\_, 79
  - middle\_box\_, 79
  - right\_box\_, 79
  - StatusBar, 78
- pssp::structs, 26
- pssp::structs::Geometry, 51
  - height, 52
  - width, 52
  - x\_pos, 52
  - y\_pos, 52
- pssp::structs::Grid, 52
  - col, 53
  - col\_span, 53
  - row, 53
  - row\_span, 53
- pssp::trace\_info, 79
  - array\_col, 80
  - col, 80
  - name, 80
  - type, 80
- pssp::welcome, 26
  - button\_height, 27
  - button\_width, 27
  - height, 27
  - text\_height, 27
  - text\_width, 27
  - width, 27
- pssp::WelcomeWindow, 81
  - continue\_button, 84
  - continue\_cb, 84
  - message, 84
  - message\_, 84
  - WelcomeWindow, 83
- quit\_cb
  - pssp::MainWindow, 66
- resize\_data
  - pssp::SheetManager, 73
- resp0
  - pssp, 13
- resp1
  - pssp, 13
- resp2
  - pssp, 13
- resp3
  - pssp, 13
- resp4
  - pssp, 13
- resp5
  - pssp, 13
- resp6
  - pssp, 13
- resp7
  - pssp, 13
- resp8
  - pssp, 13
- resp9
  - pssp, 13
- right\_box\_
  - pssp::StatusBar, 79
- row
  - pssp::structs::Grid, 53
- row\_span
  - pssp::structs::Grid, 53
- rows
  - pssp::SheetManager, 74
- sac\_bool
  - pssp::constants, 23
- sac\_data
  - pssp::constants, 23
- sac\_double
  - pssp::constants, 23
- sac\_float
  - pssp::constants, 23
- sac\_int
  - pssp::constants, 23
- sac\_string
  - pssp::constants, 24
- sb

- pssp, 14
- sdelta
  - pssp, 14
- set
  - pssp::SheetManager, 74
- set\_value\_hide
  - pssp::Datasheet, 48
- sheet\_manager
  - pssp::Datasheet, 50
- SheetManager
  - pssp::SheetManager, 71
- show\_about
  - pssp::MainWindow, 67
- sink
  - pssp::MainWindow, 68
- sink\_it\_
  - pssp::ConsoleSink< Mutex >, 39
- start\_editing
  - pssp::Datasheet, 49
  - pssp::InputManager, 58
- status\_bar\_
  - pssp::MainWindow, 69
- StatusBar
  - pssp::StatusBar, 78
- stdp
  - pssp, 13
- stel
  - pssp, 13
- stla
  - pssp, 14
- stlo
  - pssp, 14
- string\_
  - pssp, 17
- strings
  - pssp::SheetManager, 75
- t0
  - pssp, 14
- t1
  - pssp, 14
- t2
  - pssp, 14
- t3
  - pssp, 14
- t4
  - pssp, 14
- t5
  - pssp, 14
- t6
  - pssp, 14
- t7
  - pssp, 14
- t8
  - pssp, 14
- t9
  - pssp, 14
- text
  - pssp::datasheet::Cell, 36
- text\_box
  - pssp::datasheet::Cell, 36
- text\_color
  - pssp::datasheet::Cell, 37
- text\_height
  - pssp::about, 22
  - pssp::welcome, 27
- text\_width
  - pssp::about, 22
  - pssp::welcome, 27
- Todo List, 3
- tty\_
  - pssp::ConsoleSink< Mutex >, 39
- Type
  - pssp, 17
- type
  - pssp::trace\_info, 80
- type\_names
  - pssp, 21
- user0
  - pssp, 13
- user1
  - pssp, 13
- user2
  - pssp, 13
- user3
  - pssp, 14
- user4
  - pssp, 14
- user5
  - pssp, 14
- user6
  - pssp, 14
- user7
  - pssp, 14
- user8
  - pssp, 14
- user9
  - pssp, 14
- value
  - pssp::InputManager, 58
- visible
  - pssp::InputManager, 58
- welcome\_window
  - pssp::Application, 34
- WelcomeWindow
  - pssp::WelcomeWindow, 83
- width
  - pssp::about, 22
  - pssp::datasheet::Spec, 77
  - pssp::structs::Geometry, 52
  - pssp::welcome, 27
- x\_pos
  - pssp::structs::Geometry, 52
- xmaximum

- pssp, [14](#)
- xminimum
  - pssp, [14](#)
- y\_pos
  - pssp::structs::Geometry, [52](#)
- ymaximum
  - pssp, [14](#)
- yminimum
  - pssp, [14](#)