Passive-source Seismic-processing (PsSp)

0.1.0

Generated by Doxygen 1.9.8

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 PsSp/Utility/Constants.hpp

Namespace pssp

Move structs from other files to this file.

Namespace pssp::about

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

Move structs to PsSp/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:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

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 , iqual , isynth ,
  imagtyp , imagsrc , ibody , leven ,
  lpspol , lovrok , lcalda , kstnm ,
  kevnm , 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 (Datasheet) 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	
iqual	
isynth	
imagtyp	
imagsrc	
ibody	
leven	
lpspol	
lovrok	
lcalda	
kstnm	
kevnm	
khole	
ko	
ka	
kt0	
kt1	
kt2	
kt3	
kt4	
kt5	
kt6	
kt7	
kt8	
kt9	
kf	
kuser0	
kuser1	
kuser2	
kcmpnm	
knetwk	
kdatrd	
kinst	
data1	
data2	

```
                        {
  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,
  iqual,
  isynth,
  imagtyp,
  imagsrc,
  ibody,
  leven,
  lpspol,
  lovrok,
  lcalda,
  kstnm,
  kevnm,
  khole,
  ko,
  ka,
  kt0,
  kt1,
  kt2,
  kt3,
  kt4,
  kt5,
  kt6,
  kt7,
  kt8,
  kt9,
  kf,
  kuser0,
  kuser1,
  kuser2,
  kcmpnm,
  knetwk,
  kdatrd,
  kinst,
  data1,
  data2
};
```

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.

```
                  {
  string_,
  int_,
  float_,
  double_,
  bool_,
};
```

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 Datasheet.

Todo Merge field_num into this.

```
                                                    {
    // Floats
    {Field::depmin, {0, 0, "DepMin", Type::float_}},
    {Field::depmax, {1, 1, "DepMax", Type::float_}},
    {Field::odelta, {2, 2, "ODelta", Type::float_}},
    {Field::resp0, {3, 3, "Resp0", Type::float_}},
    {Field::resp1, {4, 4, "Resp1", Type::float_}},
    {Field::resp2, {5, 5, "Resp2", Type::float_}},
    {Field::resp3, {6, 6, "Resp3", Type::float_}},
    {Field::resp4, {7, 7, "Resp4", Type::float_}},
    {Field::resp5, {8, 8, "Resp5", Type::float_}},
    {Field::resp6, {9, 9, "Resp6", Type::float_}},
    {Field::resp7, {10, 10, "Resp7", Type::float_}},
    {Field::resp8, {11, 11, "Resp8", Type::float_}},
    {Field::resp9, {12, 12, "Resp9", Type::float_}},
    {Field::stel, {13, 13, "StEl", Type::float_}},
    {Field::stdp, {14, 14, "StDp", Type::float_}},
    {Field::evel, {15, 15, "EvEl", Type::float_}},
    {Field::evdp, {16, 16, "EvDp", Type::float_}},
    {Field::mag, {17, 17, "Mag", Type::float_}},
    {Field::user0, {18, 18, "User0", Type::float_}},
    {Field::user1, {19, 19, "User1", Type::float_}},
    {Field::user2, {20, 20, "User2", Type::float_}},
    {Field::user3, {21, 21, "User3", Type::float_}},
    {Field::user4, {21, 22, "User4", Type::float_}},
    {Field::user5, {23, 23, "User5", Type::float_}},
    {Field::user6, {24, 24, "User6", Type::float_}},
    {Field::user7, {25, 25, "User7", Type::float_}},
    {Field::user8, {26, 26, "User8", Type::float_}},
    {Field::user9, {27, 27, "User9", Type::float_}},
    {Field::dist, {28, 28, "Dist", Type::float_}},
    {Field::az, {29, 29, "Az", Type::float_}},
    {Field::baz, {30, 30, "BAz", Type::float_}},
    {Field::gcarc, {31, 31, "GCArc", Type::float_}},
    {Field::depmen, {32, 32, "DepMen", Type::float_}},
    {Field::cmpaz, {33, 33, "CmpAz", Type::float_}},
    {Field::cmpinc, {34, 34, "CmpInc", Type::float_}},
    {Field::xminimum, {35, 35, "XMinimum", Type::float_}},
    {Field::xmaximum, {36, 36, "XMaximum", Type::float_}},
    {Field::yminimum, {37, 37, "YMinimum", Type::float_}},
    {Field::ymaximum, {38, 38, "YMaximum", Type::float_}},
    // Doubles
    {Field::delta, {39, 0, "Delta", Type::double_}},
    {Field::b, {40, 1, "B", Type::double_}},
    {Field::e, {41, 2, "E", Type::double_}},
    {Field::o, {42, 3, "O", Type::double_}},
    {Field::a, {43, 4, "A", Type::double_}},
    {Field::t0, {44, 5, "T0", Type::double_}},
    {Field::t1, {45, 6, "T1", Type::double_}},
    {Field::t2, {46, 7, "T2", Type::double_}},
    {Field::t3, {47, 8, "T3", Type::double_}},
    {Field::t4, {48, 9, "T4", Type::double_}},
    {Field::t5, {49, 10, "T5", Type::double_}},
    {Field::t6, {50, 11, "T6", Type::double_}},
    {Field::t7, {51, 12, "T7", Type::double_}},
    {Field::t8, {52, 13, "T8", Type::double_}},
    {Field::t9, {53, 14, "T9", Type::double_}},
    {Field::f, {54, 15, "F", Type::double_}},
    {Field::stla, {55, 16, "StLa", Type::double_}},
    {Field::stlo, {56, 17, "StLo", Type::double_}},
    {Field::evla, {57, 18, "EvLa", Type::double_}},
    {Field::evlo, {58, 19, "EvLo", Type::double_}},
    {Field::sb, {59, 20, "sB", Type::double_}},
    {Field::sdelta, {60, 21, "sDelta", Type::double_}},
    // Ints
    {Field::nzyear, {61, 0, "nzYear", Type::int_}},
    {Field::nzjday, {62, 1, "nzJDay", Type::int_}},
    {Field::nzhour, {63, 2, "nzHour", Type::int_}},
    {Field::nzmin, {64, 3, "nzMin", Type::int_}},
    {Field::nzsec, {65, 4, "nzSec", Type::int_}},
    {Field::nzmsec, {66, 5, "nzMSec", Type::int_}},
    {Field::nvhdr, {67, 6, "nVHdr", Type::int_}},
    {Field::norid, {68, 7, "nOrID", Type::int_}},
    {Field::nevid, {69, 8, "nEvID", Type::int_}},
    {Field::npts, {70, 9, "nPts", Type::int_}},
    {Field::nsnpts, {71, 10, "nsnPts", Type::int_}},
    {Field::nwfid, {72, 11, "nWfID", Type::int_}},
    {Field::nxsize, {73, 12, "nXSize", Type::int_}},
    {Field::nysize, {74, 13, "nYSize", Type::int_}},
    {Field::iftype, {75, 14, "iFType", Type::int_}},
    {Field::idep, {76, 15, "iDep", Type::int_}},
    {Field::iztype, {77, 16, "iZType", Type::int_}},
    {Field::iinst, {78, 17, "iInst", Type::int_}},
    {Field::istreg, {79, 18, "iStReg", Type::int_}},
    {Field::ievreg, {80, 19, "iEvReg", Type::int_}},
    {Field::ievtyp, {81, 20, "iEvTyp", Type::int_}},
    {Field::iqual, {82, 21, "iQual", Type::int_}},
    {Field::isynth, {83, 22, "iSynth", Type::int_}},
    {Field::imagtyp, {84, 23, "iMagTyp", Type::int_}},
    {Field::imagsrc, {85, 24, "iMagSrc", Type::int_}},
    {Field::ibody, {86, 25, "iBody", Type::int_}},
    // Bools
    {Field::leven, {87, 0, "lEven", Type::bool_}},
    {Field::lpspol, {88, 1, "lPsPol", Type::bool_}},
    {Field::lovrok, {89, 2, "lOvrOK", Type::bool_}},
    {Field::lcalda, {90, 3, "lCalDA", Type::bool_}},
    // Strings
    {Field::kstnm, {91, 0, "kStNm", Type::string_}},
    {Field::kevnm, {92, 1, "kEvNm", Type::string_}},
    {Field::khole, {93, 2, "kHole", Type::string_}},
    {Field::ko, {94, 3, "kO", Type::string_}},
    {Field::ka, {95, 4, "kA", Type::string_}},
    {Field::kt0, {96, 5, "kT0", Type::string_}},
    {Field::kt1, {97, 6, "kT1", Type::string_}},
    {Field::kt2, {98, 7, "kT2", Type::string_}},
    {Field::kt3, {99, 8, "kT3", Type::string_}},
    {Field::kt4, {100, 9, "kT4", Type::string_}},
    {Field::kt5, {101, 10, "kT5", Type::string_}},
    {Field::kt6, {102, 11, "kT6", Type::string_}},
    {Field::kt7, {103, 12, "kT7", Type::string_}},
    {Field::kt8, {104, 13, "kT8", Type::string_}},
    {Field::kt9, {105, 14, "kT9", Type::string_}},
    {Field::kf, {106, 15, "kF", Type::string_}},
    {Field::kuser0, {107, 16, "kUser0", Type::string_}},
    {Field::kuser1, {108, 17, "kUser1", Type::string_}},
    {Field::kuser2, {109, 18, "kUser2", Type::string_}},
    {Field::kcmpnm, {110, 19, "kCmpNm", Type::string_}},
    {Field::knetwk, {111, 20, "kNetwk", Type::string_}},
    {Field::kdatrd, {112, 21, "kDatRd", Type::string_}},
    {Field::kinst, {113, 22, "kInst", Type::string_}},
    // Data
    {Field::data1, {114, 0, "Data1", Type::int_}},
    {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

```
                                             {// Floats
                                                {0, Field::depmin},
                                                {1, Field::depmax},
                                                {2, Field::odelta},
                                                {3, Field::resp0},
                                                {4, Field::resp1},
                                                {5, Field::resp2},
                                                {6, Field::resp3},
                                                {7, Field::resp4},
                                                {8, Field::resp5},
                                                {9, Field::resp6},
                                                {10, Field::resp7},
                                                {11, Field::resp8},
                                                {12, Field::resp9},
                                                {13, Field::stel},
                                                {14, Field::stdp},
                                                {15, Field::evel},
                                                {16, Field::evdp},
                                                {17, Field::mag},
                                                {18, Field::user0},
                                                {19, Field::user1},
                                                {20, Field::user2},
                                                {21, Field::user3},
                                                {22, Field::user4},
                                                {23, Field::user5},
                                                {24, Field::user6},
                                                {25, Field::user7},
                                                {26, Field::user8},
                                                {27, Field::user9},
                                                {28, Field::dist},
                                                {29, Field::az},
                                                {30, Field::baz},
                                                {31, Field::gcarc},
                                                {32, Field::depmen},
                                                {33, Field::cmpaz},
                                                {34, Field::cmpinc},
                                                {35, Field::xminimum},
                                                {36, Field::xmaximum},
                                                {37, Field::yminimum},
                                                {38, Field::ymaximum},
                                                // Doubles
                                                {39, Field::delta},
                                                {40, Field::b},
                                                {41, Field::e},
                                                {42, Field::o},
                                                {43, Field::a},
                                                {44, Field::t0},
                                                {45, Field::t1},
                                                {46, Field::t2},
                                                {47, Field::t3},
                                                {48, Field::t4},
                                                {49, Field::t5},
                                                {50, Field::t6},
                                                {51, Field::t7},
                                                {52, Field::t8},
                                                {53, Field::t9},
                                                {54, Field::f},
                                                {55, Field::stla},
                                                {56, Field::stlo},
                                                {57, Field::evla},
                                                {58, Field::evlo},
                                                {59, Field::sb},
                                                {60, Field::sdelta},
                                                // Ints
                                                {61, Field::nzyear},
                                                {62, Field::nzjday},
                                                {63, Field::nzhour},
                                                {64, Field::nzmin},
                                                {65, Field::nzsec},
                                                {66, Field::nzmsec},
                                                {67, Field::nvhdr},
                                                {68, Field::norid},
                                                {69, Field::nevid},
                                                {70, Field::npts},
                                                {71, Field::nsnpts},
                                                {72, Field::nwfid},
                                                {73, Field::nxsize},
                                                {74, Field::nysize},
                                                {75, Field::iftype},
                                                {76, Field::idep},
                                                {77, Field::iztype},
                                                {78, Field::iinst},
                                                {79, Field::istreg},
                                                {80, Field::ievreg},
                                                {81, Field::ievtyp},
                                                {82, Field::iqual},
                                                {83, Field::isynth},
                                                {84, Field::imagtyp},
                                                {85, Field::imagsrc},
                                                {86, Field::ibody},
                                                // Bools
                                                {87, Field::leven},
                                                {88, Field::lpspol},
                                                {89, Field::lovrok},
                                                {90, Field::lcalda},
                                                // Strings
                                                {91, Field::kstnm},
                                                {92, Field::kevnm},
                                                {93, Field::khole},
                                                {94, Field::ko},
                                                {95, Field::ka},
                                                {96, Field::kt0},
                                                {97, Field::kt1},
                                                {98, Field::kt2},
                                                {99, Field::kt3},
                                                {100, Field::kt4},
                                                {101, Field::kt5},
                                                {102, Field::kt6},
                                                {103, Field::kt7},
                                                {104, Field::kt8},
                                                {105, Field::kt9},
                                                {106, Field::kf},
                                                {107, Field::kuser0},
                                                {108, Field::kuser1},
                                                {109, Field::kuser2},
                                                {110, Field::kcmpnm},
                                                {111, Field::knetwk},
                                                {112, Field::kdatrd},
                                                {113, Field::kinst},
                                                // Data
                                                {114, Field::data1},
                                                {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

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

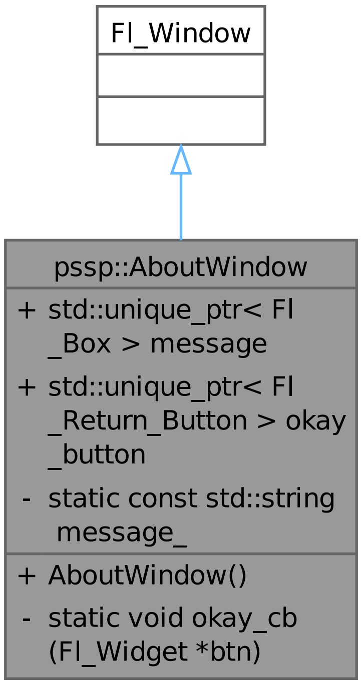
Class Documentation

7.1 pssp::AboutWindow Class Reference

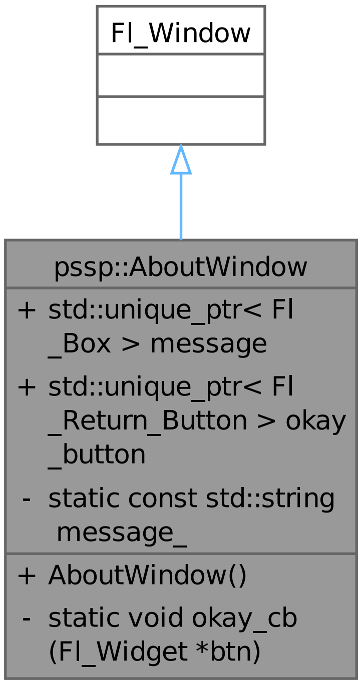
Class to provide the About Window.

`#include <About.hpp>`

Inheritance diagram for pssp::AboutWindow:


Fl_Window
pssp::AboutWindow
+ std::unique_ptr< Fl_Box > message
+ std::unique_ptr< Fl_Return_Button > okay_button
- static const std::string message_
+ AboutWindow()
- static void okay_cb (Fl_Widget *btn)

Collaboration diagram for pssp::AboutWindow:


Fl_Window
pssp::AboutWindow
+ std::unique_ptr< Fl_Box > message
+ std::unique_ptr< Fl_Return_Button > okay_button
- static const std::string message_
+ AboutWindow()
- static void okay_cb (Fl_Widget *btn)

Public Member Functions

- AboutWindow ()

  AboutWindow constructor.

Public Attributes

- std::unique_ptr< Fl_Box > message {}
- std::unique_ptr< Fl_Return_Button > okay_button {}

Static Private Member Functions

- static void okay_cb (Fl_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()

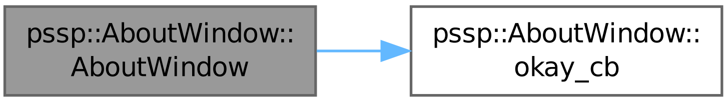
```
pssp::AboutWindow::AboutWindow ( )
```

AboutWindow constructor.

This creates the AboutWindow object with all the specified sizes from the pssp::about namespaces and centers the window.

```
                         : Fl_Window(0, 0, 0, 0, "About") {
  this->begin();
  structs::Geometry geo{};
  Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
  geo.x_pos = ((geo.width - about::width) / 2);
  geo.y_pos = ((geo.height - about::height) / 2);
  this->resize(geo.x_pos, geo.y_pos, about::width, about::height);
  this->box(FL_BORDER_BOX);
  set_modal();
  message = std::make_unique<Fl_Box>(about::width - about::text_width, 0,
                                      about::text_width, about::text_height);
  okay_button = std::make_unique<Fl_Return_Button>(
      (about::width - about::button_width) / 2, about::text_height,
      about::button_width, about::button_height, "Okay");
  message->label(message_.c_str());
  message->align(FL_ALIGN_CENTER);
  okay_button->callback(okay_cb);
  this->end();
}
```

Here is the call graph for this function:



7.1.3 Member Function Documentation

7.1.3.1 okay_cb()

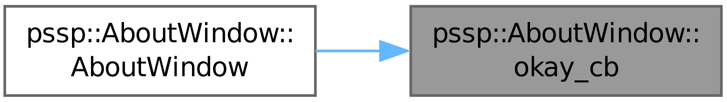
```
void pssp::AboutWindow::okay_cb (
            Fl_Widget * btn )  [static], [private]
```

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:

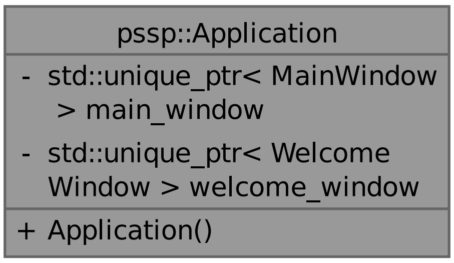
- include/PsSp/Windows/About.hpp
- src/Windows/About.cpp

7.2 pssp::Application Class Reference

Main application class.

`#include <Application.hpp>`

Collaboration diagram for pssp::Application:


pssp::Application
- std::unique_ptr< MainWindow > main_window
- std::unique_ptr< Welcome Window > welcome_window
+ 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.

```
                         {
  main_window = std::make_unique<MainWindow>();
  main_window->show();
  welcome_window = std::make_unique<WelcomeWindow>();
  welcome_window->show();
  spdlog::trace("Application ready.");
}
```

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:

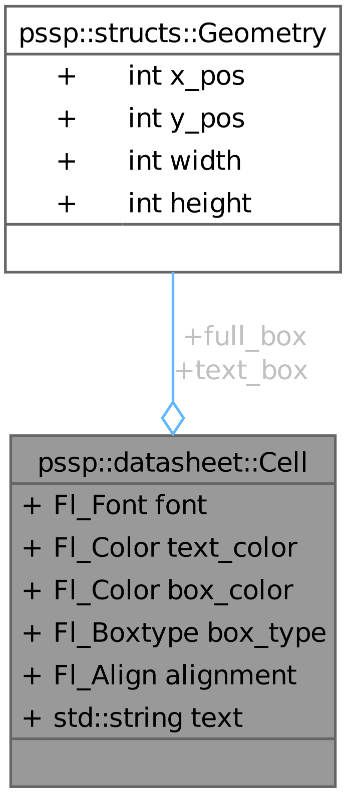
- include/PsSp/Application/Application.hpp
- src/Application/Application.cpp

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).
- Fl_Font font {FL_HELVETICA}

  Font used for Cell content.
- Fl_Color text_color {FL_BLACK}

  Color of Cell text.
- Fl_Color box_color {FL_GRAY}

  Color of cell background.
- Fl_Boxtype box_type {FL_THIN_UP_BOX}

  Type of cell drawing.
- Fl_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:

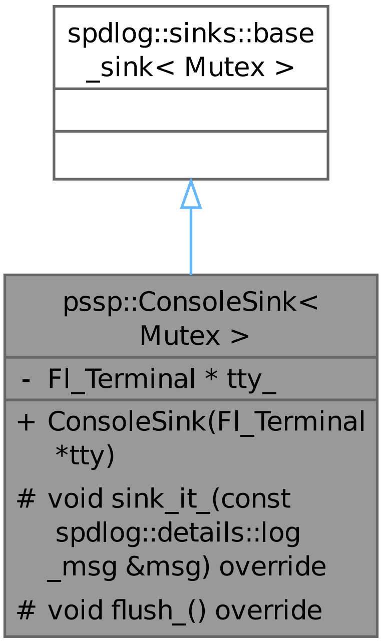
- include/PsSp/Widgets/Datasheet.hpp

7.4 pssp::ConsoleSink< Mutex > Class Template Reference

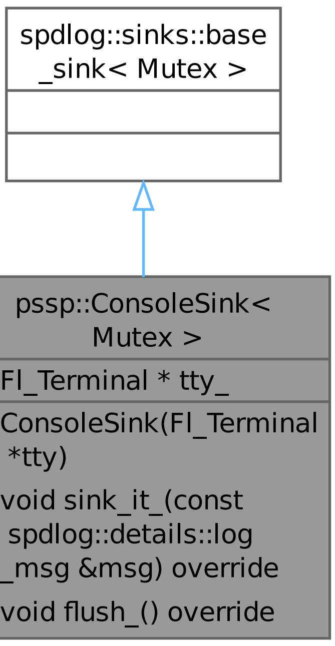
Sink (receiver) of log messages for PsSp console.

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

Inheritance diagram for pssp::ConsoleSink< Mutex >:


spdlog::sinks::base_sink< Mutex >
pssp::ConsoleSink< Mutex >
- Fl_Terminal * tty_
+ ConsoleSink(Fl_Terminal *tty)
# void sink_it_(const spdlog::details::log_msg &msg) override
# void flush_() override

Collaboration diagram for pssp::ConsoleSink< Mutex >:


spdlog::sinks::base_sink< Mutex >
pssp::ConsoleSink< Mutex >
- Fl_Terminal * tty_
+ ConsoleSink(Fl_Terminal *tty)
# void sink_it_(const spdlog::details::log_msg &msg) override
# void flush_() override

Public Member Functions

- ConsoleSink (Fl_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) Fl_Terminal.

Private Attributes

- Fl_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.

```
{ 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.

```
{ 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.

```
                                                        {
    // log_msg is a struct containing the log entry info like level, timestamp,
    // msg.raw contains the pre-formatted log

    // If needed (very likely, but not madatory), the sink formats the message
    spdlog::memory_buf_t formatted{};
    spdlog::sinks::base_sink<Mutex>::formatter_->format(msg, formatted);
    tty_->append(fmt::to_string(formatted).c_str());
  }
```

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 {};
```

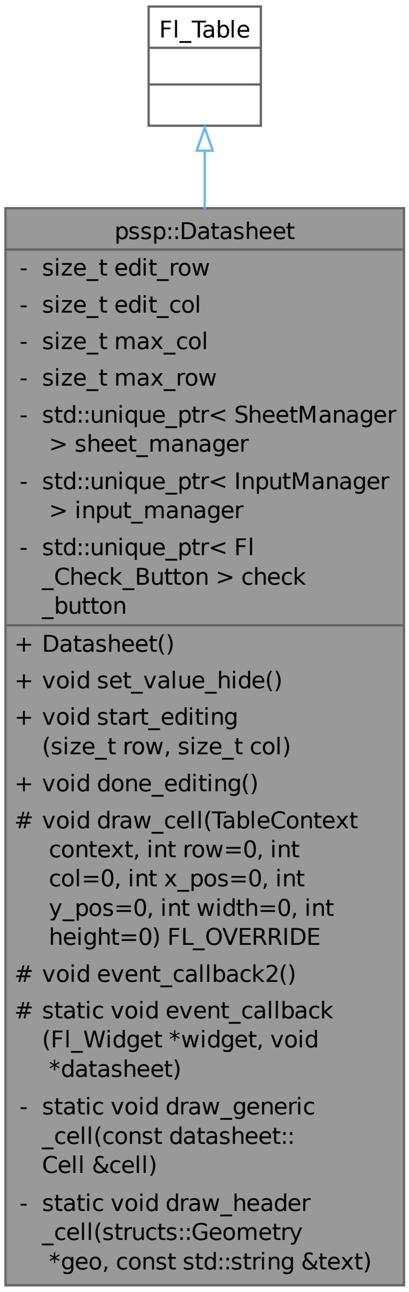
The documentation for this class was generated from the following file:

- include/PsSp/Logging/ConsoleSink.hpp

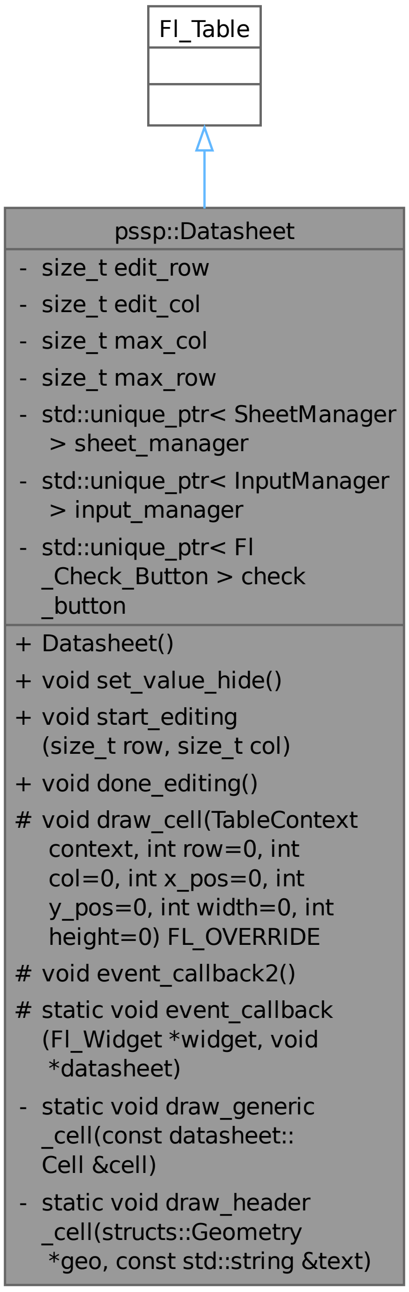
7.5 pssp::Datasheet Class Reference

`#include <Datasheet.hpp>`

Inheritance diagram for pssp::Datasheet:


Fl_Table
pssp::Datasheet
- size_t edit_row
- size_t edit_col
- size_t max_col
- size_t max_row
- std::unique_ptr< SheetManager > sheet_manager
- std::unique_ptr< InputManager > input_manager
- std::unique_ptr< Fl_Check_Button > check_button
+ Datasheet()
+ void set_value_hide()
+ void start_editing (size_t row, size_t col)
+ void done_editing()
# 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 void event_callback (Fl_Widget *widget, void *datasheet)
- static void draw_generic_cell(const datasheet::Cell &cell)
- static void draw_header_cell(structs::Geometry *geo, const std::string &text)

Collaboration diagram for pssp::Datasheet:


Fl_Table
pssp::Datasheet
- size_t edit_row
- size_t edit_col
- size_t max_col
- size_t max_row
- std::unique_ptr< SheetManager > sheet_manager
- std::unique_ptr< InputManager > input_manager
- std::unique_ptr< Fl_Check_Button > check_button
+ Datasheet()
+ void set_value_hide()
+ void start_editing (size_t row, size_t col)
+ void done_editing()
# 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 void event_callback (Fl_Widget *widget, void *datasheet)
- static void draw_generic_cell(const datasheet::Cell &cell)
- static void draw_header_cell(structs::Geometry *geo, const std::string &text)

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 ( )
```

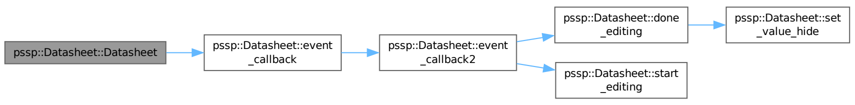
Datasheet constructor.

Builds the datasheet using the constants from the pssp::datasheet namespace.

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

```
    max_row = static_cast<size_t>(sheet_manager->rows());
    constexpr datasheet::Spec spec{25, 25, 25, 70};
    row_header(1);
    row_header_width(spec.header_width);
    row_height_all(spec.height);
    rows(static_cast<int>(max_row));
    col_header(1);
    col_header_height(spec.header_height);
    col_width_all(spec.width);
    cols(static_cast<int>(max_col));
    row_resize(1);
    col_resize(1);
    set_selection(0, 0, 0, 0);
    this->end();
    spdlog::trace("Done making \033[1mDatasheet\033[0m.");
}
```

Here is the call graph for this function:

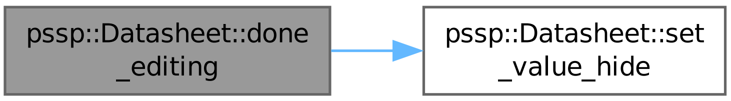

pssp::Datasheet::Datasheet
pssp::Datasheet::event_callback
pssp::Datasheet::event_callback2
pssp::Datasheet::done_editing
pssp::Datasheet::set_value_hide
pssp::Datasheet::start_editing

7.5.2 Member Function Documentation

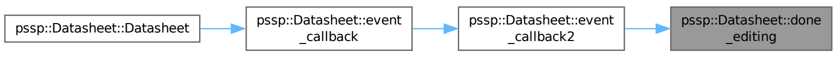
7.5.2.1 done_editing()

```
void pssp::Datasheet::done_editing ( )
                               {
    if (input_manager->visible() || input_manager->modified) {
      set_value_hide();
      edit_row = 0;
      edit_col = 0;
    }
}
```

Here is the call graph for this function:


pssp::Datasheet::done_editing
pssp::Datasheet::set_value_hide

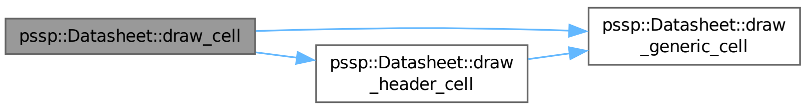
Here is the caller graph for this function:


pssp::Datasheet::Datasheet
pssp::Datasheet::event_callback
pssp::Datasheet::event_callback2
pssp::Datasheet::done_editing

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]
                                                                {
  // NOLINTEND(bugprone-easily-swappable-parameters)
  switch (context) {
  case CONTEXT_COL_HEADER: {
    structs::Geometry geo{x_pos, y_pos, width, height};
    draw_header_cell(
        &geo, field_info.at(field_num.at(static_cast<size_t>(col))).name);
  } break;
  case CONTEXT_ROW_HEADER: {
    structs::Geometry geo{x_pos, y_pos, width, height};
    draw_header_cell(&geo, std::to_string(row + 1));
  } break;
  case CONTEXT_CELL: {
    // This needs to be refactored
    datasheet::Cell cell{};
    cell.full_box = {x_pos, y_pos, width, height};
    cell.text_box = {x_pos + datasheet::cell_buffer,
                     y_pos + datasheet::cell_buffer,
                     width - (2 * datasheet::cell_buffer),
                     height - (2 * datasheet::cell_buffer)};
    const Field &field{field_num.at(static_cast<size_t>(col))};
    const trace_info &info{field_info.at(field)};
    if (info.type == Type::string_) {
      cell.text = sheet_manager->get_string(static_cast<size_t>(row), field);
    } else if (info.type == Type::int_) {
      std::ostringstream oss{};
      oss « sheet_manager->get_int(static_cast<size_t>(row), field);
      cell.text = oss.str();
    } else if (info.type == Type::float_) {
      std::ostringstream oss{};
      oss « sheet_manager->get_float(static_cast<size_t>(row), field);
      cell.text = oss.str();
    } else if (info.type == Type::double_) {
      std::ostringstream oss{};
      oss « sheet_manager->get_double(static_cast<size_t>(row), field);
      cell.text = oss.str();
    } else if (info.type == Type::bool_) {
      std::ostringstream oss{};
      oss « sheet_manager->get_bool(static_cast<size_t>(row), field);
      cell.text = oss.str();
    }
    cell.box_color = ((is_selected(row, col) != 0) ? FL_YELLOW : FL_WHITE);
    cell.alignment = FL_ALIGN_RIGHT;
    draw_generic_cell(cell);
  } break;
  default:
    return;
  }
}
```

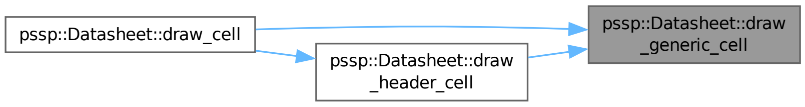
Here is the call graph for this function:


pssp::Datasheet::draw_cell
pssp::Datasheet::draw_header_cell
pssp::Datasheet::draw_generic_cell

7.5.2.3 draw_generic_cell()

```
void pssp::Datasheet::draw_generic_cell (
            const datasheet::Cell & cell )  [static], [private]
                                                                        {
  fl_font(cell.font, datasheet::font_size);
  fl_draw_box(cell.box_type, cell.full_box.x_pos, cell.full_box.y_pos,
              cell.full_box.width, cell.full_box.height, cell.box_color);
  fl_push_clip(cell.text_box.x_pos, cell.text_box.y_pos, cell.text_box.width,
               cell.text_box.height);
  fl_color(cell.text_color);
  fl_draw(cell.text.c_str(), cell.text_box.x_pos, cell.text_box.y_pos,
          cell.text_box.width, cell.text_box.height, cell.alignment);
  fl_pop_clip();
}
```

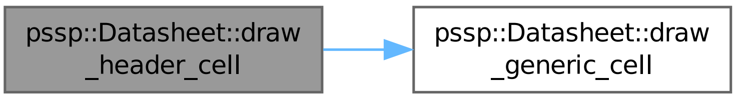
Here is the caller graph for this function:


pssp::Datasheet::draw_cell
pssp::Datasheet::draw _header_cell
pssp::Datasheet::draw _generic_cell

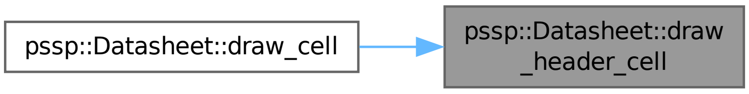
7.5.2.4 draw_header_cell()

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

Here is the call graph for this function:


pssp::Datasheet::draw _header_cell
pssp::Datasheet::draw _generic_cell

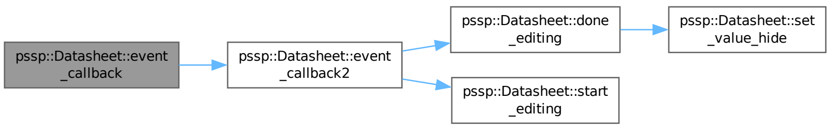
Here is the caller graph for this function:



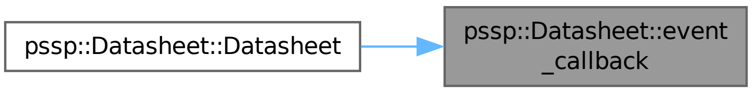
7.5.2.5 event_callback()

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

Here is the call graph for this function:



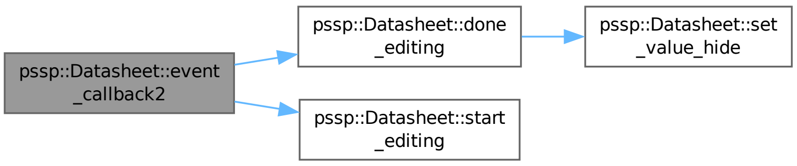
Here is the caller graph for this function:



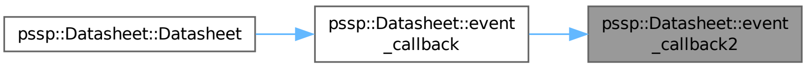
7.5.2.6 event_callback2()

```
void pssp::Datasheet::event_callback2 ( )  [protected]
                                  {
  int row{callback_row()};
  int col{callback_col()};
  TableContext context{callback_context()};
  switch (context) {
  case CONTEXT_CELL: {
    switch (Fl::event()) {
    case FL_PUSH:
      start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
      break;
    case FL_KEYBOARD:
      done_editing();
      if (Fl::event_state() == FL_COMMAND) {
        parent()->take_focus();
      } else if (datasheet::edit_chars.find(Fl::e_text[0]) !=
                 std::string::npos) {
        start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
      }
      break;
    default:
      break;
    }
  } break;
  case CONTEXT_TABLE:
  case CONTEXT_ROW_HEADER:
  case CONTEXT_COL_HEADER:
    done_editing();
    break;
  default:
    return;
  }
}
```

Here is the call graph for this function:


pssp::Datasheet::event_callback2
pssp::Datasheet::done_editing
pssp::Datasheet::set_value_hide
pssp::Datasheet::start_editing

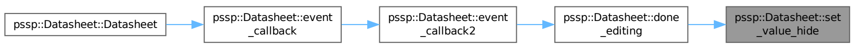
Here is the caller graph for this function:


pssp::Datasheet::Datasheet
pssp::Datasheet::event_callback
pssp::Datasheet::event_callback2

7.5.2.7 set_value_hide()

```
void pssp::Datasheet::set_value_hide ( )
                                  {
  const Field &field{field_num.at(edit_col)};
  const trace_info &info{field_info.at(field)};
  switch (info.type) {
  case Type::string_:
    sheet_manager->set(edit_row, field, input_manager->value());
    break;
  case Type::int_:
    if (!input_manager->value().empty()) {
      sheet_manager->set(edit_row, field, std::stoi(input_manager->value()));
    } else {
      sheet_manager->set(edit_row, field, 0);
    }
    break;
  case Type::float_:
    if (!input_manager->value().empty()) {
      sheet_manager->set(edit_row, field, std::stof(input_manager->value()));
    } else {
      sheet_manager->set(edit_row, field, 0.0F);
    }
    break;
  case Type::double_:
    if (!input_manager->value().empty()) {
      sheet_manager->set(edit_row, field, std::stod(input_manager->value()));
    } else {
      sheet_manager->set(edit_row, field, 0.0);
    }
    break;
  case Type::bool_:
    // This is just junk for prototyping
    sheet_manager->set(edit_row, field, !input_manager->value().empty());
    break;
  default:
    break;
  }
  input_manager->cleanup();
  input_manager->modified = false;
  window()->cursor(FL_CURSOR_DEFAULT);  // deals with disappearing cursor
}
```

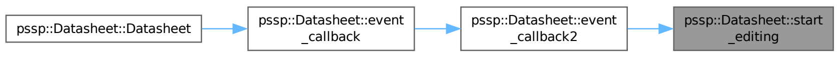
Here is the caller graph for this function:


pssp::Datasheet::Datasheet
pssp::Datasheet::event_callback
pssp::Datasheet::event_callback2
pssp::Datasheet::done_editing
pssp::Datasheet::set_value_hide

7.5.2.8 start_editing()

```
void pssp::Datasheet::start_editing (
               size_t row,
               size_t col )
                                                    {
  edit_row = row;
  edit_col = col;
  set_selection(static_cast<int>(row), static_cast<int>(col),
                static_cast<int>(row), static_cast<int>(col));
  structs::Geometry geo{};
  find_cell(CONTEXT_CELL, static_cast<int>(row), static_cast<int>(col),
            geo.x_pos, geo.y_pos, geo.width, geo.height);
  // Need to refactor
  const Field &field{field_num.at(col)};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::string_) {
    input_manager->start_editing(info, geo,
                                 sheet_manager->get_string(row, field));
  } else if (info.type == Type::int_) {
    std::ostringstream oss{};
    oss « sheet_manager->get_int(row, field);
    input_manager->start_editing(info, geo, oss.str());
  } else if (info.type == Type::float_) {
    std::ostringstream oss{};
    oss « sheet_manager->get_float(row, field);
    input_manager->start_editing(info, geo, oss.str());
  } else if (info.type == Type::double_) {
    std::ostringstream oss{};
    oss « sheet_manager->get_double(row, field);
    input_manager->start_editing(info, geo, oss.str());
  } else if (info.type == Type::bool_) {
    std::ostringstream oss{};
    oss « sheet_manager->get_bool(row, field);
    input_manager->start_editing(info, geo, oss.str());
  }
}
```

Here is the caller graph for this function:


pssp::Datasheet::Datasheet
pssp::Datasheet::event_callback
pssp::Datasheet::event_callback2
pssp::Datasheet::start_editing

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 {0};
```

7.5.3.3 edit_row

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

Row of most recently edited cell.

```
00152 {0};
```

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 {0};
```

7.5.3.6 max_row

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

Maximum number of rows in the Datasheet.

```
00158 {0};
```

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:

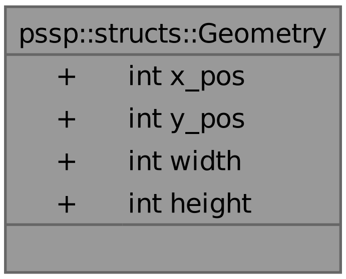
- include/PsSp/Widgets/Datasheet.hpp
- src/Widgets/Datasheet.cpp

7.6 pssp::structs::Geometry Struct Reference

FLTK Geometry handling struct.

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

Collaboration diagram for pssp::structs::Geometry:



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:

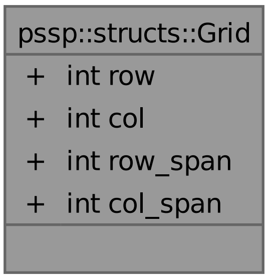
- include/PsSp/Utility/Structs.hpp

7.7 pssp::structs::Grid Struct Reference

FLTK Grid definition struct.

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

Collaboration diagram for pssp::structs::Grid:



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 (Fl_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:

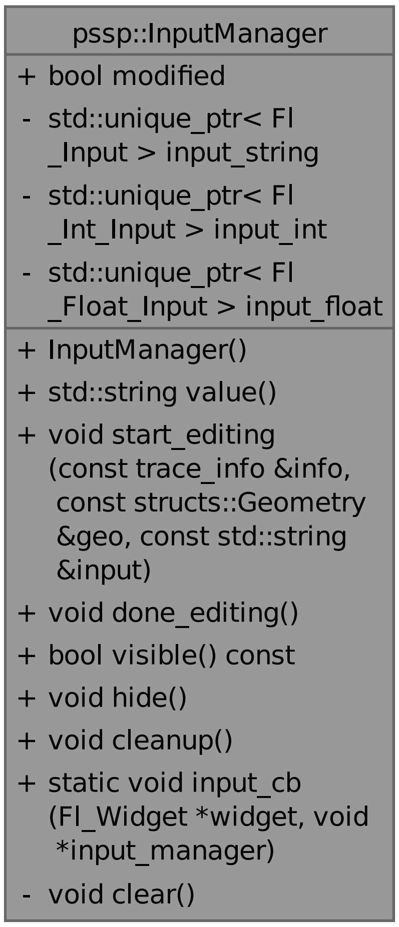
- include/PsSp/Utility/Structs.hpp

7.8 pssp::InputManager Class Reference

Manager of user-input.

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

Collaboration diagram for pssp::InputManager:


pssp::InputManager
+ bool modified
- std::unique_ptr< Fl _Input > input_string
- std::unique_ptr< Fl _Int_Input > input_int
- std::unique_ptr< Fl _Float_Input > input_float
+ 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 void input_cb (Fl_Widget *widget, void *input_manager)
- void clear()

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.

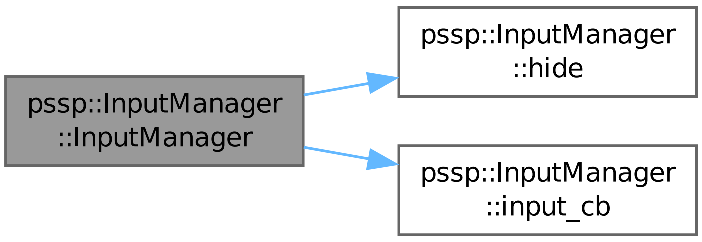
Todo

7.8.2 Constructor & Destructor Documentation

7.8.2.1 InputManager()

```
pssp::InputManager::InputManager ( )
                              {
    input_string = std::make_unique<Fl_Input>(0, 0, 0, 0);
    input_int = std::make_unique<Fl_Int_Input>(0, 0, 0, 0);
    input_float = std::make_unique<Fl_Float_Input>(0, 0, 0, 0);
    hide();
    input_string->callback(input_cb, reinterpret_cast<void *>(this));
    input_int->callback(input_cb, reinterpret_cast<void *>(this));
    input_float->callback(input_cb, reinterpret_cast<void *>(this));
    input_string->when(FL_WHEN_ENTER_KEY_ALWAYS);
    input_int->when(FL_WHEN_ENTER_KEY_ALWAYS);
    input_float->when(FL_WHEN_ENTER_KEY_ALWAYS);
    input_string->maximum_size(40);
    input_int->maximum_size(40);
    input_float->maximum_size(40);
    input_string->color(FL_YELLOW);
    input_int->color(FL_RED);
    input_float->color(FL_GREEN);
}
```

Here is the call graph for this function:

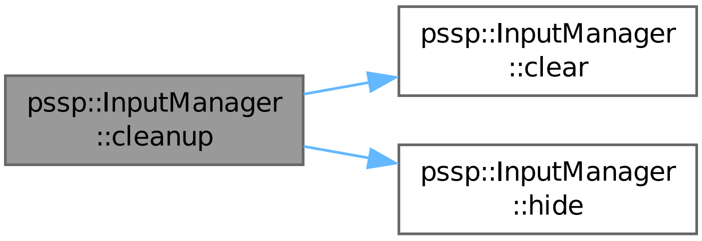

pssp::InputManager ::InputManager
pssp::InputManager ::hide
pssp::InputManager ::input_cb

7.8.3 Member Function Documentation

7.8.3.1 cleanup()

```
void pssp::InputManager::cleanup ( )
                             {
    clear();
    hide();
}
```

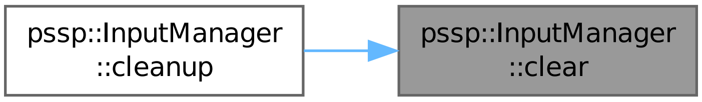
Here is the call graph for this function:


pssp::InputManager ::cleanup
pssp::InputManager ::clear
pssp::InputManager ::hide

7.8.3.2 clear()

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

Here is the caller graph for this function:



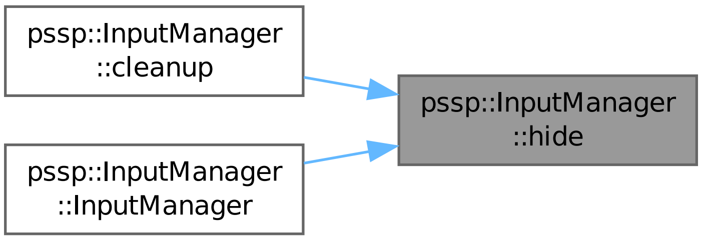
7.8.3.3 done_editing()

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

7.8.3.4 hide()

```
void pssp::InputManager::hide ( )
                          {
  input_string->hide();
  input_int->hide();
  input_float->hide();
}
```

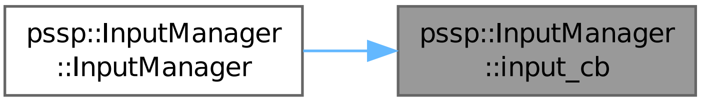
Here is the caller graph for this function:


pssp::InputManager ::cleanup
pssp::InputManager ::InputManager
pssp::InputManager ::hide

7.8.3.5 input_cb()

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

Here is the caller graph for this function:


pssp::InputManager ::InputManager
pssp::InputManager ::input_cb

7.8.3.6 start_editing()

```
void pssp::InputManager::start_editing (
             const trace_info & info,
             const structs::Geometry & geo,
             const std::string & input )
                                                          {
  if (info.type == Type::string_) {
    input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
    input_string->value(input.c_str());
    input_string->insert_position(0, static_cast<int>(input.size()));
    input_string->show();
    input_string->take_focus();
  } else if (info.type == Type::int_) {
    input_int->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
    input_int->value(input.c_str());
    input_int->insert_position(0, static_cast<int>(input.size()));
    input_int->show();
    input_int->take_focus();
  } else if (info.type == Type::float_) {
    input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
    input_float->value(input.c_str());
    input_float->insert_position(0, static_cast<int>(input.size()));
    input_float->show();
    input_float->take_focus();
  } else if (info.type == Type::double_) {
    input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
    input_float->value(input.c_str());
    input_float->insert_position(0, static_cast<int>(input.size()));
    input_float->show();
    input_float->take_focus();
  } else if (info.type == Type::bool_) {
    input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
    input_string->value(input.c_str());
    input_string->insert_position(0, static_cast<int>(input.size()));
    input_string->show();
    input_string->take_focus();
  }
}
```

7.8.3.7 value()

```
std::string pssp::InputManager::value ( )
                                 {
  std::string result{};
  // Which one is being used? They're empty after cleanup
  // so only the used one is full
  if (!std::string(input_string->value()).empty()) {
    result = input_string->value();
  } else if (!std::string(input_int->value()).empty()) {
    result = input_int->value();
  } else {
    result = input_float->value();
  }
  return result;
}
```

7.8.3.8 visible()

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

7.8.4 Member Data Documentation

7.8.4.1 input_float

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

7.8.4.2 input_int

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

7.8.4.3 input_string

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

7.8.4.4 modified

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

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

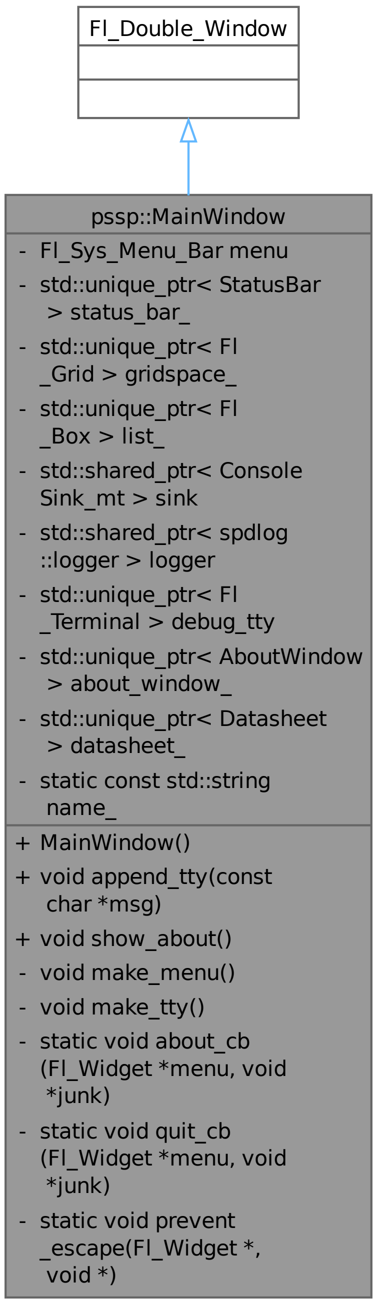
- include/PsSp/Managers/InputManager.hpp
- src/Managers/InputManager.cpp

7.9 pssp::MainWindow Class Reference

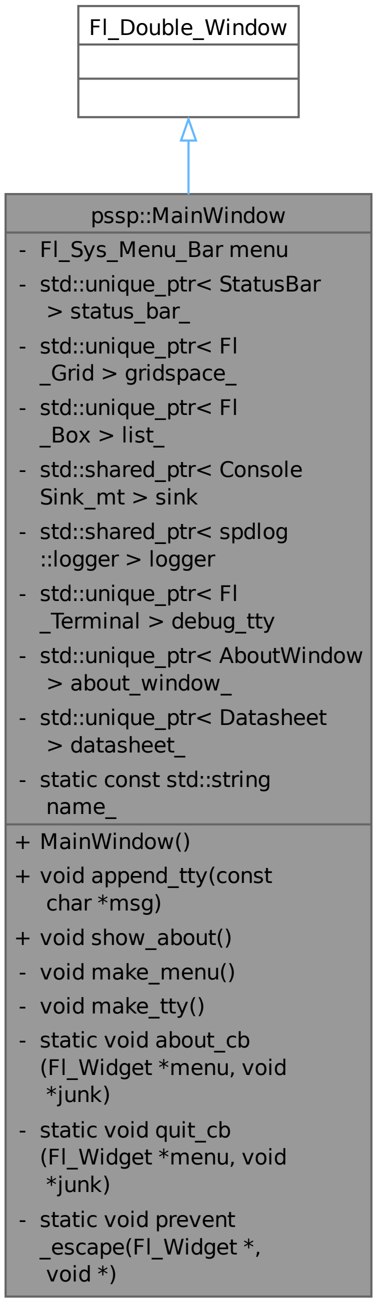
Class to provide the Main Window.

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

Inheritance diagram for pssp::MainWindow:


Fl_Double_Window
pssp::MainWindow
- Fl_Sys_Menu_Bar menu
- std::unique_ptr< StatusBar > status_bar_
- std::unique_ptr< Fl _Grid > gridspace_
- std::unique_ptr< Fl _Box > list_
- std::shared_ptr< Console Sink_mt > sink
- std::shared_ptr< spdlog ::logger > logger
- std::unique_ptr< Fl _Terminal > debug_tty
- std::unique_ptr< AboutWindow > about_window_
- std::unique_ptr< Datasheet > datasheet_
- static const std::string name_
+ MainWindow()
+ void append_tty(const char *msg)
+ void show_about()
- void make_menu()
- void make_tty()
- static void about_cb (Fl_Widget *menu, void *junk)
- static void quit_cb (Fl_Widget *menu, void *junk)
- static void prevent _escape(Fl_Widget *, void *)

Collaboration diagram for pssp::MainWindow:


Fl_Double_Window
pssp::MainWindow
- Fl_Sys_Menu_Bar menu
- std::unique_ptr< StatusBar > status_bar_
- std::unique_ptr< Fl _Grid > gridspace_
- std::unique_ptr< Fl _Box > list_
- std::shared_ptr< Console Sink_mt > sink
- std::shared_ptr< spdlog ::logger > logger
- std::unique_ptr< Fl _Terminal > debug_tty
- std::unique_ptr< AboutWindow > about_window_
- std::unique_ptr< Datasheet > datasheet_
- static const std::string name_
+ MainWindow()
+ void append_tty(const char *msg)
+ void show_about()
- void make_menu()
- void make_tty()
- static void about_cb (Fl_Widget *menu, void *junk)
- static void quit_cb (Fl_Widget *menu, void *junk)
- static void prevent _escape(Fl_Widget *, void *)

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 (Fl_Widget ∗menu, void ∗junk)

  Callback function to show the AboutWindow.
- static void quit_cb (Fl_Widget ∗menu, void ∗junk)

  Menu/Hotkey Quit callback command.
- static void prevent_escape (Fl_Widget ∗, void ∗)

Private Attributes

- Fl_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< Fl_Grid > gridspace_ {}

  Grid to layout window components.
- std::unique_ptr< Fl_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< Fl_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()

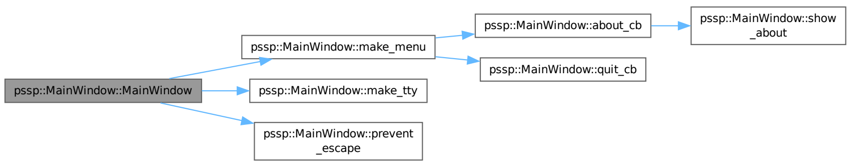
```
pssp::MainWindow::MainWindow ( )
```

MainWindow constructor.

This creates the MainWindow object with all the specified sizes from the pssp::mw namespace and maximizes the window.

```
                        : Fl_Double_Window(0, 0, name_.c_str()) {
  this->callback(prevent_escape);
  make_tty();
  spdlog::trace("Building \033[1mMainWindow\033[0m.");
  this->begin();
  resizable(this);
  // Minimum window size width/height
  this->size_range(mw::minimum_x, mw::minimum_y);
  structs::Geometry geo{};
  Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
  this->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
  make_menu();
  menu.resize(0, 0, geo.width, menu.h());
  status_bar_ = std::make_unique<StatusBar>(this->h(), this->w(), menu.h());
#if defined(__APPLE__)
  const int menu_shift{0};
#else
  const int menu_shift{menu.h()};
#endif
  gridspace_ = std::make_unique<Fl_Grid>(0, menu_shift, this->w(),
                                          this->h() - menu_shift - menu.h());
  gridspace_->begin();
  gridspace_->add(debug_tty.get());
  gridspace_->show_grid(0);  // 1 to show guide lines
  constexpr structs::Grid layout{10, 10, 1, 1};
  gridspace_->layout(layout.row, layout.col, layout.row_span, layout.col_span);
  list_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "List");
  list_->box(FL_BORDER_BOX);
  list_->color(FL_WHITE);
  datasheet_ = std::make_unique<Datasheet>();
  constexpr structs::Grid tty_grid{7, 0, 3, 10};
  gridspace_->widget(debug_tty.get(), tty_grid.row, tty_grid.col,
                     tty_grid.row_span, tty_grid.col_span);
  constexpr structs::Grid list_grid{0, 0, 7, 2};
  gridspace_->widget(list_.get(), list_grid.row, list_grid.col,
                     list_grid.row_span, list_grid.col_span);
  constexpr structs::Grid ds_grid{0, 2, 7, 8};
  gridspace_->widget(datasheet_.get(), ds_grid.row, ds_grid.col,
                     ds_grid.row_span, ds_grid.col_span);
  gridspace_->end();
  this->end();
  this->resizable(status_bar_.get());
  this->resizable(datasheet_.get());
  this->resizable(gridspace_.get());
  about_window_ = std::make_unique<AboutWindow>();
  about_window_->hide();
  spdlog::trace("Done making \033[1mMainWindow\033[0m.");
}
```

Here is the call graph for this function:


pssp::MainWindow::MainWindow
pssp::MainWindow::make_menu
pssp::MainWindow::about_cb
pssp::MainWindow::show_about
pssp::MainWindow::quit_cb
pssp::MainWindow::make_tty
pssp::MainWindow::prevent_escape

7.9.3 Member Function Documentation

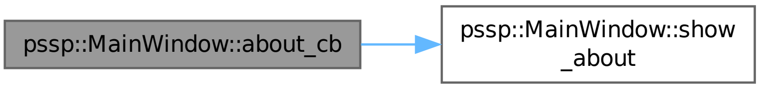
7.9.3.1 about_cb()

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

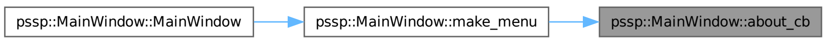
Callback function to show the AboutWindow.

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

Here is the call graph for this function:



Here is the caller graph for this function:


pssp::MainWindow::MainWindow
pssp::MainWindow::make_menu
pssp::MainWindow::about_cb

7.9.3.2 append_tty()

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

Add a message to the end of the console log.

```
{ 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.

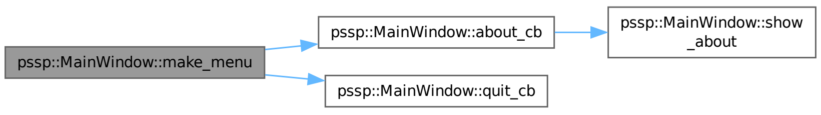
On macOS this is a system-menu bar that does not take up any window space.

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

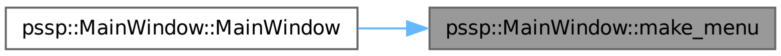
Todo Fix shallow menus that do not display on macOS (all menus must have depth).

```
                              {
    spdlog::trace("Making \033[1mMenu\033[0m.");
    // Program
    menu.add("&Program/&Quit", FL_COMMAND + 'q', quit_cb, this);
    // Project
    menu.add("&Project/&New", FL_COMMAND + 'n', nullptr, this, FL_MENU_INACTIVE);
    menu.add("&Project/&Load", FL_COMMAND + 'o', nullptr, this, FL_MENU_INACTIVE);
    menu.add("&Project/&Close", FL_COMMAND + 'c', nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Project/&Bookmark", FL_COMMAND + 'b', nullptr, this,
             FL_MENU_INACTIVE);
    // Data
    menu.add("&Data/&Add File", 0, nullptr, this, FL_MENU_INACTIVE);
    menu.add("&Data/&Add Directory", 0, nullptr, this, FL_MENU_INACTIVE);
    menu.add("&Data/&Download Data", 0, nullptr, this, FL_MENU_INACTIVE);
    // Processing
    menu.add("&Processing/&Filters/&Butterworth/&Lowpass", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Processing/&Filters/&Butterworth/&Highpass", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Processing/&Filters/&Butterworth/&Bandpass", 0, nullptr, this,
             FL_MENU_INACTIVE);
    // Plotting
    menu.add("&Plot/&Single Component/&Time-series", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Plot/&Single Component/&Spectrum/&Real-Imaginary", 0, nullptr,
             this, FL_MENU_INACTIVE);
    menu.add("&Plot/&Single Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
             this, FL_MENU_INACTIVE);
    menu.add("&Plot/&Single Component/&Spectrogram", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Plot/&Three Component/&Time-series", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Plot/&Three Component/&Spectrum/&Real-Imaginary", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Plot/&Three Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
             this, FL_MENU_INACTIVE);
    menu.add("&Plot/&Three Component/&Spectrogram", 0, nullptr, this,
             FL_MENU_INACTIVE);
    menu.add("&Plot/&Profile", 0, nullptr, this, FL_MENU_INACTIVE);
    // Settings
    menu.add("&Settings", 0, nullptr, this, FL_MENU_INACTIVE);
    // Help
    menu.add("&Help", 0, nullptr, this, FL_MENU_INACTIVE);
    // About
    menu.add("&About", 0, about_cb, this);
    spdlog::trace("Done making \033[1mMenu\033[0m.");
  }
```

Here is the call graph for this function:


pssp::MainWindow::make_menu
pssp::MainWindow::about_cb
pssp::MainWindow::show_about
pssp::MainWindow::quit_cb

Here is the caller graph for this function:



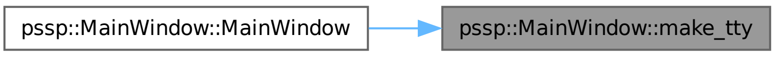
7.9.3.4 make_tty()

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

Construct and initialize the FL_Terminal log display.

```
                                {
  // Debug terminal
  debug_tty = std::make_unique<Fl_Terminal>(0, 0, 0, 0);
  sink = std::make_shared<ConsoleSink_mt>(debug_tty.get());
  logger = std::make_shared<spdlog::logger>("tty logger", sink);
  spdlog::set_default_logger(logger);
  // levels are critical, error, warn, info, debug, trace
  spdlog::set_level(spdlog::level::trace);
  spdlog::set_pattern(
      "\33[1m\33[32m[%Y-%m-%d %T]\33[33m[%l]\33[36m[thread %t]\33[0m %v");
  debug_tty->begin();
  constexpr int font_size{14};
  debug_tty->textsize(font_size);
  debug_tty->redraw_style(Fl_Terminal::NO_REDRAW);
  constexpr int num_columns{80};
  debug_tty->display_columns(num_columns);
  spdlog::trace("Logger started.");
  debug_tty->end();
  resizable();
}
```

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]
```

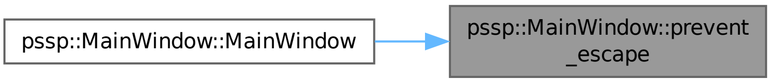
FLTK has the odd-behavior or having a built-in auto-close callback upon either the Escape key or the Q key being hit. Just immediate closure with no questions asked.

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.

```
                                                            {
  (void)caller;
  (void)data;
  if ((Fl::event() == FL_SHORTCUT) && (Fl::event_key() == FL_Escape)) {
    return;  // ignore Escape
  }
  exit(0);
}
```

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?).

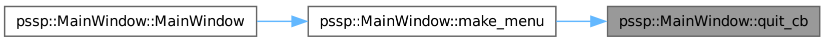
Todo Request if the user wants to save first (if unsaved work).

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

BugFix: Doesn't display when keyboard input is captured by Datasheet.

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

Here is the caller graph for this function:


pssp::MainWindow::MainWindow
pssp::MainWindow::make_menu
pssp::MainWindow::quit_cb

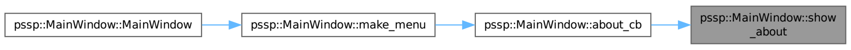
7.9.3.7 show_about()

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

Show the AboutWindow.

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

Here is the caller graph for this function:


pssp::MainWindow::MainWindow
pssp::MainWindow::make_menu
pssp::MainWindow::about_cb
pssp::MainWindow::show _about

7.9.4 Member Data Documentation

7.9.4.1 about_window_

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

The AboutWindow.

```
{};
```

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 {};
```

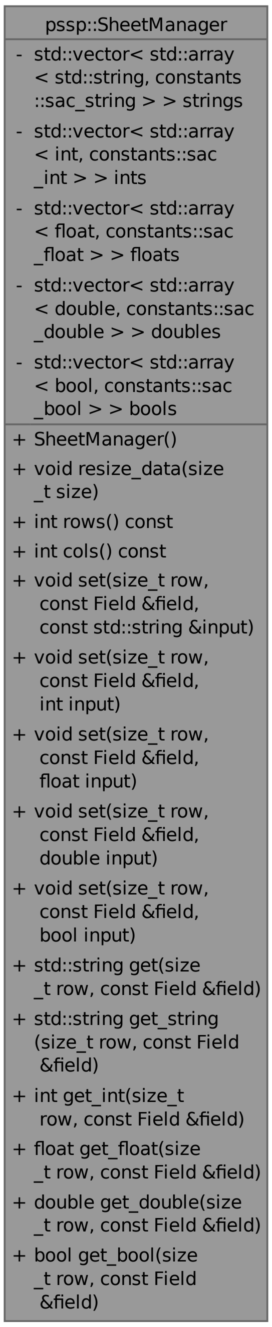
The documentation for this class was generated from the following files:

- 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
- 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
+ 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)

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

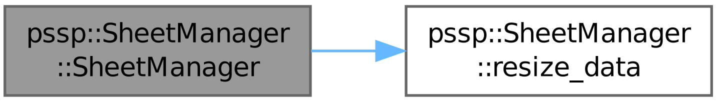
- std::vector< std::array< std::string, constants::sac_string > > strings {}
- std::vector< std::array< int, constants::sac_int > > ints {}
- std::vector< std::array< float, constants::sac_float > > floats {}
- std::vector< std::array< double, constants::sac_double > > doubles {}
- std::vector< std::array< bool, constants::sac_bool > > bools {}

7.10.1 Constructor & Destructor Documentation

7.10.1.1 SheetManager()

```
pssp::SheetManager::SheetManager ( )
{ resize_data(100); }
```

Here is the call graph for this function:


pssp::SheetManager ::SheetManager
pssp::SheetManager ::resize_data

7.10.2 Member Function Documentation

7.10.2.1 cols()

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

7.10.2.2 get()

```
std::string pssp::SheetManager::get (
            size_t row,
            const Field & field )
```

```
{
  std::string result{};
  const trace_info &info{field_info.at(field)};
  switch (info.type) {
  case Type::string_:
    break;
  case Type::int_:
    result = std::to_string(ints[row][info.array_col]);
    break;
  case Type::float_:
    result = std::to_string(floats[row][info.array_col]);
    break;
  case Type::double_:
    result = std::to_string(doubles[row][info.array_col]);
    break;
  case Type::bool_:
    result = std::to_string(static_cast<int>(bools[row][info.array_col]));
    break;
  default:
    break;
  }
  return result;
}
```

7.10.2.3 get_bool()

```
bool pssp::SheetManager::get_bool (
            size_t row,
            const Field & field )
```

```
{
  bool result{};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::bool_) {
    result = bools[row][info.array_col];
  } else {
    spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
                  type_names.at(info.type));
  }
  return result;
}
```

7.10.2.4 get_double()

```
double pssp::SheetManager::get_double (
            size_t row,
            const Field & field )
```

```
{
  double result{};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::double_) {
    result = doubles[row][info.array_col];
  } else {
    spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
                  type_names.at(info.type));
  }
  return result;
}
```

7.10.2.5 get_float()

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

```
                                                                          {
  float result{};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::float_) {
    result = floats[row][info.array_col];
  } else {
    spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
                  type_names.at(info.type));
  }
  return result;
}
```

7.10.2.6 get_int()

```
int pssp::SheetManager::get_int (
              size_t row,
              const Field & field )
```

```
                                                                      {
  int result{};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::int_) {
    result = ints[row][info.array_col];
  } else {
    spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
                  type_names.at(info.type));
  }
  return result;
}
```

7.10.2.7 get_string()

```
std::string pssp::SheetManager::get_string (
              size_t row,
              const Field & field )
```

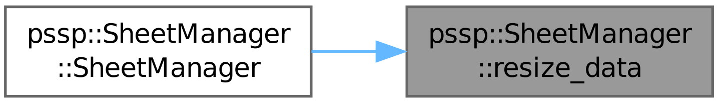
```
                                                                               {
  std::string result{};
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::string_) {
    result = strings[row][info.array_col];
  } else {
    spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
                  type_names.at(info.type));
  }
  return result;
}
```

7.10.2.8 resize_data()

```
void pssp::SheetManager::resize_data (
              size_t size )
```

```
                                                    {
  strings.resize(size);
  ints.resize(size);
  floats.resize(size);
  doubles.resize(size);
  bools.resize(size);
}
```

Here is the caller graph for this function:


pssp::SheetManager ::SheetManager
pssp::SheetManager ::resize_data

7.10.2.9 rows()

```
int pssp::SheetManager::rows ( ) const
```

```
{ 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 )
```

```
{
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::bool_) {
    bools[row][info.array_col] = input;
  } else {
    spdlog::error("Wrong type {0} inserted into field {1}.",
                  type_names.at(info.type), info.name);
  }
}
```

7.10.2.11 set() [2/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            const std::string & input )
```

```
{
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::string_) {
    strings[row][info.array_col] = input;
  } else {
    spdlog::error("Wrong type {0} inserted into field {1}.",
                  type_names.at(info.type), info.name);
  }
}
```

7.10.2.12 set() [3/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            double input )
```

```
{
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::double_) {
    doubles[row][info.array_col] = input;
  } else {
    spdlog::error("Wrong type {0} inserted into field {1}.",
                  type_names.at(info.type), info.name);
  }
}
```

7.10.2.13 set() [4/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            float input )
```

```
{
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::float_) {
    floats[row][info.array_col] = input;
  } else {
    spdlog::error("Wrong type {0} inserted into field {1}.",
                  type_names.at(info.type), info.name);
  }
}
```

7.10.2.14 set() [5/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            int input )
```

```
                                                                {
  const trace_info &info{field_info.at(field)};
  if (info.type == Type::int_) {
    ints[row][info.array_col] = input;
  } else {
    spdlog::error("Wrong type {0} inserted into field {1}.",
                  type_names.at(info.type), info.name);
  }
}
```

7.10.3 Member Data Documentation

7.10.3.1 bools

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

```
{};
```

7.10.3.2 doubles

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

```
{};
```

7.10.3.3 floats

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

```
{};
```

7.10.3.4 ints

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

```
{};
```

7.10.3.5 strings

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

```
{};
```

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

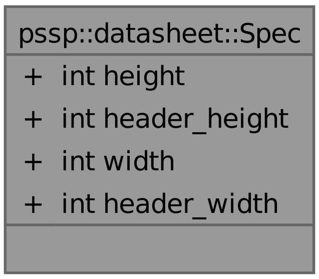
- include/PsSp/Managers/SheetManager.hpp
- src/Managers/SheetManager.cpp

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};
```

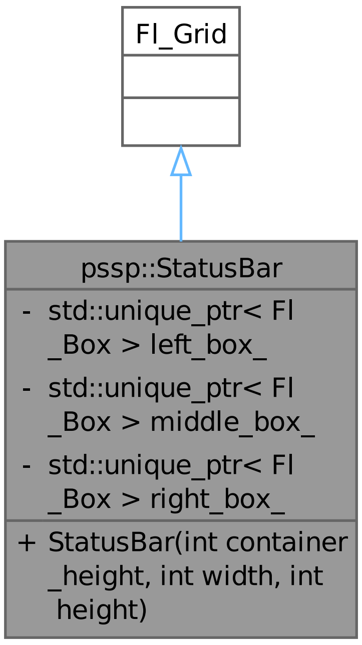
The documentation for this struct was generated from the following file:

- include/PsSp/Widgets/Datasheet.hpp

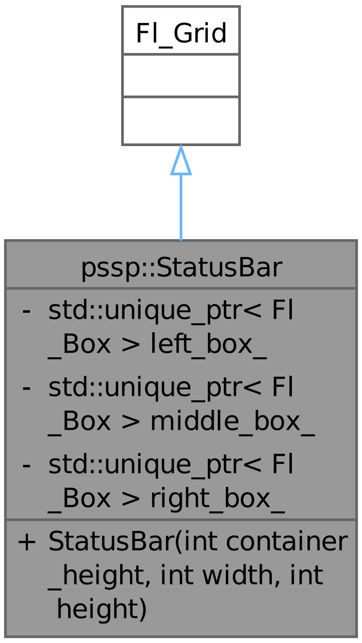
7.12 pssp::StatusBar Class Reference

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

Inheritance diagram for pssp::StatusBar:


Fl_Grid
pssp::StatusBar
- std::unique_ptr< Fl _Box > left_box_
- std::unique_ptr< Fl _Box > middle_box_
- std::unique_ptr< Fl _Box > right_box_
+ StatusBar(int container _height, int width, int height)

Collaboration diagram for pssp::StatusBar:


Fl_Grid
pssp::StatusBar
- std::unique_ptr< Fl_Box > left_box_
- std::unique_ptr< Fl_Box > middle_box_
- std::unique_ptr< Fl_Box > right_box_
+ StatusBar(int container_height, int width, int height)

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 )
    : Fl_Grid(0, container_height - height, width, height) {
  spdlog::trace("Making \033[1mStatus_Bar\033[0m.");
  this->begin();
  constexpr structs::Grid layout{1, 10, 1, 1};
  this->layout(layout.row, layout.col, layout.row_span, layout.col_span);
  left_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Left Box");
  left_box_->box(FL_BORDER_BOX);
```

```
    constexpr structs::Grid left{0, 0, 1, 2};
    this->widget(left_box_.get(), left.row, left.col, left.row_span,
                 left.col_span);
    middle_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Middle Box");
    middle_box_->box(FL_BORDER_BOX);
    constexpr structs::Grid middle{0, 2, 1, 6};
    this->widget(middle_box_.get(), middle.row, middle.col, middle.row_span,
                 middle.col_span);
    right_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Right Box");
    right_box_->box(FL_BORDER_BOX);
    constexpr structs::Grid right{0, 8, 1, 2};
    this->widget(right_box_.get(), right.row, right.col, right.row_span,
                 right.col_span);
    this->end();
    spdlog::trace("Done making \033[1mStatus_Bar\033[0m.");
}
```

7.12.2 Member Data Documentation

7.12.2.1 left_box_

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

7.12.2.2 middle_box_

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

7.12.2.3 right_box_

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

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

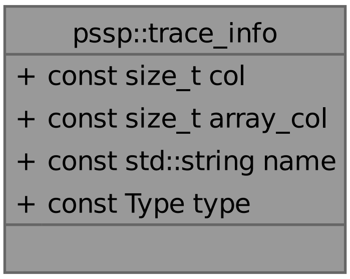
- include/PsSp/Widgets/StatusBar.hpp
- src/Widgets/StatusBar.cpp

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 {0};
```

7.13.2.2 col

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

Location of value in Datasheet.

```
00065 {0};
```

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:

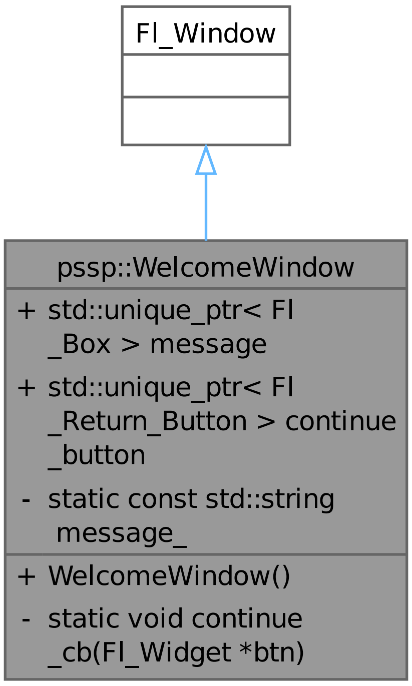
- include/PsSp/Utility/Enums.hpp

7.14 pssp::WelcomeWindow Class Reference

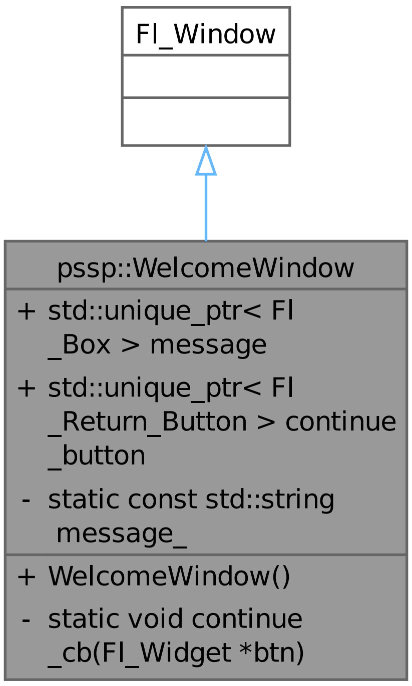
Class to provide a Welcome Window.

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

Inheritance diagram for pssp::WelcomeWindow:


Fl_Window
pssp::WelcomeWindow
+ std::unique_ptr< Fl _Box > message
+ std::unique_ptr< Fl _Return_Button > continue _button
- static const std::string message_
+ WelcomeWindow()
- static void continue _cb(Fl_Widget *btn)

Collaboration diagram for pssp::WelcomeWindow:


Fl_Window
pssp::WelcomeWindow
+ std::unique_ptr< Fl_Box > message
+ std::unique_ptr< Fl_Return_Button > continue_button
- static const std::string message_
+ WelcomeWindow()
- static void continue_cb(Fl_Widget *btn)

Public Member Functions

- WelcomeWindow ()

  WelcomeWindow constructor.

Public Attributes

- std::unique_ptr< Fl_Box > message {}

  Box to contain message.
- std::unique_ptr< Fl_Return_Button > continue_button {}

  Button to close window.

Static Private Member Functions

- static void continue_cb (Fl_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()

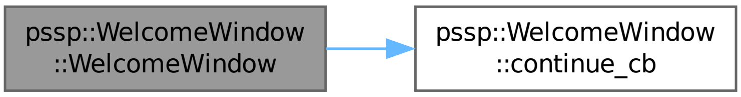
```
pssp::WelcomeWindow::WelcomeWindow ( )
```

WelcomeWindow constructor.

This creates a WelcomeWindow object with all specified sizes from the welcome namespace and centers it on the screen.

```
                              : Fl_Window(0, 0, 0, 0, "Welcome!") {
    this->begin();
    int x_start{};
    int y_start{};
    int width{};
    int height{};
    Fl::screen_work_area(x_start, y_start, width, height);
    x_start = ((width - welcome::width) / 2);
    y_start = ((height - welcome::height) / 2);
    this->resize(x_start, y_start, welcome::width, welcome::height);
    this->box(FL_BORDER_BOX);
    set_modal();
    message =
        std::make_unique<Fl_Box>((welcome::width - welcome::text_width) / 2, 0,
                                 welcome::text_width, welcome::text_height);
    continue_button = std::make_unique<Fl_Return_Button>(
        (welcome::width - welcome::button_width) / 2, welcome::text_height,
        welcome::button_width, welcome::button_height, "Continue");
    message->label(message_.c_str());
    message->align(FL_ALIGN_CENTER);
    continue_button->callback(continue_cb);
    this->end();
}
```

Here is the call graph for this function:


pssp::WelcomeWindow ::WelcomeWindow
pssp::WelcomeWindow ::continue_cb

7.14.3 Member Function Documentation

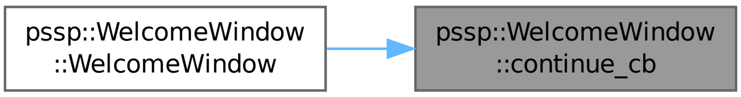
7.14.3.1 continue_cb()

```
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:


pssp::WelcomeWindow ::WelcomeWindow
pssp::WelcomeWindow ::continue_cb

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