

peakfinder-0.1 Reference Manual

Generated by Doxygen 1.3.9.1

Sun Feb 20 17:46:14 2005

Contents

1	peakfinder-0.1 Data Structure Index	1
1.1	peakfinder-0.1 Data Structures	1
2	peakfinder-0.1 File Index	3
2.1	peakfinder-0.1 File List	3
3	peakfinder-0.1 Data Structure Documentation	5
3.1	bg_poly_data Struct Reference	5
3.2	cal_fit_info Struct Reference	7
3.3	def_file_dev Struct Reference	9
3.4	peak_pearson7_data Struct Reference	10
3.5	pf_cal_fit Struct Reference	11
3.6	pf_cal_pt Struct Reference	13
3.7	pf_cal_pts Struct Reference	14
3.8	pf_data Struct Reference	16
3.9	pf_peak Struct Reference	19
3.10	pf_peak_bg Struct Reference	22
3.11	pf_peak_fit Struct Reference	24
3.12	pf_plot_params Struct Reference	26
3.13	pf_plot_range Struct Reference	29
3.14	yy_buffer_state Struct Reference	30
3.15	yyalloc Union Reference	31
3.16	YYSTYPE Union Reference	32
4	peakfinder-0.1 File Documentation	33
4.1	calibrate.c File Reference	33
4.2	cmdlex.c File Reference	36
4.3	cmdparse.c File Reference	45
4.4	cmdparse.h File Reference	61

4.5	commands.h File Reference	66
4.6	common.h File Reference	72
4.7	loaddata.c File Reference	80
4.8	main.c File Reference	82
4.9	peakfit.c File Reference	87
4.10	plotdata.c File Reference	91
4.11	printdata.c File Reference	95

Chapter 1

peakfinder-0.1 Data Structure Index

1.1 peakfinder-0.1 Data Structures

Here are the data structures with brief descriptions:

bg_poly_data (Data on the background for fitting)	5
cal_fit_info (Data to which the calibration curve is fit)	7
def_file_dev (An association between a file-name extension and a plotting device name)	9
peak_pearson7_data (Data on a peak for fitting)	10
pf_cal_fit (Calibration fit)	11
pf_cal_pt (Calibration point)	13
pf_cal_pts (Calibration points)	14
pf_data (Data set)	16
pf_peak (Peak data)	19
pf_peak_bg (Peak background fit)	22
pf_peak_fit (Peak fit)	24
pf_plot_params (Plot parameters)	26
pf_plot_range (A plot-range specification)	29
yy_buffer_state	30
yyalloc	31
YYSTYPE	32

Chapter 2

peakfinder-0.1 File Index

2.1 peakfinder-0.1 File List

Here is a list of all files with brief descriptions:

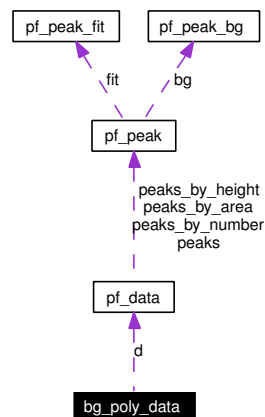
calibrate.c	33
cmdlex.c	36
cmdparse.c	45
cmdparse.h	61
commands.h	66
common.h	72
loaddata.c	80
main.c	82
peakfit.c	87
plotdata.c	91
printdata.c	95

Chapter 3

peakfinder-0.1 Data Structure Documentation

3.1 bg_poly_data Struct Reference

Collaboration diagram for bg_poly_data:



3.1.1 Detailed Description

Data on the background for fitting.

Data Fields

- `int peak`
The index of the peak for which to fit background.
- `int npts`
The number of background points.
- `int pts [NCHAN]`

The list of bins associated with the background.

- [pf_data * d](#)

The data set.

3.1.2 Field Documentation

3.1.2.1 struct [pf_data](#)* [bg_poly_data::d](#)

The data set.

3.1.2.2 int [bg_poly_data::npts](#)

The number of background points.

3.1.2.3 int [bg_poly_data::peak](#)

The index of the peak for which to fit background.

3.1.2.4 int [bg_poly_data::pts](#)[NCHAN]

The list of bins associated with the background.

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

- [peakfit.c](#)

3.2 cal_fit_info Struct Reference

3.2.1 Detailed Description

Data to which the calibration curve is fit.

Data Fields

- int `npts`
The number of points.
- double `x` [MAXCALPTS]
The channel for each point.
- double `x_err` [MAXCALPTS]
The error in the channel for each point.
- double `y` [MAXCALPTS]
The calibrated value for each point.
- int `pt` [MAXCALPTS]
The index of the point in the `pf_cal_pts` structure (ordered by value).
- int `pk` [MAXCALPTS]
The index of the peak in the `pf_data` structure.

3.2.2 Field Documentation

3.2.2.1 int `cal_fit_info::npts`

The number of points.

3.2.2.2 int `cal_fit_info::pk`[MAXCALPTS]

The index of the peak in the `pf_data` structure.

3.2.2.3 int `cal_fit_info::pt`[MAXCALPTS]

The index of the point in the `pf_cal_pts` structure (ordered by value).

3.2.2.4 double `cal_fit_info::x`[MAXCALPTS]

The channel for each point.

3.2.2.5 double `cal_fit_info::x_err`[MAXCALPTS]

The error in the channel for each point.

3.2.2.6 double `cal_fit_info::y`[MAXCALPTS]

The calibrated value for each point.

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

- [calibrate.c](#)

3.3 `def_file_dev` Struct Reference

3.3.1 Detailed Description

An association between a file-name extension and a plotting device name.

Data Fields

- `char * ext`
The file-name extension.
- `char * dev`
The plotting device name.

3.3.2 Field Documentation

3.3.2.1 `char* def_file_dev::dev`

The plotting device name.

3.3.2.2 `char* def_file_dev::ext`

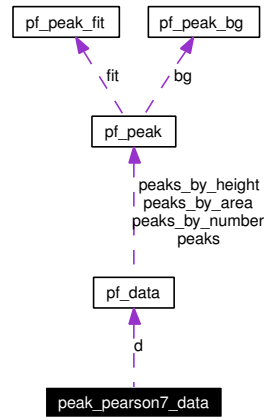
The file-name extension.

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

- `plotdata.c`

3.4 peak_pearson7_data Struct Reference

Collaboration diagram for peak_pearson7_data:



3.4.1 Detailed Description

Data on a peak for fitting.

Data Fields

- `int peak`
The index of the peak to fit.
- `pf_data * d`
The data set.

3.4.2 Field Documentation

3.4.2.1 struct `pf_data*` `peak_pearson7_data::d`

The data set.

3.4.2.2 int `peak_pearson7_data::peak`

The index of the peak to fit.

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

- `peakfit.c`

3.5 pf_cal_fit Struct Reference

```
#include <common.h>
```

3.5.1 Detailed Description

Calibration fit.

Data Fields

- char [unit](#) [MAXUNIT]
The name of the calibration unit.
- double [a](#)
The quadratic coefficient.
- double [b](#)
The linear coefficient.
- double [c](#)
The constant term.
- double [a_err](#)
Error in a.
- double [b_err](#)
Error in b.
- double [c_err](#)
Error in c.
- double [chisq](#)
chi² of the fit
- double [chisq_dof](#)
chi² per degree of freedom
- int [conv](#)
Boolean convergance flag.
- int [valid](#)
Boolean validity flag.

3.5.2 Field Documentation

3.5.2.1 double [pf_cal_fit::a](#)

The quadratic coefficient.

3.5.2.2 double [pf_cal_fit::a_err](#)

Error in a.

3.5.2.3 double [pf_cal_fit::b](#)

The linear coefficient.

3.5.2.4 double [pf_cal_fit::b_err](#)

Error in b.

3.5.2.5 double [pf_cal_fit::c](#)

The constant term.

3.5.2.6 double [pf_cal_fit::c_err](#)

Error in c.

3.5.2.7 double [pf_cal_fit::chisq](#)

χ^2 of the fit

3.5.2.8 double [pf_cal_fit::chisq_dof](#)

χ^2 per degree of freedom

3.5.2.9 int [pf_cal_fit::conv](#)

Boolean convergence flag.

3.5.2.10 char [pf_cal_fit::unit](#)[MAXUNIT]

The name of the calibration unit.

3.5.2.11 int [pf_cal_fit::valid](#)

Boolean validity flag.

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

- [common.h](#)

3.6 pf_cal_pt Struct Reference

```
#include <common.h>
```

3.6.1 Detailed Description

Calibration point.

Data Fields

- `int num`
Index of the calibration point.
- `double pt`
The value of the point.
- `double ri`
The relative intensity of the point.

3.6.2 Field Documentation

3.6.2.1 `int pf_cal_pt::num`

Index of the calibration point.

3.6.2.2 `double pf_cal_pt::pt`

The value of the point.

3.6.2.3 `double pf_cal_pt::ri`

The relative intensity of the point.

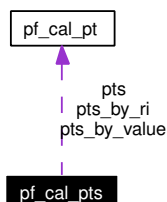
The documentation for this struct was generated from the following file:

- `common.h`

3.7 pf_cal_pts Struct Reference

```
#include <common.h>
```

Collaboration diagram for pf_cal_pts:



3.7.1 Detailed Description

Calibration points.

Data Fields

- char [unit](#) [MAXUNIT]
The name of the calibration unit.
- int [npts](#)
The number of calibration points.
- [pf_cal_pt pts](#) [MAXCALPTS]
The calibration points.
- [pf_cal_pt * pts_by_value](#) [MAXCALPTS]
Calibration points sorted by value.
- [pf_cal_pt * pts_by_ri](#) [MAXCALPTS]
Calibration points sorted by relative intensity.

3.7.2 Field Documentation

3.7.2.1 int [pf_cal_pts::npts](#)

The number of calibration points.

3.7.2.2 struct [pf_cal_pt pf_cal_pts::pts](#)[MAXCALPTS]

The calibration points.

3.7.2.3 struct [pf_cal_pt* pf_cal_pts::pts_by_ri](#)[MAXCALPTS]

Calibration points sorted by relative intensity.

3.7.2.4 `struct pf_cal_pt* pf_cal_pts::pts_by_value`[MAXCALPTS]

Calibration points sorted by value.

3.7.2.5 `char pf_cal_pts::unit`[MAXUNIT]

The name of the calibration unit.

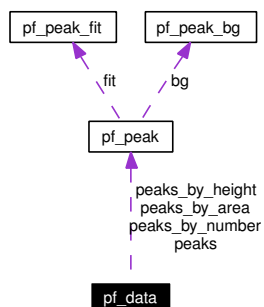
The documentation for this struct was generated from the following file:

- [common.h](#)

3.8 pf_data Struct Reference

```
#include <common.h>
```

Collaboration diagram for pf_data:



3.8.1 Detailed Description

Data set.

Data Fields

- double [data_raw](#) [NCHAN]
Raw data.
- double [data_norm](#) [NCHAN]
Normalized data.
- double [max_raw](#)
Maximum raw value.
- double [min_raw](#)
Minimum raw value.
- double [max_norm](#)
Maximum normalized value.
- double [min_norm](#)
Minimum normalized value.
- int [total_counts](#)
Total number of raw counts.
- double [total_chisq](#)
Total number of normalized counts.
- int [data_bg](#) [NCHAN]
Bin background marking.

- `int npeaks`
The number of peaks.
- `pf_peak peaks [NCHAN]`
Identified peaks.
- `pf_peak * peaks_by_number [NCHAN]`
Peaks sorted by number.
- `pf_peak * peaks_by_height [NCHAN]`
Peaks sorted by height.
- `pf_peak * peaks_by_area [NCHAN]`
Peaks sorted by area.

3.8.2 Field Documentation

3.8.2.1 `int pf_data::data_bg[NCHAN]`

Bin background marking.

3.8.2.2 `double pf_data::data_norm[NCHAN]`

Normalized data.

3.8.2.3 `double pf_data::data_raw[NCHAN]`

Raw data.

3.8.2.4 `double pf_data::max_norm`

Maximum normalized value.

3.8.2.5 `double pf_data::max_raw`

Maximum raw value.

3.8.2.6 `double pf_data::min_norm`

Minimum normalized value.

3.8.2.7 `double pf_data::min_raw`

Minimum raw value.

3.8.2.8 int [pf_data::npeaks](#)

The number of peaks.

3.8.2.9 struct [pf_peak](#) [pf_data::peaks](#)[NCHAN]

Identified peaks.

3.8.2.10 struct [pf_peak*](#) [pf_data::peaks_by_area](#)[NCHAN]

Peaks sorted by area.

3.8.2.11 struct [pf_peak*](#) [pf_data::peaks_by_height](#)[NCHAN]

Peaks sorted by height.

3.8.2.12 struct [pf_peak*](#) [pf_data::peaks_by_number](#)[NCHAN]

Peaks sorted by number.

3.8.2.13 double [pf_data::total_chisq](#)

Total number of normalized counts.

3.8.2.14 int [pf_data::total_counts](#)

Total number of raw counts.

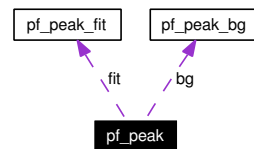
The documentation for this struct was generated from the following file:

- [common.h](#)

3.9 pf_peak Struct Reference

```
#include <common.h>
```

Collaboration diagram for pf_peak:



3.9.1 Detailed Description

Peak data.

Data Fields

- int `num`
Index of the peak.
- int `lbin`
Starting bin of the peak region.
- int `cbin`
Center (maximum) bin of the peak region.
- int `rbin`
Ending bin of the peak region.
- `pf_peak_bg` `bg`
Background fit.
- `pf_peak_fit` `fit`
Peak fit.
- double `center`
Peak center location.
- double `width`
Peak FWHM.
- double `lbound`
Starting value of the peak.
- double `rbound`
Ending value of the peak.
- double `bglbound`

Starting value of the peak background.

- double `bgrbound`

Ending value of the peak background.

- double `area`

Total peak area.

- double `height`

Maximum peak height.

3.9.2 Field Documentation

3.9.2.1 double `pf_peak::area`

Total peak area.

3.9.2.2 struct `pf_peak_bg pf_peak::bg`

Background fit.

3.9.2.3 double `pf_peak::bglbound`

Starting value of the peak background.

3.9.2.4 double `pf_peak::bgrbound`

Ending value of the peak background.

3.9.2.5 int `pf_peak::cbin`

Center (maximum) bin of the peak region.

3.9.2.6 double `pf_peak::center`

Peak center location.

3.9.2.7 struct `pf_peak_fit pf_peak::fit`

Peak fit.

3.9.2.8 double `pf_peak::height`

Maximum peak height.

3.9.2.9 int pf_peak::lbin

Starting bin of the peak region.

3.9.2.10 double pf_peak::lbound

Starting value of the peak.

3.9.2.11 int pf_peak::num

Index of the peak.

3.9.2.12 int pf_peak::rbin

Ending bin of the peak region.

3.9.2.13 double pf_peak::rbound

Ending value of the peak.

3.9.2.14 double pf_peak::width

Peak FWHM.

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

- [common.h](#)

3.10 pf_peak_bg Struct Reference

```
#include <common.h>
```

3.10.1 Detailed Description

Peak background fit.

Data Fields

- double **a**
Coefficient of the cubic term.
- double **b**
Coefficient of the quadratic term.
- double **c**
Coefficient of the linear term.
- double **d**
The constant term.
- double **a_err**
Error in a.
- double **b_err**
Error in b.
- double **c_err**
Error in c.
- double **d_err**
Error in d.
- double **chisq**
 χ^2 of the fit
- double **chisq_dof**
 χ^2 per degree of freedom
- int **conv**
Boolean convergence flag.

3.10.2 Field Documentation

3.10.2.1 double pf_peak_bg::a

Coefficient of the cubic term.

3.10.2.2 `double pf_peak_bg::a_err`

Error in a.

3.10.2.3 `double pf_peak_bg::b`

Coefficient of the quadratic term.

3.10.2.4 `double pf_peak_bg::b_err`

Error in b.

3.10.2.5 `double pf_peak_bg::c`

Coefficient of the linear term.

3.10.2.6 `double pf_peak_bg::c_err`

Error in c.

3.10.2.7 `double pf_peak_bg::chisq`

χ^2 of the fit

3.10.2.8 `double pf_peak_bg::chisq_dof`

χ^2 per degree of freedom

3.10.2.9 `int pf_peak_bg::conv`

Boolean convergence flag.

3.10.2.10 `double pf_peak_bg::d`

The constant term.

3.10.2.11 `double pf_peak_bg::d_err`

Error in d.

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

- [common.h](#)

3.11 pf_peak_fit Struct Reference

```
#include <common.h>
```

3.11.1 Detailed Description

Peak fit.

Data Fields

- double **a**
The a parameter.
- double **k**
The k parameter.
- double **x0**
The x0 parameter.
- double **m**
The m parameter.
- double **a_err**
Error in a.
- double **k_err**
Error in k.
- double **x0_err**
Error in x0.
- double **m_err**
Error in m.
- double **chisq**
chi² of the fit
- double **chisq_dof**
chi² per degree of freedom
- int **conv**
Boolean convergence flag.

3.11.2 Field Documentation

3.11.2.1 double pf_peak_fit::a

The a parameter.

3.11.2.2 `double pf_peak_fit::a_err`

Error in a.

3.11.2.3 `double pf_peak_fit::chisq`

χ^2 of the fit

3.11.2.4 `double pf_peak_fit::chisq_dof`

χ^2 per degree of freedom

3.11.2.5 `int pf_peak_fit::conv`

Boolean convergence flag.

3.11.2.6 `double pf_peak_fit::k`

The k parameter.

3.11.2.7 `double pf_peak_fit::k_err`

Error in k.

3.11.2.8 `double pf_peak_fit::m`

The m parameter.

3.11.2.9 `double pf_peak_fit::m_err`

Error in m.

3.11.2.10 `double pf_peak_fit::x0`

The x0 parameter.

3.11.2.11 `double pf_peak_fit::x0_err`

Error in x0.

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

- [common.h](#)

3.12 pf_plot_params Struct Reference

```
#include <common.h>
```

3.12.1 Detailed Description

Plot parameters.

Data Fields

- double `xstart`
Starting domain value.
- double `xend`
Ending domain value.
- int `xstartcal`
Flag indicating if xstart is in calibrated units.
- int `xendcal`
Flag indicating if xend is in calibrated units.
- int `peak`
Peak to plot.
- char * `fn`
Output file name.
- char * `dev`
Plotting device name.
- double `rot`
Number of degrees to rotate the plot.
- int `norm`
Flag indicating whether or not to plot normalized values.
- int `rescale`
Flag indicating whether or not to rescale plot.
- int `cal`
Flag indicating whether or not to plot in calibrated units.
- int `marked`
Flag indicating whether or not to mark non-background regions.
- int `annot`
Flag indicating whether or not to annotate plot with peak numbers.

3.12.2 Field Documentation

3.12.2.1 int pf_plot_params::annot

Flag indicating whether or not to annotate plot with peak numbers.

3.12.2.2 int pf_plot_params::cal

Flag indicating whether or not to plot in calibrated units.

3.12.2.3 char* pf_plot_params::dev

Plotting device name.

3.12.2.4 char* pf_plot_params::fn

Output file name.

3.12.2.5 int pf_plot_params::marked

Flag indicating whether or not to mark non-background regions.

3.12.2.6 int pf_plot_params::norm

Flag indicating whether or not to plot normalized values.

3.12.2.7 int pf_plot_params::peak

Peak to plot.

3.12.2.8 int pf_plot_params::rescale

Flag indicating whether or not to rescale plot.

3.12.2.9 double pf_plot_params::rot

Number of degrees to rotate the plot.

3.12.2.10 double pf_plot_params::xend

Ending domain value.

3.12.2.11 int pf_plot_params::xendcal

Flag indicating if xend is in calibrated units.

3.12.2.12 double [pf_plot_params::xstart](#)

Starting domain value.

3.12.2.13 int [pf_plot_params::xstartcal](#)

Flag indicating if xstart is in calibrated units.

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

- [common.h](#)

3.13 `pf_plot_range` Struct Reference

```
#include <commands.h>
```

3.13.1 Detailed Description

A plot-range specification.

Data Fields

- double `xstart`
Starting domain value.
- double `xend`
Ending domain value.
- int `xstartcal`
Flag indicating if `xstart` is in calibrated units.
- int `xendcal`
Flag indicating if `xend` is in calibrated units.

3.13.2 Field Documentation

3.13.2.1 double `pf_plot_range::xend`

Ending domain value.

3.13.2.2 int `pf_plot_range::xendcal`

Flag indicating if `xend` is in calibrated units.

3.13.2.3 double `pf_plot_range::xstart`

Starting domain value.

3.13.2.4 int `pf_plot_range::xstartcal`

Flag indicating if `xstart` is in calibrated units.

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

- `commands.h`

3.14 yy_buffer_state Struct Reference

Data Fields

- FILE * [yy_input_file](#)
- char * [yy_ch_buf](#)
- char * [yy_buf_pos](#)
- [yy_size_t](#) [yy_buf_size](#)
- int [yy_n_chars](#)
- int [yy_is_our_buffer](#)
- int [yy_is_interactive](#)
- int [yy_at_bol](#)
- int [yy_fill_buffer](#)
- int [yy_buffer_status](#)

3.14.1 Field Documentation

3.14.1.1 int [yy_buffer_state::yy_at_bol](#)

3.14.1.2 char* [yy_buffer_state::yy_buf_pos](#)

3.14.1.3 [yy_size_t](#) [yy_buffer_state::yy_buf_size](#)

3.14.1.4 int [yy_buffer_state::yy_buffer_status](#)

3.14.1.5 char* [yy_buffer_state::yy_ch_buf](#)

3.14.1.6 int [yy_buffer_state::yy_fill_buffer](#)

3.14.1.7 FILE* [yy_buffer_state::yy_input_file](#)

3.14.1.8 int [yy_buffer_state::yy_is_interactive](#)

3.14.1.9 int [yy_buffer_state::yy_is_our_buffer](#)

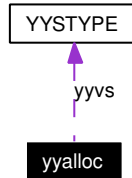
3.14.1.10 int [yy_buffer_state::yy_n_chars](#)

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

- [cmdlex.c](#)

3.15 yyalloC Union Reference

Collaboration diagram for yyalloC:



Data Fields

- short int `yyss`
- `YYSTYPE yyvs`

3.15.1 Field Documentation

3.15.1.1 short int `yyalloC::yyss`

3.15.1.2 `YYSTYPE yyalloC::yyvs`

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

- `cmdparse.c`

3.16 YYSTYPE Union Reference

```
#include <cmdparse.h>
```

Data Fields

- double [num](#)
- char * [str](#)
- char * [str](#)

3.16.1 Field Documentation

3.16.1.1 double [YYSTYPE::num](#)

3.16.1.2 char* [YYSTYPE::str](#)

3.16.1.3 char* [YYSTYPE::str](#)

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

- [cmdparse.c](#)
- [cmdparse.h](#)

Chapter 4

peakfinder-0.1 File Documentation

4.1 calibrate.c File Reference

4.1.1 Detailed Description

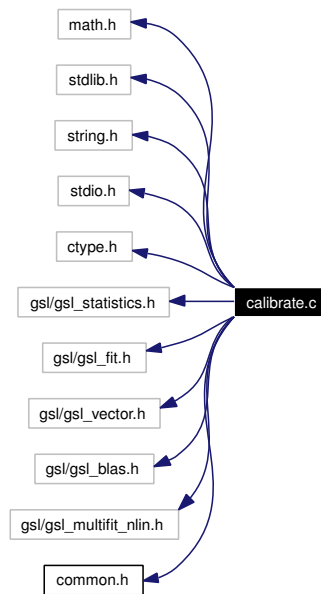
Author:

Hal Finkel

Calibration and fitting functions

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <gsl/gsl_statistics.h>
#include <gsl/gsl_fit.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multifit_nlin.h>
#include "common.h"
```

Include dependency graph for calibrate.c:



Data Structures

- struct [cal_fit_info](#)
Data to which the calibration curve is fit.

Defines

- #define [MAXITER](#) 1000
The maximum number of fitting iterations.
- #define [MAXFLEX](#) 6
The maximum number of channels by which a peak location can vary from the fit.

Functions

- int [pf_calibrate](#) (struct [pf_data](#) *d, struct [pf_cal_fit](#) *f, char *fn)
Calibrate input using known data points.
- double [pf_calfunc](#) (struct [pf_cal_fit](#) *f, double ch)
Evaluate the calibration fit function for a given peak.

4.1.2 Define Documentation

4.1.2.1 #define MAXFLEX 6

The maximum number of channels by which a peak location can vary from the fit.

4.1.2.2 #define MAXITER 1000

The maximum number of fitting iterations.

4.1.3 Function Documentation

4.1.3.1 double pf_calfunc (struct pf_cal_fit **f*, double *ch*)

Evaluate the calibration fit function for a given peak.

Parameters:

← *f* The calibration fit

← *ch* The value at which to evaluate the function

Returns:

The value of the function

4.1.3.2 int pf_calibrate (struct pf_data **d*, struct pf_cal_fit **f*, char **fn*)

Calibrate input using known data points.

Parameters:

← *d* The data set

→ *f* The calibration fit

← *fn* The file containing the calibration points

Returns:

Boolean success flag

4.2 cmdlex.c File Reference

4.2.1 Detailed Description

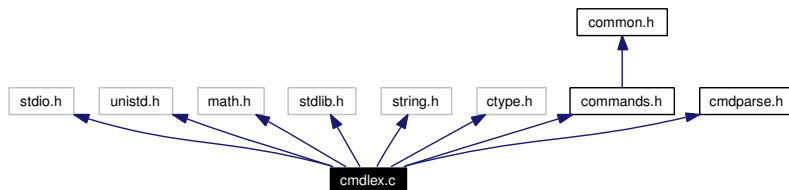
Author:

Hal Finkel

The command lexer

```
#include <stdio.h>
#include <unistd.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "commands.h"
#include "cmdparse.h"
```

Include dependency graph for cmdlex.c:



Data Structures

- struct [yy_buffer_state](#)

Defines

- #define [FLEX_SCANNER](#)
- #define [YY_FLEX_MAJOR_VERSION](#) 2
- #define [YY_FLEX_MINOR_VERSION](#) 5
- #define [yyconst](#)
- #define [YY_PROTO\(proto\)](#) ()
- #define [YY_NULL](#) 0
- #define [YY_SC_TO_UI\(c\)](#) ((unsigned int) (unsigned char) c)
- #define [BEGIN](#) yy_start = 1 + 2 *
- #define [YY_START](#) ((yy_start - 1) / 2)
- #define [YYSTATE](#) YY_START
- #define [YY_STATE_EOF\(state\)](#) (YY_END_OF_BUFFER + state + 1)
- #define [YY_NEW_FILE](#) yyrestart([yyin](#))
- #define [YY_END_OF_BUFFER_CHAR](#) 0
- #define [YY_BUF_SIZE](#) 16384
- #define [BOB_ACT_CONTINUE_SCAN](#) 0

- #define `EOB_ACT_END_OF_FILE` 1
- #define `EOB_ACT_LAST_MATCH` 2
- #define `yless(n)`
- #define `unput(c)` `yyunput(c, yytext_ptr)`
- #define `YY_BUFFER_NEW` 0
- #define `YY_BUFFER_NORMAL` 1
- #define `YY_BUFFER_EOF_PENDING` 2
- #define `YY_CURRENT_BUFFER` `yy_current_buffer`
- #define `YY_FLUSH_BUFFER` `yy_flush_buffer(yy_current_buffer)`
- #define `yy_new_buffer` `yy_create_buffer`
- #define `yy_set_interactive(is_interactive)`
- #define `yy_set_bol(at_bol)`
- #define `YY_AT_BOL()` (`yy_current_buffer` → `yy_at_bol`)
- #define `yytext_ptr` `yytext`
- #define `YY_DO_BEFORE_ACTION`
- #define `YY_NUM_RULES` 34
- #define `YY_END_OF_BUFFER` 35
- #define `REJECT` `reject_used_but_not_detected`
- #define `yymore()` `yymore_used_but_not_detected`
- #define `YY_MORE_ADJ` 0
- #define `YY_RESTORE_YY_MORE_OFFSET`
- #define `INITIAL` 0
- #define `YY_INPUT(buf, result, max_size)`
- #define `YY_SKIP_YYWRAP`
- #define `YY_NO_UNPUT`
- #define `YY_NO_PUSH_STATE` 1
- #define `YY_NO_POP_STATE` 1
- #define `YY_NO_TOP_STATE` 1
- #define `YY_READ_BUF_SIZE` 8192
- #define `ECHO` (void) `fwrite(yytext, yyleng, 1, yyout)`
- #define `yyterminate()` return `YY_NULL`
- #define `YY_START_STACK_INCR` 25
- #define `YY_FATAL_ERROR(msg)` `yy_fatal_error(msg)`
- #define `YY_DECL` int `yylex YY_PROTO((void))`
- #define `YY_BREAK` `break;`
- #define `YY_RULE_SETUP` `YY_USER_ACTION`
- #define `YY_EXIT_FAILURE` 2
- #define `yless(n)`

Typedefs

- typedef `yy_buffer_state` * `YY_BUFFER_STATE`
- typedef unsigned int `yy_size_t`
- typedef unsigned char `YY_CHAR`
- typedef int `yy_state_type`

Functions

- void `yyrestart YY_PROTO` ((FILE *input_file))
- void `yy_switch_to_buffer YY_PROTO` ((YY_BUFFER_STATE new_buffer))
- void `yy_load_buffer_state YY_PROTO` ((void))
- `YY_BUFFER_STATE yy_create_buffer YY_PROTO` ((FILE *file, int size))
- void `yy_delete_buffer YY_PROTO` ((YY_BUFFER_STATE b))
- void `yy_init_buffer YY_PROTO` ((YY_BUFFER_STATE b, FILE *file))
- `YY_BUFFER_STATE yy_scan_buffer YY_PROTO` ((char *base, `yy_size_t` size))
- `YY_BUFFER_STATE yy_scan_string YY_PROTO` ((yyconst char *yy_str))
- `YY_BUFFER_STATE yy_scan_bytes YY_PROTO` ((yyconst char *bytes, int len))
- int `yywrap` ()

Dummy yywrap function.

Variables

- int `yylen`
- FILE * `yyin` = (FILE *) 0 *`yyout` = (FILE *) 0
- FILE * `yyout`
- char * `yytext`
- char * `parse_buffer`

The current buffer to parse.

- int `parse_pos`

The current position in the parse buffer.

- int `size`
- FILE * `file`
- int `len`

4.2.2 Define Documentation

- 4.2.2.1 `#define BEGIN yy_start = 1 + 2 *`
- 4.2.2.2 `#define ECHO (void) fwrite(yytext, yyleng, 1, yyout)`
- 4.2.2.3 `#define EOB_ACT_CONTINUE_SCAN 0`
- 4.2.2.4 `#define EOB_ACT_END_OF_FILE 1`
- 4.2.2.5 `#define EOB_ACT_LAST_MATCH 2`
- 4.2.2.6 `#define FLEX_SCANNER`
- 4.2.2.7 `#define INITIAL 0`
- 4.2.2.8 `#define REJECT reject_used_but_not_detected`
- 4.2.2.9 `#define unput(c) yyunput(c, yytext_ptr)`
- 4.2.2.10 `#define YY_AT_BOL() (yy_current_buffer → yy_at_bol)`
- 4.2.2.11 `#define YY_BREAK break;`
- 4.2.2.12 `#define YY_BUF_SIZE 16384`
- 4.2.2.13 `#define YY_BUFFER_EOF_PENDING 2`
- 4.2.2.14 `#define YY_BUFFER_NEW 0`
- 4.2.2.15 `#define YY_BUFFER_NORMAL 1`
- 4.2.2.16 `#define YY_CURRENT_BUFFER yy_current_buffer`
- 4.2.2.17 `#define YY_DECL int yylex YY_PROTO((void))`
- 4.2.2.18 `#define YY_DO_BEFORE_ACTION`

Value:

```
yytext_ptr = yy_bp; \  
  yyleng = (int) (yy_cp - yy_bp); \  
  yy_hold_char = *yy_cp; \  
  *yy_cp = '\0'; \  
  yy_c_buf_p = yy_cp;
```

4.2.2.19 **#define YY_END_OF_BUFFER 35**

4.2.2.20 **#define YY_END_OF_BUFFER_CHAR 0**

4.2.2.21 **#define YY_EXIT_FAILURE 2**

4.2.2.22 **#define YY_FATAL_ERROR(msg) yy_fatal_error(msg)**

4.2.2.23 **#define YY_FLEX_MAJOR_VERSION 2**

4.2.2.24 **#define YY_FLEX_MINOR_VERSION 5**

4.2.2.25 **#define YY_FLUSH_BUFFER yy_flush_buffer(yy_current_buffer)**

4.2.2.26 **#define YY_INPUT(buf, result, max_size)**

Value:

```
{ \
    char c = parse_buffer[parse_pos++]; \
    result = (c == '\0') ? YY_NULL : (buf[0] = c, 1); \
}
```

- 4.2.2.27 **#define YY_MORE_ADJ 0**
- 4.2.2.28 **#define yy_new_buffer yy_create_buffer**
- 4.2.2.29 **#define YY_NEW_FILE yyrestart([yyin](#))**
- 4.2.2.30 **#define YY_NO_POP_STATE 1**
- 4.2.2.31 **#define YY_NO_PUSH_STATE 1**
- 4.2.2.32 **#define YY_NO_TOP_STATE 1**
- 4.2.2.33 **#define YY_NO_UNPUT**
- 4.2.2.34 **#define YY_NULL 0**
- 4.2.2.35 **#define YY_NUM_RULES 34**
- 4.2.2.36 **#define YY_PROTO(proto) ()**
- 4.2.2.37 **#define YY_READ_BUF_SIZE 8192**
- 4.2.2.38 **#define YY_RESTORE_YY_MORE_OFFSET**
- 4.2.2.39 **#define YY_RULE_SETUP YY_USER_ACTION**
- 4.2.2.40 **#define YY_SC_TO_UI(c) ((unsigned int) (unsigned char) c)**
- 4.2.2.41 **#define yy_set_bol(at_bol)**

Value:

```
{ \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_at_bol = at_bol; \
}
```

- 4.2.2.42 **#define yy_set_interactive(is_interactive)**

Value:

```
{ \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_is_interactive = is_interactive; \
}
```

4.2.2.43 #define YY_SKIP_YYWRAP

4.2.2.44 #define YY_START ((yy_start - 1) / 2)

4.2.2.45 #define YY_START_STACK_INCR 25

4.2.2.46 #define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)

4.2.2.47 #define yyconst

4.2.2.48 #define yyless(n)

Value:

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        yytext[yy leng] = yy_hold_char; \
        yy_c_buf_p = yytext + n; \
        yy_hold_char = *yy_c_buf_p; \
        *yy_c_buf_p = '\0'; \
        yy leng = n; \
    } \
while ( 0 )
```

4.2.2.49 #define yyless(n)

Value:

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        *yy_cp = yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yy_c_buf_p = yy_cp = yy_bp + n - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
while ( 0 )
```

4.2.2.50 `#define yymore() yymore_used_but_not_detected`

4.2.2.51 `#define YYSTATE YY_START`

4.2.2.52 `#define yyterminate() return YY_NULL`

4.2.2.53 `#define yytext_ptr yytext`

4.2.3 Typedef Documentation

4.2.3.1 `typedef struct yy_buffer_state* YY_BUFFER_STATE`

4.2.3.2 `typedef unsigned char YY_CHAR`

4.2.3.3 `typedef unsigned int yy_size_t`

4.2.3.4 `typedef int yy_state_type`

4.2.4 Function Documentation

4.2.4.1 `YY_BUFFER_STATE yy_scan_bytes YY_PROTO ((yyconst char *bytes, int len))`

4.2.4.2 `YY_BUFFER_STATE yy_scan_string YY_PROTO ((yyconst char *yy_str))`

4.2.4.3 `YY_BUFFER_STATE yy_scan_buffer YY_PROTO ((char *base, yy_size_t size))`

4.2.4.4 `void yy_init_buffer YY_PROTO ((YY_BUFFER_STATE b, FILE *file))`

4.2.4.5 `void yy_flush_buffer YY_PROTO ((YY_BUFFER_STATE b))`

4.2.4.6 `YY_BUFFER_STATE yy_create_buffer YY_PROTO ((FILE *file, int size))`

4.2.4.7 `int input YY_PROTO ((void))`

4.2.4.8 `void yy_switch_to_buffer YY_PROTO ((YY_BUFFER_STATE new_buffer))`

4.2.4.9 `void yyrestart YY_PROTO ((FILE *input_file))`

4.2.4.10 `int yywrap ()`

Dummy yywrap function.

Returns:

Always a value of 1

4.2.5 Variable Documentation

4.2.5.1 FILE* **file**

4.2.5.2 int **len**

4.2.5.3 char* **parse_buffer**

The current buffer to parse.

4.2.5.4 int **parse_pos**

The current position in the parse buffer.

4.2.5.5 **yy_size_t** size

4.2.5.6 FILE * **yyin** = (FILE *) 0 ***yyout** = (FILE *) 0

4.2.5.7 int **yylen**

4.2.5.8 FILE * **yyout**

4.2.5.9 char * **yytext**

4.3 cmdparse.c File Reference

4.3.1 Detailed Description

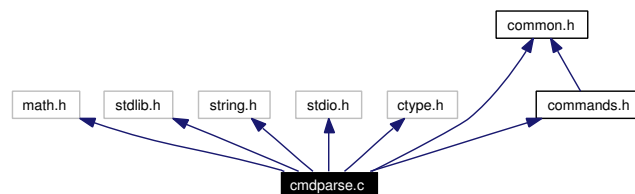
Author:

Hal Finkel

The command line parser

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include "commands.h"
#include "common.h"
```

Include dependency graph for cmdparse.c:



Data Structures

- union [YYSTYPE](#)
- union [yyalloc](#)

Defines

- #define [YBISON](#) 1
- #define [YYSKELETON_NAME](#) "yacc.c"
- #define [YYPURE](#) 0
- #define [YYLSP_NEEDED](#) 0
- #define [QUIT](#) 258
- #define [PLOT](#) 259
- #define [PRINT](#) 260
- #define [DATA](#) 261
- #define [NORM](#) 262
- #define [SMOOTH](#) 263
- #define [LOAD](#) 264
- #define [TO](#) 265
- #define [RESCALE](#) 266
- #define [MARKED](#) 267
- #define [PEAKS](#) 268

- #define [PEAK](#) 269
- #define [ANNOTATE](#) 270
- #define [DRIVER](#) 271
- #define [ROTATED](#) 272
- #define [SORTED](#) 273
- #define [BY](#) 274
- #define [NUMBER](#) 275
- #define [HEIGHT](#) 276
- #define [AREA](#) 277
- #define [CALIBRATE](#) 278
- #define [USING](#) 279
- #define [CLEAR](#) 280
- #define [CALIBRATION](#) 281
- #define [CALIBRATED](#) 282
- #define [CHANNEL](#) 283
- #define [STRING](#) 284
- #define [NUM](#) 285
- #define [YYDEBUG](#) 0
- #define [YYERROR_VERBOSE](#) 0
- #define [yystype](#) YYSTYPE
- #define [YYSTYPE_IS_DECLARED](#) 1
- #define [YYSTYPE_IS_TRIVIAL](#) 1
- #define [YYFREE](#) free
- #define [YYMALLOC](#) malloc
- #define [YYSTACK_ALLOC](#) YYMALLOC
- #define [YYSTACK_FREE](#) YYFREE
- #define [YYSTACK_GAP_MAXIMUM](#) (sizeof (union yyallo) - 1)
- #define [YYSTACK_BYTES\(N\)](#)
- #define [YYCOPY](#)(To, From, Count)
- #define [YYSTACK_RELOCATE](#)(Stack)
- #define [YYFINAL](#) 23
- #define [YYLAST](#) 55
- #define [YYNTOKENS](#) 31
- #define [YYNNTS](#) 27
- #define [YYNRULES](#) 55
- #define [YYNSTATES](#) 77
- #define [YYUNDEFTOK](#) 2
- #define [YYMAXUTOK](#) 285
- #define [YYTRANSLATE\(YYX\)](#) ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)
- #define [YYPACT_NINF](#) -25
- #define [YYTABLE_NINF](#) -1
- #define [YYSIZE_T](#) unsigned int
- #define [yyerrok](#) (yyerrstatus = 0)
- #define [yyclearin](#) (yychar = YYEMPTY)
- #define [YYEMPTY](#) (-2)
- #define [YYEOF](#) 0
- #define [YYACCEPT](#) goto yyacceptlab
- #define [YYABORT](#) goto yyabortlab
- #define [YYERROR](#) goto yyerrorlab

- #define `YYFAIL` goto `yyerrlab`
- #define `YYRECOVERING()` (!`yyerrstatus`)
- #define `YYBACKUP`(Token, Value)
- #define `YYTERROR` 1
- #define `YYERRCODE` 256
- #define `YYLLOC_DEFAULT`(Current, Rhs, N)
- #define `YYLEX` `yylex` ()
- #define `YYDPRINTF`(Args)
- #define `YYDSYMPRINT`(Args)
- #define `YYDSYMPRINTF`(Title, Token, Value, Location)
- #define `YY_STACK_PRINT`(Bottom, Top)
- #define `YY_REDUCE_PRINT`(Rule)
- #define `YYINITDEPTH` 200
- #define `YYMAXDEPTH` 10000
- #define `YYPOPSTACK` (`yyvsp-`, `yyssp-`)

Typedefs

- typedef `YYSTYPE` `YYSTYPE`
- typedef short int `yysigned_char`

Enumerations

- enum `yytokentype` {
 `QUIT` = 258, `PLOT` = 259, `PRINT` = 260, `DATA` = 261,
 `NORM` = 262, `SMOOTH` = 263, `LOAD` = 264, `TO` = 265,
 `RESCALE` = 266, `MARKED` = 267, `PEAKS` = 268, `PEAK` = 269,
 `ANNOTATE` = 270, `DRIVER` = 271, `ROTATED` = 272, `SORTED` = 273,
 `BY` = 274, `NUMBER` = 275, `HEIGHT` = 276, `AREA` = 277,
 `CALIBRATE` = 278, `USING` = 279, `CLEAR` = 280, `CALIBRATION` = 281,
 `CALIBRATED` = 282, `CHANNEL` = 283, `STRING` = 284, `NUM` = 285 }

Functions

- int `yylex` ()
 yylex function
- void `yyerror` (char const *s)
 The provided yyerror function.
- int `yyvsparse` ()
- void `pf_init_parse_data` ()
 Initialize the parse context data.
- void `pf_free_parse_data` ()
 Free any data associated with the last parse.

- void [pf_get_parsed_plot_params](#) (struct [pf_plot_params](#) *p)
Get the parsed plot parameters.
- int [pf_get_parsed_rescale](#) ()
Get the parsed rescale flag.
- int [pf_get_parsed_cal](#) ()
Get the parsed calibration flag.
- int [pf_get_parsed_norm](#) ()
Get the parsed normalization flag.
- int [pf_get_parsed_marked](#) ()
Get the parsed mark flag.
- int [pf_get_parsed_annot](#) ()
Get the parsed annotate flag.
- int [pf_get_parsed_peak](#) ()
Get the parsed peak number.
- double [pf_get_parsed_rotation](#) ()
The the parsed rotation amount.
- enum [pf_command](#) [pf_get_parsed_command](#) ()
Get the parsed command.
- enum [pf_peak_sort](#) [pf_get_parsed_sort](#) ()
Get the parsed sort flag.
- [pf_plot_range](#) * [pf_get_parsed_plot_range](#) ()
Get the parsed plot range.
- char * [pf_get_parsed_file](#) ()
Get the parsed file name.
- char * [pf_get_parsed_driver](#) ()
Get the parsed driver name.

Variables

- [pf_plot_range](#) [plot_range](#)
The range of the requested plot.
- char * [load_file](#)
The provided file name.
- char * [driver_name](#)

The provided plotting driver name.

- int `use_norm`

Flag indicating if the plot should use normalized data.

- int `need_rescale`

Flag indicating if the plot should be rescaled for the data displayed.

- int `need_cal`

Flag indicating if the plot should use a calibrated x-axis.

- int `should_mark`

Flag indicating if the plot should mark non-background areas.

- int `should_annot`

Flag indicating if the peaks should be numbered on the plot.

- int `peak_num`

The peak number of the peak requested.

- double `rot_angle`

The number of degrees to rotate the plot from the default angle.

- int `yychar`

- `YYSTYPE` `yylval`

- int `yynerrs`

4.3.2 Define Documentation

4.3.2.1 `#define ANNOTATE` 270

4.3.2.2 `#define AREA` 277

4.3.2.3 `#define BY` 274

4.3.2.4 `#define CALIBRATE` 278

4.3.2.5 `#define CALIBRATED` 282

4.3.2.6 `#define CALIBRATION` 281

4.3.2.7 `#define CHANNEL` 283

4.3.2.8 `#define CLEAR` 280

4.3.2.9 `#define DATA` 261

4.3.2.10 `#define DRIVER` 271

4.3.2.11 `#define HEIGHT` 276

4.3.2.12 `#define LOAD` 264

4.3.2.13 `#define MARKED` 267

4.3.2.14 `#define NORM` 262

4.3.2.15 `#define NUM` 285

4.3.2.16 `#define NUMBER` 275

4.3.2.17 `#define PEAK` 269

4.3.2.18 `#define PEAKS` 268

4.3.2.19 `#define PLOT` 259

4.3.2.20 `#define PRINT` 260

4.3.2.21 `#define QUIT` 258

4.3.2.22 `#define RESCALE` 266

4.3.2.23 `#define ROTATED` 272

4.3.2.24 `#define SMOOTH` 263

4.3.2.25 `#define SORTED` 273

4.3.2.26 `#define STRING` 284

4.3.2.27 `#define TQ` 265
Generated on Sun Feb 20 17:46:14 2005 for peakfinder-0.1 by Doxygen

4.3.2.28 `#define USING` 279

4.3.2.29 `#define YY_REDUCE_PRINT(Rule)`

4.3.2.30 `#define YY_STACK_PRINT(Bottom, Top)`

```

do
  if (yychar == YYEMPTY && yylen == 1)
    {
      yychar = (Token);
      yylval = (Value);
      yytoken = YYTRANSLATE (yychar);
      YYPOPSTACK;
      goto yybackup;
    }
  else
    {
      yyerror ("syntax error: cannot back up");\
      YYERROR;
    }
while (0)

```

4.3.2.34 #define YYBISON 1

4.3.2.35 #define yyclearin (**yychar** = YYEMPTY)

4.3.2.36 #define YYCOPY(**To**, **From**, **Count**)

Value:

```

do
  {
    register YYSIZE_T yyi;
    for (yyi = 0; yyi < (Count); yyi++)
      (To)[yyi] = (From)[yyi];
  }
while (0)

```


- 4.3.2.37 **#define YYDEBUG 0**
- 4.3.2.38 **#define YYDPRINTF(Args)**
- 4.3.2.39 **#define YYDSYMPRINT(Args)**
- 4.3.2.40 **#define YYDSYMPRINTF(Title, Token, Value, Location)**
- 4.3.2.41 **#define YYEMPTY (-2)**
- 4.3.2.42 **#define YYEOF 0**
- 4.3.2.43 **#define YYERRCODE 256**
- 4.3.2.44 **#define yyerrok (yyerrstatus = 0)**
- 4.3.2.45 **#define YYERROR goto yyerrorlab**
- 4.3.2.46 **#define YYERROR_VERBOSE 0**
- 4.3.2.47 **#define YYFAIL goto yyerrlab**
- 4.3.2.48 **#define YYFINAL 23**
- 4.3.2.49 **#define YYFREE free**
- 4.3.2.50 **#define YYINITDEPTH 200**
- 4.3.2.51 **#define YYLAST 55**
- 4.3.2.52 **#define YYLEX yylex ()**
- 4.3.2.53 **#define YYLLOC_DEFAULT(Current, Rhs, N)**

Value:

```
((Current).first_line   = (Rhs)[1].first_line,  \  
  (Current).first_column = (Rhs)[1].first_column,  \  
  (Current).last_line    = (Rhs)[N].last_line,    \  
  (Current).last_column  = (Rhs)[N].last_column)
```

- 4.3.2.54 **#define YYLSP_NEEDED 0**
- 4.3.2.55 **#define YYMALLOC malloc**
- 4.3.2.56 **#define YYMAXDEPTH 10000**
- 4.3.2.57 **#define YYMAXUTOK 285**
- 4.3.2.58 **#define YYNNTS 27**
- 4.3.2.59 **#define YYNRULES 55**
- 4.3.2.60 **#define YYNSTATES 77**
- 4.3.2.61 **#define YYNTOKENS 31**
- 4.3.2.62 **#define YYPACT_NINF -25**
- 4.3.2.63 **#define YYPOPSTACK (yyvsp-, yyssp-)**
- 4.3.2.64 **#define YYPURE 0**
- 4.3.2.65 **#define YYRECOVERING() (!yyerrstatus)**
- 4.3.2.66 **#define YYSIZE_T unsigned int**
- 4.3.2.67 **#define YYSKELETON_NAME "yacc.c"**
- 4.3.2.68 **#define YYSTACK_ALLOC YYMALLOC**
- 4.3.2.69 **#define YYSTACK_BYTES(N)**

Value:

```
((N) * (sizeof (short int) + sizeof (YYSTYPE)) \
+ YYSTACK_GAP_MAXIMUM)
```

- 4.3.2.70 **#define YYSTACK_FREE YFREE**
- 4.3.2.71 **#define YYSTACK_GAP_MAXIMUM (sizeof (union yyallo) - 1)**
- 4.3.2.72 **#define YYSTACK_RELOCATE(Stack)**

Value:

```
do
{
    YYSIZE_T yynewbytes;
    YCOPY (&yptr->Stack, Stack, yysize);
    Stack = &yptr->Stack;
    yynewbytes = yystacksize * sizeof (*Stack) + YYSTACK_GAP_MAXIMUM; \
    yyptr += yynewbytes / sizeof (*yyptr);
}
while (0)
```

4.3.2.73 `#define YYSTYPE` [YYSTYPE](#)

4.3.2.74 `#define YYSTYPE_IS_DECLARED` 1

4.3.2.75 `#define YYSTYPE_IS_TRIVIAL` 1

4.3.2.76 `#define YYTABLE_NINF` -1

4.3.2.77 `#define YYTERROR` 1

4.3.2.78 `#define YYTRANSLATE(YYX) ((unsigned int) (YYX) <= YYMAXUTOK ?
yytranslate[YYX] : YYUNDEFTOK)`

4.3.2.79 `#define YYUNDEFTOK` 2

4.3.3 Typedef Documentation

4.3.3.1 `typedef short int` [yysigned_char](#)

4.3.3.2 `typedef union` [YYSTYPE](#) [YYSTYPE](#)

4.3.4 Enumeration Type Documentation

4.3.4.1 `enum` [ytoken_type](#)

Enumeration values:

QUIT

PLOT

PRINT

DATA

NORM

SMOOTH

LOAD

TO

RESCALE

MARKED

PEAKS

PEAK

ANNOTATE

DRIVER

ROTATED

SORTED

BY

NUMBER

HEIGHT

AREA

CALIBRATE

USING

CLEAR

CALIBRATION

CALIBRATED

CHANNEL

STRING

NUM

4.3.5 Function Documentation

4.3.5.1 void pf_free_parse_data ()

Free any data associated with the last parse.

4.3.5.2 int pf_get_parsed_annot ()

Get the parsed annotate flag.

Returns:

The parsed annotate flag

4.3.5.3 int pf_get_parsed_cal ()

Get the parsed calibration flag.

Returns:

The parsed calibration flag

4.3.5.4 enum pf_command pf_get_parsed_command ()

Get the parsed command.

Returns:

The parsed command

4.3.5.5 char* pf_get_parsed_driver ()

Get the parsed driver name.

Returns:

The parsed driver name

4.3.5.6 char* pf_get_parsed_file ()

Get the parsed file name.

Returns:

The parsed file name

4.3.5.7 int pf_get_parsed_marked ()

Get the parsed mark flag.

Returns:

The parsed mark flag

4.3.5.8 int pf_get_parsed_norm ()

Get the parsed normalization flag.

Returns:

The parsed normalization flag

4.3.5.9 int pf_get_parsed_peak ()

Get the parsed peak number.

Returns:

The parsed peak number

4.3.5.10 void pf_get_parsed_plot_params (struct [pf_plot_params](#) * *p*)

Get the parsed plot parameters.

Parameters:

→ *p* The plot parameters

4.3.5.11 struct [pf_plot_range](#)* pf_get_parsed_plot_range ()

Get the parsed plot range.

Returns:

The parsed plot range

4.3.5.12 int pf_get_parsed_rescale ()

Get the parsed rescale flag.

Returns:

The parsed rescale flag

4.3.5.13 double pf_get_parsed_rotation ()

The the parsed rotation amount.

Returns:

The parsed rotation amount

4.3.5.14 enum pf_peak_sort pf_get_parsed_sort ()

Get the parsed sort flag.

Returns:

The parsed sort flag

4.3.5.15 void pf_init_parse_data ()

Initialize the parse context data.

4.3.5.16 void yyerror (char const * s)

The provided yyerror function.

Parameters:

← s The parser-provided error string

4.3.5.17 int yylex ()

yylex function

Returns:

The token type

4.3.5.18 int yyparse ()

Here is the call graph for this function:



4.3.6 Variable Documentation

4.3.6.1 char* `driver_name`

The provided plotting driver name.

4.3.6.2 char* `load_file`

The provided file name.

4.3.6.3 int `need_cal`

Flag indicating if the plot should use a calibrated x-axis.

4.3.6.4 int `need_rescale`

Flag indicating if the plot should be rescaled for the data displayed.

4.3.6.5 int `peak_num`

The peak number of the peak requested.

4.3.6.6 struct `pf_plot_range plot_range`

The range of the requested plot.

4.3.6.7 double `rot_angle`

The number of degrees to rotate the plot from the default angle.

4.3.6.8 int `should_annot`

Flag indicating if the peaks should be numbered on the plot.

4.3.6.9 int `should_mark`

Flag indicating if the plot should mark non-background areas.

4.3.6.10 int `use_norm`

Flag indicating if the plot should use normalized data.

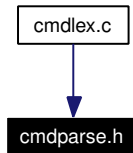
4.3.6.11 int `ychar`

4.3.6.12 `YYSTYPE` `yylval`

4.3.6.13 int `yynerrs`

4.4 cmdparse.h File Reference

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



Data Structures

- union [YYSTYPE](#)

Defines

- #define [QUIT](#) 258
- #define [PLOT](#) 259
- #define [PRINT](#) 260
- #define [DATA](#) 261
- #define [NORM](#) 262
- #define [SMOOTH](#) 263
- #define [LOAD](#) 264
- #define [TO](#) 265
- #define [RESCALE](#) 266
- #define [MARKED](#) 267
- #define [PEAKS](#) 268
- #define [PEAK](#) 269
- #define [ANNOTATE](#) 270
- #define [DRIVER](#) 271
- #define [ROTATED](#) 272
- #define [SORTED](#) 273
- #define [BY](#) 274
- #define [NUMBER](#) 275
- #define [HEIGHT](#) 276
- #define [AREA](#) 277
- #define [CALIBRATE](#) 278
- #define [USING](#) 279
- #define [CLEAR](#) 280
- #define [CALIBRATION](#) 281
- #define [CALIBRATED](#) 282
- #define [CHANNEL](#) 283
- #define [STRING](#) 284
- #define [NUM](#) 285
- #define [yystype YYSTYPE](#)
- #define [YYSTYPE_IS_DECLARED](#) 1
- #define [YYSTYPE_IS_TRIVIAL](#) 1

Typedefs

- typedef [YYSTYPE](#) YYSTYPE

Enumerations

- enum [yytokentype](#) {

[QUIT](#) = 258, [PLOT](#) = 259, [PRINT](#) = 260, [DATA](#) = 261,

[NORM](#) = 262, [SMOOTH](#) = 263, [LOAD](#) = 264, [TO](#) = 265,

[RESCALE](#) = 266, [MARKED](#) = 267, [PEAKS](#) = 268, [PEAK](#) = 269,

[ANNOTATE](#) = 270, [DRIVER](#) = 271, [ROTATED](#) = 272, [SORTED](#) = 273,

[BY](#) = 274, [NUMBER](#) = 275, [HEIGHT](#) = 276, [AREA](#) = 277,

[CALIBRATE](#) = 278, [USING](#) = 279, [CLEAR](#) = 280, [CALIBRATION](#) = 281,

[CALIBRATED](#) = 282, [CHANNEL](#) = 283, [STRING](#) = 284, [NUM](#) = 285 }

Variables

- YYSTYPE [yylval](#)

4.4.1 Define Documentation

4.4.1.1 `#define ANNOTATE` 270

4.4.1.2 `#define AREA` 277

4.4.1.3 `#define BY` 274

4.4.1.4 `#define CALIBRATE` 278

4.4.1.5 `#define CALIBRATED` 282

4.4.1.6 `#define CALIBRATION` 281

4.4.1.7 `#define CHANNEL` 283

4.4.1.8 `#define CLEAR` 280

4.4.1.9 `#define DATA` 261

4.4.1.10 `#define DRIVER` 271

4.4.1.11 `#define HEIGHT` 276

4.4.1.12 `#define LOAD` 264

4.4.1.13 `#define MARKED` 267

4.4.1.14 `#define NORM` 262

4.4.1.15 `#define NUM` 285

4.4.1.16 `#define NUMBER` 275

4.4.1.17 `#define PEAK` 269

4.4.1.18 `#define PEAKS` 268

4.4.1.19 `#define PLOT` 259

4.4.1.20 `#define PRINT` 260

4.4.1.21 `#define QUIT` 258

4.4.1.22 `#define RESCALE` 266

4.4.1.23 `#define ROTATED` 272

4.4.1.24 `#define SMOOTH` 263

4.4.1.25 `#define SORTED` 273

4.4.1.26 `#define STRING` 284

4.4.1.27 `#define TO` 265

4.4.1.28 `#define USING` 279

4.4.1.29 `#define yystype` **YYSTYPE**

4.4.1.30 `#define YYSTYPE IS DECLARED` 1

PLOT
PRINT
DATA
NORM
SMOOTH
LOAD
TO
RESCALE
MARKED
PEAKS
PEAK
ANNOTATE
DRIVER
ROTATED
SORTED
BY
NUMBER
HEIGHT
AREA
CALIBRATE
USING
CLEAR
CALIBRATION
CALIBRATED
CHANNEL
STRING
NUM

4.4.4 Variable Documentation

4.4.4.1 [YYSTYPE](#) yylval

4.5 commands.h File Reference

4.5.1 Detailed Description

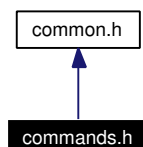
Author:

Hal Finkel

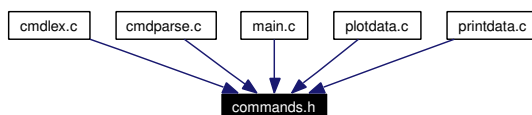
Command lexing and parsing

```
#include "common.h"
```

Include dependency graph for commands.h:



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



Data Structures

- struct [pf_plot_range](#)
A plot-range specification.

Enumerations

- enum [pf_command](#) {
[command_invalid](#), [command_quit](#), [command_load](#), [command_print_data](#),
[command_print_peaks](#), [command_print_peak](#), [command_print_calibration](#), [command_plot_data](#),
[command_plot_peak](#), [command_calibrate](#), [command_clear_calibration](#) }
A command type.

Functions

- enum [pf_command](#) [pf_get_parsed_command](#) ()
Get the parsed command.
- enum [pf_peak_sort](#) [pf_get_parsed_sort](#) ()
Get the parsed sort flag.

- `pf_plot_range * pf_get_parsed_plot_range ()`
Get the parsed plot range.
- `char * pf_get_parsed_file ()`
Get the parsed file name.
- `char * pf_get_parsed_driver ()`
Get the parsed driver name.
- `int pf_get_parsed_rescale ()`
Get the parsed rescale flag.
- `int pf_get_parsed_cal ()`
Get the parsed calibration flag.
- `int pf_get_parsed_norm ()`
Get the parsed normalization flag.
- `int pf_get_parsed_marked ()`
Get the parsed mark flag.
- `int pf_get_parsed_annot ()`
Get the parsed annotate flag.
- `int pf_get_parsed_peak ()`
Get the parsed peak number.
- `double pf_get_parsed_rotation ()`
The the parsed rotation amount.
- `void pf_get_parsed_plot_params (struct pf_plot_params *p)`
Get the parsed plot parameters.
- `void pf_init_parse_data ()`
Initialize the parse context data.
- `void pf_free_parse_data ()`
Free any data associated with the last parse.
- `void pf_execute_command ()`
Execute the next command.
- `void pf_set_command_parse_buffer (char *cmd)`
Set the buffer to parse.

4.5.2 Enumeration Type Documentation

4.5.2.1 enum `pf_command`

A command type.

Enumeration values:

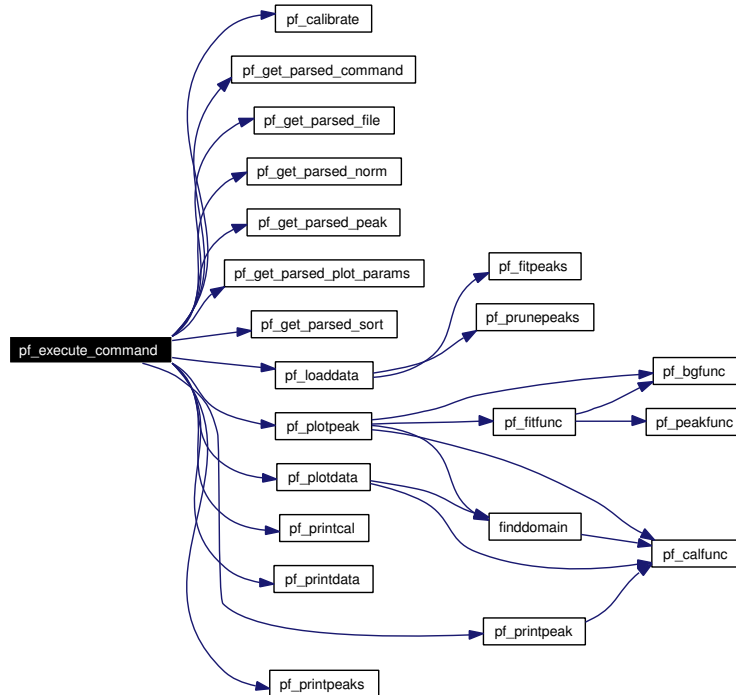
- command_invalid* Invalid command.
- command_quit* Quit command.
- command_load* Load command.
- command_print_data* Print-data command.
- command_print_peaks* Print-peaks command.
- command_print_peak* Print-peak command.
- command_print_calibration* Print-calibration command.
- command_plot_data* Plot-data command.
- command_plot_peak* Plot-peak command.
- command_calibrate* Calibrate command.
- command_clear_calibration* Clear-calibration command.

4.5.3 Function Documentation

4.5.3.1 void `pf_execute_command ()`

Execute the next command.

Here is the call graph for this function:



4.5.3.2 void pf_free_parse_data ()

Free any data associated with the last parse.

4.5.3.3 int pf_get_parsed_annot ()

Get the parsed annotate flag.

Returns:

The parsed annotate flag

4.5.3.4 int pf_get_parsed_cal ()

Get the parsed calibration flag.

Returns:

The parsed calibration flag

4.5.3.5 enum pf_command pf_get_parsed_command ()

Get the parsed command.

Returns:

The parsed command

4.5.3.6 char* pf_get_parsed_driver ()

Get the parsed driver name.

Returns:

The parsed driver name

4.5.3.7 char* pf_get_parsed_file ()

Get the parsed file name.

Returns:

The parsed file name

4.5.3.8 int pf_get_parsed_marked ()

Get the parsed mark flag.

Returns:

The parsed mark flag

4.5.3.9 int pf_get_parsed_norm ()

Get the parsed normalization flag.

Returns:

The parsed normalization flag

4.5.3.10 int pf_get_parsed_peak ()

Get the parsed peak number.

Returns:

The parsed peak number

4.5.3.11 void pf_get_parsed_plot_params (struct [pf_plot_params](#) * *p*)

Get the parsed plot parameters.

Parameters:

→ *p* The plot parameters

4.5.3.12 struct [pf_plot_range](#)* pf_get_parsed_plot_range ()

Get the parsed plot range.

Returns:

The parsed plot range

4.5.3.13 int pf_get_parsed_rescale ()

Get the parsed rescale flag.

Returns:

The parsed rescale flag

4.5.3.14 double pf_get_parsed_rotation ()

The the parsed rotation amount.

Returns:

The parsed rotation amount

4.5.3.15 enum `pf_peak_sort` `pf_get_parsed_sort ()`

Get the parsed sort flag.

Returns:

The parsed sort flag

4.5.3.16 void `pf_init_parse_data ()`

Initialize the parse context data.

4.5.3.17 void `pf_set_command_parse_buffer (char * cmd)`

Set the buffer to parse.

Parameters:

← *cmd* The buffer with the command string

4.6 common.h File Reference

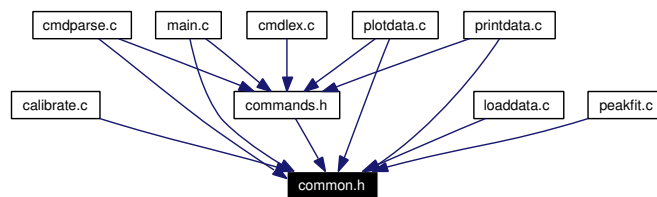
4.6.1 Detailed Description

Author:

Hal Finkel

Main peakfinder program

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



Data Structures

- struct [pf_peak_bg](#)
Peak background fit.
- struct [pf_peak_fit](#)
Peak fit.
- struct [pf_peak](#)
Peak data.
- struct [pf_cal_fit](#)
Calibration fit.
- struct [pf_cal_pt](#)
Calibration point.
- struct [pf_cal_pts](#)
Calibration points.
- struct [pf_data](#)
Data set.
- struct [pf_plot_params](#)
Plot parameters.

Defines

- #define [NCHAN](#) 4096
The maximum number of channels in the data set.

- #define `MAXCALPTS` 1024
The maximum number of calibration points.
- #define `MAXUNIT` 256
The maximum length of a calibration unit string.

Enumerations

- enum `pf_peak_sort` { `sort_number`, `sort_height`, `sort_area` }
Peak sort type.

Functions

- int `pf_parse_command` (char *command)
Parse a command string.
- int `pf_loaddata` (char *fn, struct `pf_data` *d)
Load data from a file.
- double `pf_chanwindowavg` (double *data, int ds, int ci, int ac)
Compute the average counts per bin over some window.
- void `pf_plotdata` (struct `pf_data` *d, struct `pf_plot_params` *p, struct `pf_cal_fit` *cal)
Plot data.
- void `pf_plotpeak` (struct `pf_data` *d, struct `pf_plot_params` *p, struct `pf_cal_fit` *cal)
Plot a given peak.
- int `pf_printdata` (struct `pf_data` *d, char *fn, int norm)
Print the data set.
- int `pf_printpeaks` (struct `pf_data` *d, char *fn, enum `pf_peak_sort` srt)
Print the peaks.
- int `pf_printpeak` (struct `pf_data` *d, struct `pf_cal_fit` *cal, int peak, char *fn)
Print data about a given peak.
- int `pf_printcal` (struct `pf_data` *d, struct `pf_cal_fit` *f, char *fn)
Print data about the calibration fit.
- void `pf_fitpeaks` (struct `pf_data` *d)
Fit peaks in the data set.
- void `pf_prunepeaks` (struct `pf_data` *d)
Prune peaks without reasonable fits.

- double `pf_bgfunc` (struct `pf_data` *d, int peak, double x)
Evaluate the background function for a given peak.
- double `pf_peakfunc` (struct `pf_data` *d, int peak, double x)
Evaluate the peak fit function for a given peak.
- double `pf_fitfunc` (struct `pf_data` *d, int peak, double x)
Evaluate the total fit function for a given peak.
- int `pf_calibrate` (struct `pf_data` *d, struct `pf_cal_fit` *f, char *fn)
Calibrate input using known data points.
- double `pf_calfunc` (struct `pf_cal_fit` *f, double ch)
Evaluate the calibration fit function for a given peak.

4.6.2 Define Documentation

4.6.2.1 #define MAXCALPTS 1024

The maximum number of calibration points.

4.6.2.2 #define MAXUNIT 256

The maximum length of a calibration unit string.

4.6.2.3 #define NCHAN 4096

The maximum number of channels in the data set.

4.6.3 Enumeration Type Documentation

4.6.3.1 enum pf_peak_sort

Peak sort type.

Enumeration values:

sort_number Sort by peak number.

sort_height Sort by peak height.

sort_area Sort by peak area.

4.6.4 Function Documentation

4.6.4.1 double pf_bgfunc (struct pf_data * d, int peak, double x)

Evaluate the background function for a given peak.

Parameters:

- ← *d* The data set
- ← *peak* The peak for which to evaluate the background function
- ← *x* The value at which to evaluate the function

Returns:

The value of the function

4.6.4.2 double pf_calfunc (struct pf_cal_fit *f, double ch)

Evaluate the calibration fit function for a given peak.

Parameters:

- ← *f* The calibration fit
- ← *ch* The value at which to evaluate the function

Returns:

The value of the function

4.6.4.3 int pf_calibrate (struct pf_data *d, struct pf_cal_fit *f, char *fn)

Calibrate input using known data points.

Parameters:

- ← *d* The data set
- *f* The calibration fit
- ← *fn* The file containing the calibration points

Returns:

Boolean success flag

4.6.4.4 double pf_chanwindowavg (double *data, int ds, int ci, int ac)

Compute the average counts per bin over some window.

Parameters:

- ← *data* The data array
- ← *ds* The size of the data array
- ← *ci* The center of the window
- ← *ac* The size of the window

Returns:

The average over the window

4.6.4.5 double pf_fitfunc (struct pf_data * *d*, int *peak*, double *x*)

Evaluate the total fit function for a given peak.

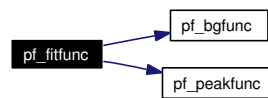
Parameters:

- ← *d* The data set
- ← *peak* The peak for which to evaluate the total fit function
- ← *x* The value at which to evaluate the function

Returns:

The value of the function

Here is the call graph for this function:



4.6.4.6 void pf_fitpeaks (struct pf_data * *d*)

Fit peaks in the data set.

Parameters:

- ↔ *d* The data set

4.6.4.7 int pf_loaddata (char * *fn*, struct pf_data * *d*)

Load data from a file.

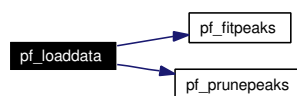
Parameters:

- ← *fn* The file name
- *d* The data set

Returns:

Boolean status flag

Here is the call graph for this function:



4.6.4.8 int pf_parse_command (char * *command*)

Parse a command string.

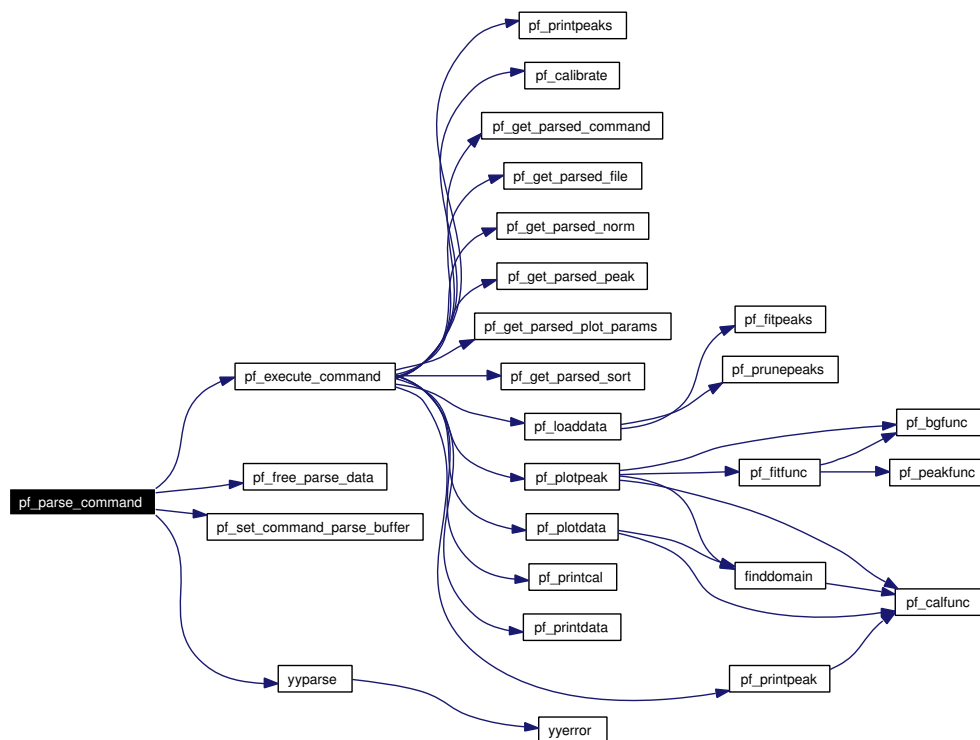
Parameters:

← *command* The command to parse and execute

Returns:

Status code

Here is the call graph for this function:



4.6.4.9 double pf_peakfunc (struct pf_data * *d*, int *peak*, double *x*)

Evaluate the peak fit function for a given peak.

Parameters:

← *d* The data set

← *peak* The peak for which to evaluate the peak fit function

← *x* The value at which to evaluate the function

Returns:

The value of the function

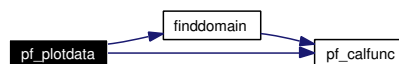
4.6.4.10 void pf_plotdata (struct pf_data * *d*, struct pf_plot_params * *p*, struct pf_cal_fit * *cal*)

Plot data.

Parameters:

- ← *d* The data set
- ← *p* The plotting parameters
- ← *cal* The calibration fit

Here is the call graph for this function:



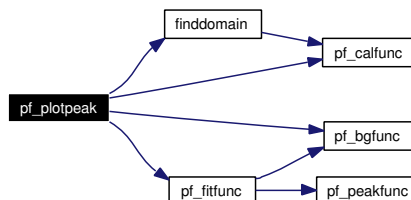
4.6.4.11 void pf_plotpeak (struct pf_data * *d*, struct pf_plot_params * *p*, struct pf_cal_fit * *cal*)

Plot a given peak.

Parameters:

- ← *d* The data set
- ← *p* The plotting parameters
- ← *cal* The calibration fit

Here is the call graph for this function:



4.6.4.12 int pf_printcal (struct pf_data * *d*, struct pf_cal_fit * *f*, char * *fn*)

Print data about the calibration fit.

Parameters:

- ← *d* The data set
- ← *f* The calibration fit
- ← *fn* The output file name

Returns:

Boolean status flag

4.6.4.13 `int pf_printdata (struct pf_data * d, char * fn, int norm)`

Print the data set.

Parameters:

- ← *d* The data set
- ← *fn* The output file name
- ← *norm* Flag indicating whether to print normalized data

Returns:

Boolean status flag

4.6.4.14 `int pf_printpeak (struct pf_data * d, struct pf_cal_fit * cal, int peak, char * fn)`

Print data about a given peak.

Parameters:

- ← *d* The data set
- ← *cal* The calibration fit
- ← *peak* The peak about which to print data
- ← *fn* The output file name

Returns:

Boolean status flag

Here is the call graph for this function:

**4.6.4.15** `int pf_printpeaks (struct pf_data * d, char * fn, enum pf_peak_sort srt)`

Print the peaks.

Parameters:

- ← *d* The data set
- ← *fn* The output file name
- ← *srt* Peak sort type

Returns:

Boolean status flag

4.6.4.16 `void pf_prunepeaks (struct pf_data * d)`

Prune peaks without reasonable fits.

Parameters:

- ↔ *d* The data set

4.7 loaddata.c File Reference

4.7.1 Detailed Description

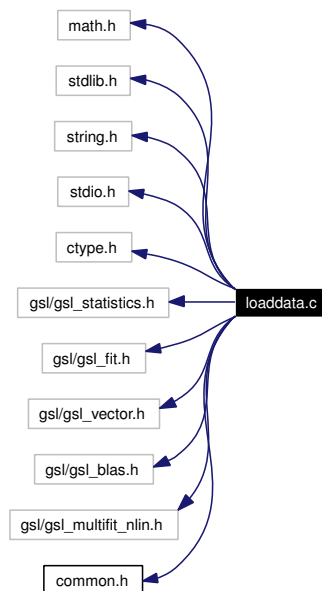
Author:

Hal Finkel

Data loading, normalization and background identification

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <gsl/gsl_statistics.h>
#include <gsl/gsl_fit.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multifit_nlin.h>
#include "common.h"
```

Include dependency graph for loaddata.c:



Defines

- #define `MINWIDTH` 4
Minimum width of a peak.

Functions

- int `pf_loaddata` (char *fn, struct `pf_data` *d)
Load data from a file.
- double `pf_chanwindowavg` (double *data, int ds, int ci, int ac)
Compute the average counts per bin over some window.

4.7.2 Define Documentation

4.7.2.1 #define MINWIDTH 4

Minimum width of a peak.

This number must be greater than or equal to the number of parameters in the peak fit

4.7.3 Function Documentation

4.7.3.1 double `pf_chanwindowavg` (double * data, int ds, int ci, int ac)

Compute the average counts per bin over some window.

Parameters:

- ← *data* The data array
- ← *ds* The size of the data array
- ← *ci* The center of the window
- ← *ac* The size of the window

Returns:

The average over the window

4.7.3.2 int `pf_loaddata` (char *fn, struct `pf_data` * d)

Load data from a file.

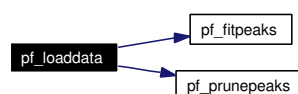
Parameters:

- ← *fn* The file name
- *d* The data set

Returns:

Boolean status flag

Here is the call graph for this function:



4.8 main.c File Reference

4.8.1 Detailed Description

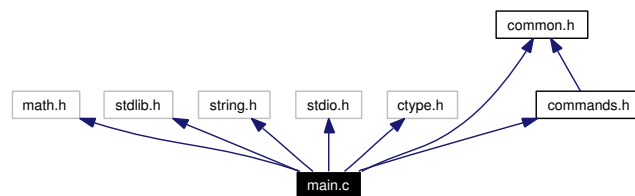
Author:

Hal Finkel

The main program, command parsing and execution

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include "commands.h"
#include "common.h"
```

Include dependency graph for main.c:



Defines

- #define `HISTORY_FILE` ".peakfind_history"
The name of the command-history file.
- #define `MAXCMDLINE` 1024
If readline is not available, the maximum number of characters per line.

Functions

- int `yyvsparse` ()
The generated parsing function.
- int `main` (int argc, char *argv[])
The main function.
- int `pf_parse_command` (char *command)
Parse a command string.
- void `pf_execute_command` ()
Execute the next command.

Variables

- `int should_quit`
Should the main loop terminate.
- `pf_cal_fit current_cal`
The current calibration.
- `pf_data current_data`
The current data set.
- `int has_data`
Valid data-set flag.

4.8.2 Define Documentation

4.8.2.1 `#define HISTORY_FILE ".peakfind_history"`

The name of the command-history file.

4.8.2.2 `#define MAXCMDLINE 1024`

If readline is not available, the maximum number of characters per line.

4.8.3 Function Documentation

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

The main function.

Parameters:

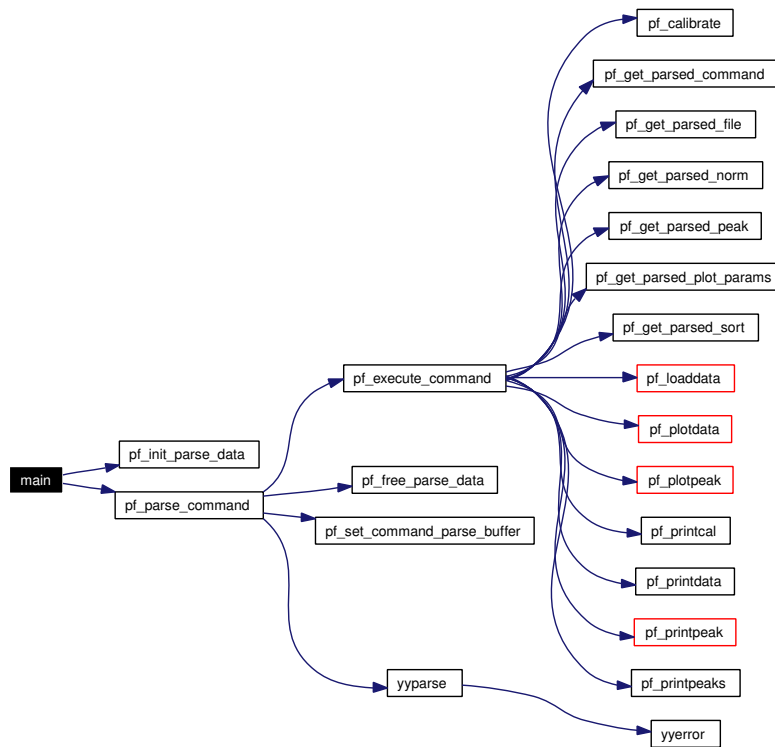
← *argc* The number of command line arguments

← *argv* The command line arguments

Returns:

The program exit status

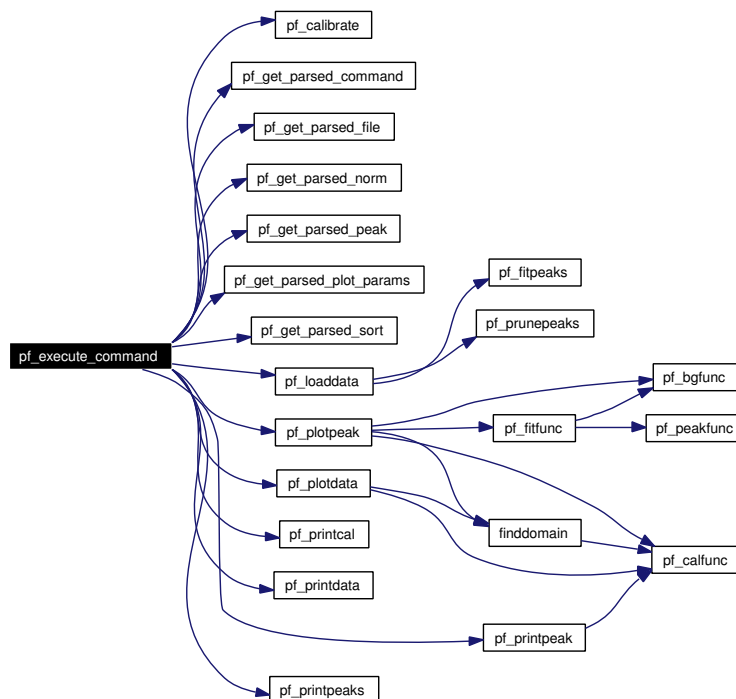
Here is the call graph for this function:



4.8.3.2 void pf_execute_command ()

Execute the next command.

Here is the call graph for this function:



4.8.3.3 int pf_parse_command (char * *command*)

Parse a command string.

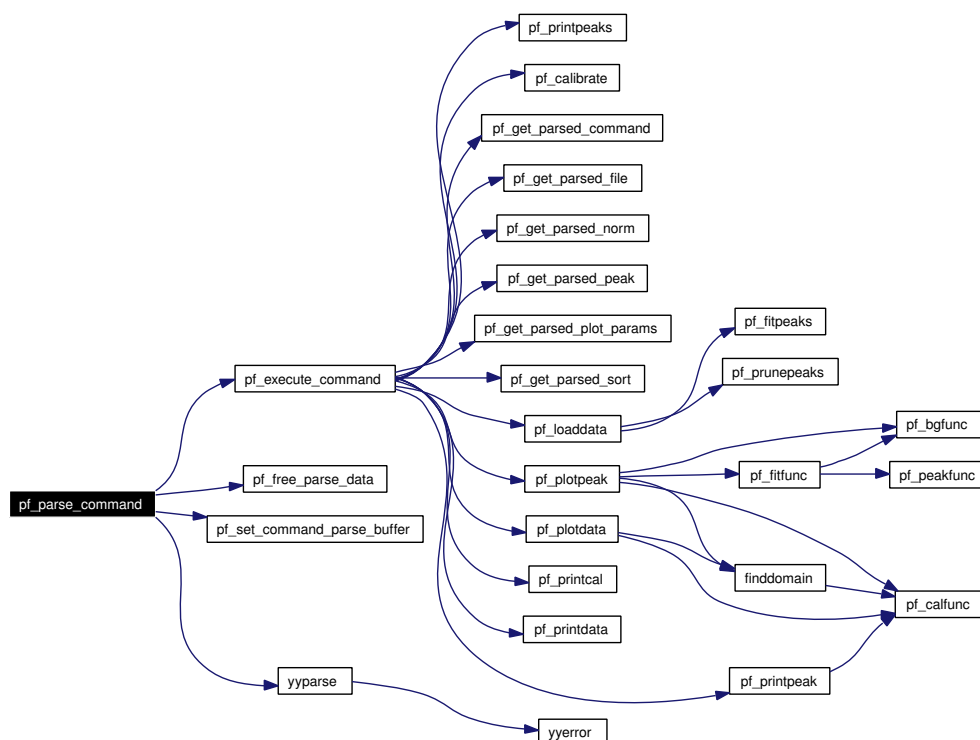
Parameters:

← *command* The command to parse and execute

Returns:

Status code

Here is the call graph for this function:



4.8.3.4 int `yyparse ()`

The generated parsing function.

Returns:

Parsing status code

4.8.4 Variable Documentation

4.8.4.1 struct `pf_cal_fit current_cal`

The current calibration.

4.8.4.2 struct `pf_data current_data`

The current data set.

4.8.4.3 int `has_data`

Valid data-set flag.

4.8.4.4 int `should_quit`

Should the main loop terminate.

4.9 peakfit.c File Reference

4.9.1 Detailed Description

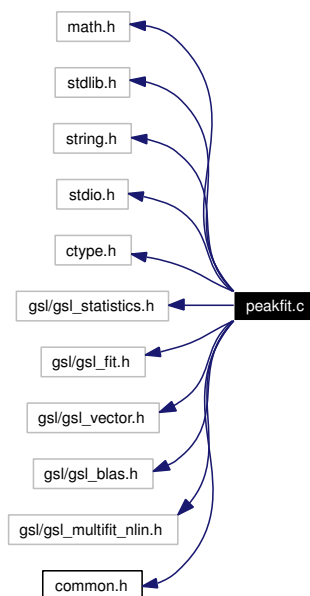
Author:

Hal Finkel

Peak identification and fitting

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <gsl/gsl_statistics.h>
#include <gsl/gsl_fit.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_multifit_nlin.h>
#include "common.h"
```

Include dependency graph for peakfit.c:



Data Structures

- struct [bg_poly_data](#)

Data on the background for fitting.

- struct [peak_pearson7_data](#)

Data on a peak for fitting.

Defines

- #define [MAXITER](#) 100000

The maximum number of iterations for fitting peaks and background.

- #define [MINBGPTS](#) 3

Minimum number of background points per side.

- #define [BGWIDTH](#) 2

The number of peak widths to look for background if possible.

- #define [SEARCHSTEP](#) 0.1

Search step for peak bounds.

Functions

- void [pf_prunepeaks](#) (struct [pf_data](#) *d)

Prune peaks without reasonable fits.

- void [pf_fitpeaks](#) (struct [pf_data](#) *d)

Fit peaks in the data set.

- double [pf_bgfunc](#) (struct [pf_data](#) *d, int peak, double x)

Evaluate the background function for a given peak.

- double [pf_peakfunc](#) (struct [pf_data](#) *d, int peak, double x)

Evaluate the peak fit function for a given peak.

- double [pf_fitfunc](#) (struct [pf_data](#) *d, int peak, double x)

Evaluate the total fit function for a given peak.

4.9.2 Define Documentation

4.9.2.1 #define BGWIDTH 2

The number of peak widths to look for background if possible.

4.9.2.2 #define MAXITER 100000

The maximum number of iterations for fitting peaks and background.

4.9.2.3 #define MINBGPTS 3

Minimum number of background points per side.

Total from both sides needs to be at least the number of fit parameters

4.9.2.4 #define SEARCHSTEP 0.1

Search step for peak bounds.

4.9.3 Function Documentation

4.9.3.1 double pf_bgfunc (struct pf_data * d, int peak, double x)

Evaluate the background function for a given peak.

Parameters:

- ← *d* The data set
- ← *peak* The peak for which to evaluate the background function
- ← *x* The value at which to evaluate the function

Returns:

The value of the function

4.9.3.2 double pf_fitfunc (struct pf_data * d, int peak, double x)

Evaluate the total fit function for a given peak.

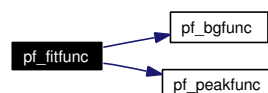
Parameters:

- ← *d* The data set
- ← *peak* The peak for which to evaluate the total fit function
- ← *x* The value at which to evaluate the function

Returns:

The value of the function

Here is the call graph for this function:



4.9.3.3 void pf_fitpeaks (struct pf_data * d)

Fit peaks in the data set.

Parameters:

- ↔ *d* The data set

4.9.3.4 double pf_peakfunc (struct pf_data * *d*, int *peak*, double *x*)

Evaluate the peak fit function for a given peak.

Parameters:

- ← *d* The data set
- ← *peak* The peak for which to evaluate the peak fit function
- ← *x* The value at which to evaluate the function

Returns:

The value of the function

4.9.3.5 void pf_prunepeaks (struct pf_data * *d*)

Prune peaks without reasonable fits.

Parameters:

- ↔ *d* The data set

4.10 plotdata.c File Reference

4.10.1 Detailed Description

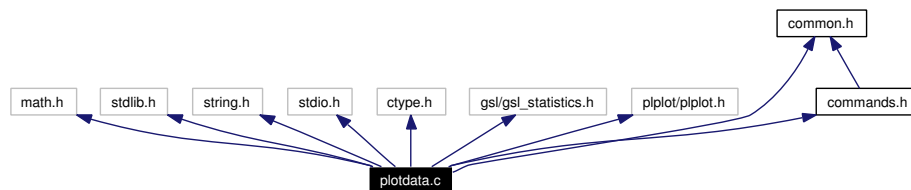
Author:

Hal Finkel

Data and peak plotting

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <gsl/gsl_statistics.h>
#include <plplot/plplot.h>
#include "commands.h"
#include "common.h"
```

Include dependency graph for plotdata.c:



Data Structures

- struct [def_file_dev](#)

An association between a file-name extension and a plotting device name.

Defines

- #define [MAXTITLE](#) 256

The maximum length of the plot title.

- #define [CHANPTS](#) 10

The number of points to draw per channel.

- #define [DEFDEV](#) "xwin"

The default plplot output device.

Functions

- int `finddomain` (struct `pf_data` *d, struct `pf_plot_params` *p, struct `pf_cal_fit` *cal)
Calculate the domain in channels if specified in calibrated units.
- void `pf_plotdata` (struct `pf_data` *d, struct `pf_plot_params` *p, struct `pf_cal_fit` *cal)
Plot data.
- void `pf_plotpeak` (struct `pf_data` *d, struct `pf_plot_params` *p, struct `pf_cal_fit` *cal)
Plot a given peak.

Variables

- const struct `def_file_dev def_file_devs` []
Mapping of file-name extensions to plotting device names.
- const int `def_file_devsn` = sizeof(`def_file_devs`)/sizeof(`def_file_devs`[0])
Number of entries in the `def_file_devs` array.

4.10.2 Define Documentation

4.10.2.1 #define CHANPTS 10

The number of points to draw per channel.

4.10.2.2 #define DEFDEV "xwin"

The default pplot output device.

4.10.2.3 #define MAXTITLE 256

The maximum length of the plot title.

4.10.3 Function Documentation

4.10.3.1 int finddomain (struct `pf_data` * d, struct `pf_plot_params` * p, struct `pf_cal_fit` * cal)

Calculate the domain in channels if specified in calibrated units.

Parameters:

- ← *d* The data set
- ↔ *p* The plot parameters
- ← *cal* The calibration fit

Returns:

Boolean success indicator

Here is the call graph for this function:



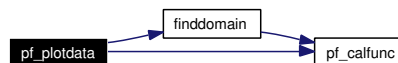
4.10.3.2 void pf_plotdata (struct pf_data * *d*, struct pf_plot_params * *p*, struct pf_cal_fit * *cal*)

Plot data.

Parameters:

- ← *d* The data set
- ← *p* The plotting parameters
- ← *cal* The calibration fit

Here is the call graph for this function:



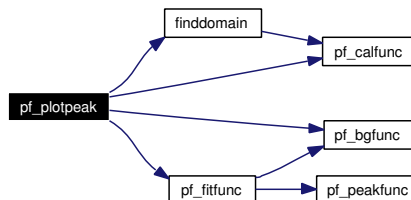
4.10.3.3 void pf_plotpeak (struct pf_data * *d*, struct pf_plot_params * *p*, struct pf_cal_fit * *cal*)

Plot a given peak.

Parameters:

- ← *d* The data set
- ← *p* The plotting parameters
- ← *cal* The calibration fit

Here is the call graph for this function:



4.10.4 Variable Documentation

4.10.4.1 const struct def_file_dev def_file_devs[]

Initial value:

```
{
    { "plmeta", "plmeta" },
    { "plm", "plmeta" },
    { "ps", "ps" },
    { "psc", "psc" },
    { "xfig", "xfig" },
    { "imp", "imp" },
    { "pbm", "pbm" },
    { "jpg", "jpeg" },
    { "jpeg", "jpeg" },
    { "png", "png" },
    { "cgm", "cgm" }
}
```

Mapping of file-name extensions to plotting device names.

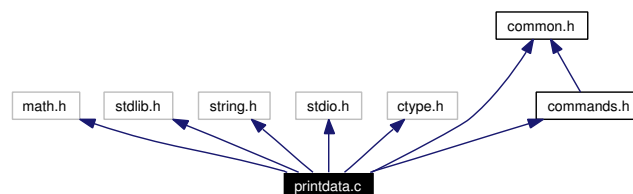
4.10.4.2 `const int def_file_devsn = sizeof(def_file_devs)/sizeof(def_file_devs[0])`

Number of entries in the def_file_devs array.

4.11 printdata.c File Reference

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include "commands.h"
#include "common.h"
```

Include dependency graph for printdata.c:



Functions

- `int pf_printdata` (struct `pf_data` *d, char *fn, int norm)
Print the data set.
- `int pf_printpeaks` (struct `pf_data` *d, char *fn, enum `pf_peak_sort` srt)
Print the peaks.
- `int pf_printpeak` (struct `pf_data` *d, struct `pf_cal_fit` *cal, int peak, char *fn)
Print data about a given peak.
- `int pf_printcal` (struct `pf_data` *d, struct `pf_cal_fit` *f, char *fn)
Print data about the calibration fit.

4.11.1 Function Documentation

4.11.1.1 `int pf_printcal` (struct `pf_data` *d, struct `pf_cal_fit` *f, char *fn)

Print data about the calibration fit.

Parameters:

- ← *d* The data set
- ← *f* The calibration fit
- ← *fn* The output file name

Returns:

Boolean status flag

4.11.1.2 int pf_printdata (struct pf_data * *d*, char * *fn*, int *norm*)

Print the data set.

Parameters:

- ← *d* The data set
- ← *fn* The output file name
- ← *norm* Flag indicating whether to print normalized data

Returns:

Boolean status flag

4.11.1.3 int pf_printpeak (struct pf_data * *d*, struct pf_cal_fit * *cal*, int *peak*, char * *fn*)

Print data about a given peak.

Parameters:

- ← *d* The data set
- ← *cal* The calibration fit
- ← *peak* The peak about which to print data
- ← *fn* The output file name

Returns:

Boolean status flag

Here is the call graph for this function:



4.11.1.4 int pf_printpeaks (struct pf_data * *d*, char * *fn*, enum pf_peak_sort *srt*)

Print the peaks.

Parameters:

- ← *d* The data set
- ← *fn* The output file name
- ← *srt* Peak sort type

Returns:

Boolean status flag

Index

- a
 - pf_cal_fit, 11
 - pf_peak_bg, 22
 - pf_peak_fit, 24
- a_err
 - pf_cal_fit, 11
 - pf_peak_bg, 22
 - pf_peak_fit, 24
- annot
 - pf_plot_params, 27
- ANNOTATE
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- AREA
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- area
 - pf_peak, 20
- b
 - pf_cal_fit, 12
 - pf_peak_bg, 23
- b_err
 - pf_cal_fit, 12
 - pf_peak_bg, 23
- BEGIN
 - cmdlex.c, 39
- bg
 - pf_peak, 20
- bg_poly_data, 5
 - d, 6
 - npts, 6
 - peak, 6
 - pts, 6
- bglbound
 - pf_peak, 20
- bgrbound
 - pf_peak, 20
- BGWIDTH
 - peakfit.c, 88
- BY
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- c
 - pf_cal_fit, 12
 - pf_peak_bg, 23
- c_err
 - pf_cal_fit, 12
 - pf_peak_bg, 23
- cal
 - pf_plot_params, 27
- cal_fit_info, 7
 - npts, 7
 - pk, 7
 - pt, 7
 - x, 7
 - x_err, 7
 - y, 7
- CALIBRATE
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- calibrate.c, 33
 - MAXFLEX, 34
 - MAXITER, 34
 - pf_calfunc, 35
 - pf_calibrate, 35
- CALIBRATED
 - cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- CALIBRATION
 - cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- cbin
 - pf_peak, 20
- center
 - pf_peak, 20
- CHANNEL
 - cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- CHANPTS
 - plotdata.c, 92
- chisq
 - pf_cal_fit, 12
 - pf_peak_bg, 23
 - pf_peak_fit, 25
- chisq_dof
 - pf_cal_fit, 12
 - pf_peak_bg, 23
 - pf_peak_fit, 25

- CLEAR
- cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- cmdlex.c, 36
- BEGIN, 39
 - ECHO, 39
 - EOB_ACT_CONTINUE_SCAN, 39
 - EOB_ACT_END_OF_FILE, 39
 - EOB_ACT_LAST_MATCH, 39
 - file, 44
 - FLEX_SCANNER, 39
 - INITIAL, 39
 - len, 44
 - parse_buffer, 44
 - parse_pos, 44
 - REJECT, 39
 - size, 44
 - unput, 39
 - YY_AT_BOL, 39
 - YY_BREAK, 39
 - YY_BUF_SIZE, 39
 - YY_BUFFER_EOF_PENDING, 39
 - YY_BUFFER_NEW, 39
 - YY_BUFFER_NORMAL, 39
 - YY_BUFFER_STATE, 43
 - YY_CHAR, 43
 - YY_CURRENT_BUFFER, 39
 - YY_DECL, 39
 - YY_DO_BEFORE_ACTION, 39
 - YY_END_OF_BUFFER, 39
 - YY_END_OF_BUFFER_CHAR, 40
 - YY_EXIT_FAILURE, 40
 - YY_FATAL_ERROR, 40
 - YY_FLEX_MAJOR_VERSION, 40
 - YY_FLEX_MINOR_VERSION, 40
 - YY_FLUSH_BUFFER, 40
 - YY_INPUT, 40
 - YY_MORE_ADJ, 40
 - yy_new_buffer, 41
 - YY_NEW_FILE, 41
 - YY_NO_POP_STATE, 41
 - YY_NO_PUSH_STATE, 41
 - YY_NO_TOP_STATE, 41
 - YY_NO_UNPUT, 41
 - YY_NULL, 41
 - YY_NUM_RULES, 41
 - YY_PROTO, 41, 43
 - YY_READ_BUF_SIZE, 41
 - YY_RESTORE_YY_MORE_OFFSET, 41
 - YY_RULE_SETUP, 41
 - YY_SC_TO_UI, 41
 - yy_set_bol, 41
 - yy_set_interactive, 41
 - yy_size_t, 43
 - YY_SKIP_YYWRAP, 41
 - YY_START, 42
 - YY_START_STACK_INCR, 42
 - YY_STATE_EOF, 42
 - yy_state_type, 43
 - yyconst, 42
 - yyin, 44
 - yylen, 44
 - yyless, 42
 - yymore, 42
 - yyout, 44
 - YYSTATE, 43
 - yyterminate, 43
 - yytext, 44
 - yytext_ptr, 43
 - yywrap, 43
- cmdparse.c, 45
- ANNOTATE, 51, 55
 - AREA, 51, 55
 - BY, 51, 55
 - CALIBRATE, 51, 55
 - CALIBRATED, 51, 56
 - CALIBRATION, 51, 56
 - CHANNEL, 51, 56
 - CLEAR, 51, 56
 - DATA, 51, 55
 - DRIVER, 51, 55
 - driver_name, 59
 - HEIGHT, 51, 55
 - LOAD, 51, 55
 - load_file, 59
 - MARKED, 51, 55
 - need_cal, 59
 - need_rescale, 59
 - NORM, 51, 55
 - NUM, 51, 56
 - NUMBER, 51, 55
 - PEAK, 51, 55
 - peak_num, 59
 - PEAKS, 51, 55
 - pf_free_parse_data, 56
 - pf_get_parsed_annot, 56
 - pf_get_parsed_cal, 56
 - pf_get_parsed_command, 56
 - pf_get_parsed_driver, 56
 - pf_get_parsed_file, 56
 - pf_get_parsed_marked, 57
 - pf_get_parsed_norm, 57
 - pf_get_parsed_peak, 57
 - pf_get_parsed_plot_params, 57
 - pf_get_parsed_plot_range, 57
 - pf_get_parsed_rescale, 57
 - pf_get_parsed_rotation, 58
 - pf_get_parsed_sort, 58

- pf_init_parse_data, 58
- PLOT, 51, 55
- plot_range, 59
- PRINT, 51, 55
- QUIT, 51, 55
- RESCALE, 51, 55
- rot_angle, 59
- ROTATED, 51, 55
- should_annot, 59
- should_mark, 59
- SMOOTH, 51, 55
- SORTED, 51, 55
- STRING, 51, 56
- TO, 51, 55
- use_norm, 59
- USING, 51, 55
- YY_REDUCE_PRINT, 51
- YY_STACK_PRINT, 51
- YYABORT, 51
- YYACCEPT, 51
- YYBACKUP, 51
- YYBISON, 52
- yychar, 59
- yyclearin, 52
- YYCOPY, 52
- YYDEBUG, 52
- YYDPRINTF, 53
- YYDSYMPRINT, 53
- YYDSYMPRINTF, 53
- YYEMPTY, 53
- YYEOF, 53
- YYERRCODE, 53
- yyerrok, 53
- YYERROR, 53
- yterror, 58
- YYERROR_VERBOSE, 53
- YYFAIL, 53
- YYFINAL, 53
- YYFREE, 53
- YYINITDEPTH, 53
- YYLAST, 53
- YYLEX, 53
- yyllex, 58
- YYLLOC_DEFAULT, 53
- YYLSP_NEEDED, 53
- yylval, 60
- YYMALLOC, 54
- YYMAXDEPTH, 54
- YYMAXUTOK, 54
- yynerrs, 60
- YYNNTS, 54
- YYNRULES, 54
- YYNSTATES, 54
- YYNTOKENS, 54
- YYPACT_NINF, 54
- yyparse, 58
- YYPOPSTACK, 54
- YYPURE, 54
- YYRECOVERING, 54
- yysigned_char, 55
- YYSIZE_T, 54
- YYSKELETON_NAME, 54
- YYSTACK_ALLOC, 54
- YYSTACK_BYTES, 54
- YYSTACK_FREE, 54
- YYSTACK_GAP_MAXIMUM, 54
- YYSTACK_RELOCATE, 54
- YYSTYPE, 55
- yystate, 54
- YYSTYPE_IS_DECLARED, 55
- YYSTYPE_IS_TRIVIAL, 55
- YYTABLE_NINF, 55
- YYTERROR, 55
- yytokentype, 55
- YYTRANSLATE, 55
- YYUNDEFTOK, 55
- cmdparse.h, 61
 - ANNOTATE, 64, 65
 - AREA, 64, 65
 - BY, 64, 65
 - CALIBRATE, 64, 65
 - CALIBRATED, 64, 65
 - CALIBRATION, 64, 65
 - CHANNEL, 64, 65
 - CLEAR, 64, 65
 - DATA, 64, 65
 - DRIVER, 64, 65
 - HEIGHT, 64, 65
 - LOAD, 64, 65
 - MARKED, 64, 65
 - NORM, 64, 65
 - NUM, 64, 65
 - NUMBER, 64, 65
 - PEAK, 64, 65
 - PEAKS, 64, 65
 - PLOT, 64
 - PRINT, 64, 65
 - QUIT, 64
 - RESCALE, 64, 65
 - ROTATED, 64, 65
 - SMOOTH, 64, 65
 - SORTED, 64, 65
 - STRING, 64, 65
 - TO, 64, 65
 - USING, 64, 65
 - yylval, 65
 - YYSTYPE, 64
 - yystate, 64

- YYSTYPE_IS_DECLARED, 64
- YYSTYPE_IS_TRIVIAL, 64
- yytokentype, 64
- command_calibrate
 - commands.h, 68
- command_clear_calibration
 - commands.h, 68
- command_invalid
 - commands.h, 68
- command_load
 - commands.h, 68
- command_plot_data
 - commands.h, 68
- command_plot_peak
 - commands.h, 68
- command_print_calibration
 - commands.h, 68
- command_print_data
 - commands.h, 68
- command_print_peak
 - commands.h, 68
- command_print_peaks
 - commands.h, 68
- command_quit
 - commands.h, 68
- commands.h, 66
 - command_calibrate, 68
 - command_clear_calibration, 68
 - command_invalid, 68
 - command_load, 68
 - command_plot_data, 68
 - command_plot_peak, 68
 - command_print_calibration, 68
 - command_print_data, 68
 - command_print_peak, 68
 - command_print_peaks, 68
 - command_quit, 68
 - pf_command, 68
 - pf_execute_command, 68
 - pf_free_parse_data, 68
 - pf_get_parsed_annot, 69
 - pf_get_parsed_cal, 69
 - pf_get_parsed_command, 69
 - pf_get_parsed_driver, 69
 - pf_get_parsed_file, 69
 - pf_get_parsed_marked, 69
 - pf_get_parsed_norm, 69
 - pf_get_parsed_peak, 70
 - pf_get_parsed_plot_params, 70
 - pf_get_parsed_plot_range, 70
 - pf_get_parsed_rescale, 70
 - pf_get_parsed_rotation, 70
 - pf_get_parsed_sort, 70
 - pf_init_parse_data, 71
 - pf_set_command_parse_buffer, 71
- common.h, 72
 - MAXCALPTS, 74
 - MAXUNIT, 74
 - NCHAN, 74
 - pf_bgfunc, 74
 - pf_calfunc, 75
 - pf_calibrate, 75
 - pf_chanwindowavg, 75
 - pf_fitfunc, 75
 - pf_fitpeaks, 76
 - pf_loaddata, 76
 - pf_parse_command, 76
 - pf_peak_sort, 74
 - pf_peakfunc, 77
 - pf_plotdata, 77
 - pf_plotpeak, 78
 - pf_printcal, 78
 - pf_printdata, 78
 - pf_printpeak, 79
 - pf_printpeaks, 79
 - pf_prunepeaks, 79
 - sort_area, 74
 - sort_height, 74
 - sort_number, 74
- conv
 - pf_cal_fit, 12
 - pf_peak_bg, 23
 - pf_peak_fit, 25
- current_cal
 - main.c, 86
- current_data
 - main.c, 86
- d
 - bg_poly_data, 6
 - peak_pearson7_data, 10
 - pf_peak_bg, 23
- d_err
 - pf_peak_bg, 23
- DATA
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- data_bg
 - pf_data, 17
- data_norm
 - pf_data, 17
- data_raw
 - pf_data, 17
- def_file_dev, 9
 - dev, 9
 - ext, 9
- def_file_devs
 - plotdata.c, 93

- def_file_devsn
 - plotdata.c, 94
- DEFDEV
 - plotdata.c, 92
- dev
 - def_file_dev, 9
 - pf_plot_params, 27
- DRIVER
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- driver_name
 - cmdparse.c, 59
- ECHO
 - cmdlex.c, 39
- EOB_ACT_CONTINUE_SCAN
 - cmdlex.c, 39
- EOB_ACT_END_OF_FILE
 - cmdlex.c, 39
- EOB_ACT_LAST_MATCH
 - cmdlex.c, 39
- ext
 - def_file_dev, 9
- file
 - cmdlex.c, 44
- finddomain
 - plotdata.c, 92
- fit
 - pf_peak, 20
- FLEX_SCANNER
 - cmdlex.c, 39
- fn
 - pf_plot_params, 27
- has_data
 - main.c, 86
- HEIGHT
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- height
 - pf_peak, 20
- HISTORY_FILE
 - main.c, 83
- INITIAL
 - cmdlex.c, 39
- k
 - pf_peak_fit, 25
- k_err
 - pf_peak_fit, 25
- lbin
 - pf_peak, 20
- lbound
 - pf_peak, 21
- len
 - cmdlex.c, 44
- LOAD
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- load_file
 - cmdparse.c, 59
- loaddata.c, 80
 - MINWIDTH, 81
 - pf_chanwindowavg, 81
 - pf_loaddata, 81
- m
 - pf_peak_fit, 25
- m_err
 - pf_peak_fit, 25
- main
 - main.c, 83
- main.c, 82
 - current_cal, 86
 - current_data, 86
 - has_data, 86
 - HISTORY_FILE, 83
 - main, 83
 - MAXCMDLINE, 83
 - pf_execute_command, 84
 - pf_parse_command, 85
 - should_quit, 86
 - yyparse, 86
- MARKED
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- marked
 - pf_plot_params, 27
- max_norm
 - pf_data, 17
- max_raw
 - pf_data, 17
- MAXCALPTS
 - common.h, 74
- MAXCMDLINE
 - main.c, 83
- MAXFLEX
 - calibrate.c, 34
- MAXITER
 - calibrate.c, 34
 - peakfit.c, 88
- MAXTITLE
 - plotdata.c, 92
- MAXUNIT
 - common.h, 74
- min_norm

- pf_data, 17
- min_raw
 - pf_data, 17
- MINBGPTS
 - peakfit.c, 88
- MINWIDTH
 - loaddata.c, 81
- NCHAN
 - common.h, 74
- need_cal
 - cmdparse.c, 59
- need_rescale
 - cmdparse.c, 59
- NORM
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- norm
 - pf_plot_params, 27
- npeaks
 - pf_data, 17
- npts
 - bg_poly_data, 6
 - cal_fit_info, 7
 - pf_cal_pts, 14
- NUM
 - cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- num
 - pf_cal_pt, 13
 - pf_peak, 21
 - YYSTYPE, 32
- NUMBER
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- parse_buffer
 - cmdlex.c, 44
- parse_pos
 - cmdlex.c, 44
- PEAK
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- peak
 - bg_poly_data, 6
 - peak_pearson7_data, 10
 - pf_plot_params, 27
- peak_num
 - cmdparse.c, 59
- peak_pearson7_data, 10
 - d, 10
 - peak, 10
- peakfit.c, 87
 - BGWIDTH, 88
 - MAXITER, 88
 - MINBGPTS, 88
 - pf_bfunc, 89
 - pf_fitfunc, 89
 - pf_fitpeaks, 89
 - pf_peakfunc, 89
 - pf_prunepeaks, 90
 - SEARCHSTEP, 89
- PEAKS
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- peaks
 - pf_data, 18
- peaks_by_area
 - pf_data, 18
- peaks_by_height
 - pf_data, 18
- peaks_by_number
 - pf_data, 18
- pf_bfunc
 - common.h, 74
 - peakfit.c, 89
- pf_cal_fit, 11
 - a, 11
 - a_err, 11
 - b, 12
 - b_err, 12
 - c, 12
 - c_err, 12
 - chisq, 12
 - chisq_dof, 12
 - conv, 12
 - unit, 12
 - valid, 12
- pf_cal_pt, 13
 - num, 13
 - pt, 13
 - ri, 13
- pf_cal_pts, 14
 - npts, 14
 - pts, 14
 - pts_by_ri, 14
 - pts_by_value, 14
 - unit, 15
- pf_calfunc
 - calibrate.c, 35
 - common.h, 75
- pf_calibrate
 - calibrate.c, 35
 - common.h, 75
- pf_chanwindowavg
 - common.h, 75
 - loaddata.c, 81
- pf_command

- commands.h, 68
- pf_data, 16
 - data_bg, 17
 - data_norm, 17
 - data_raw, 17
 - max_norm, 17
 - max_raw, 17
 - min_norm, 17
 - min_raw, 17
 - npeaks, 17
 - peaks, 18
 - peaks_by_area, 18
 - peaks_by_height, 18
 - peaks_by_number, 18
 - total_chisq, 18
 - total_counts, 18
- pf_execute_command
 - commands.h, 68
 - main.c, 84
- pf_fitfunc
 - common.h, 75
 - peakfit.c, 89
- pf_fitpeaks
 - common.h, 76
 - peakfit.c, 89
- pf_free_parse_data
 - cmdparse.c, 56
 - commands.h, 68
- pf_get_parsed_annot
 - cmdparse.c, 56
 - commands.h, 69
- pf_get_parsed_cal
 - cmdparse.c, 56
 - commands.h, 69
- pf_get_parsed_command
 - cmdparse.c, 56
 - commands.h, 69
- pf_get_parsed_driver
 - cmdparse.c, 56
 - commands.h, 69
- pf_get_parsed_file
 - cmdparse.c, 56
 - commands.h, 69
- pf_get_parsed_marked
 - cmdparse.c, 57
 - commands.h, 69
- pf_get_parsed_norm
 - cmdparse.c, 57
 - commands.h, 69
- pf_get_parsed_peak
 - cmdparse.c, 57
 - commands.h, 70
- pf_get_parsed_plot_params
 - cmdparse.c, 57
 - commands.h, 70
- pf_get_parsed_plot_range
 - cmdparse.c, 57
 - commands.h, 70
- pf_get_parsed_rescale
 - cmdparse.c, 57
 - commands.h, 70
- pf_get_parsed_rotation
 - cmdparse.c, 58
 - commands.h, 70
- pf_get_parsed_sort
 - cmdparse.c, 58
 - commands.h, 70
- pf_init_parse_data
 - cmdparse.c, 58
 - commands.h, 71
- pf_loaddata
 - common.h, 76
 - loaddata.c, 81
- pf_parse_command
 - common.h, 76
 - main.c, 85
- pf_peak, 19
 - area, 20
 - bg, 20
 - bglbound, 20
 - bgrbound, 20
 - cbin, 20
 - center, 20
 - fit, 20
 - height, 20
 - lbin, 20
 - lbound, 21
 - num, 21
 - rbin, 21
 - rbound, 21
 - width, 21
- pf_peak_bg, 22
 - a, 22
 - a_err, 22
 - b, 23
 - b_err, 23
 - c, 23
 - c_err, 23
 - chisq, 23
 - chisq_dof, 23
 - conv, 23
 - d, 23
 - d_err, 23
- pf_peak_fit, 24
 - a, 24
 - a_err, 24
 - chisq, 25
 - chisq_dof, 25

- conv, 25
- k, 25
- k_err, 25
- m, 25
- m_err, 25
- x0, 25
- x0_err, 25
- pf_peak_sort
 - common.h, 74
- pf_peakfunc
 - common.h, 77
 - peakfit.c, 89
- pf_plot_params, 26
 - annot, 27
 - cal, 27
 - dev, 27
 - fn, 27
 - marked, 27
 - norm, 27
 - peak, 27
 - rescale, 27
 - rot, 27
 - xend, 27
 - xendcal, 27
 - xstart, 27
 - xstartcal, 28
- pf_plot_range, 29
 - xend, 29
 - xendcal, 29
 - xstart, 29
 - xstartcal, 29
- pf_plotdata
 - common.h, 77
 - plotdata.c, 93
- pf_plotpeak
 - common.h, 78
 - plotdata.c, 93
- pf_printcal
 - common.h, 78
 - printdata.c, 95
- pf_printdata
 - common.h, 78
 - printdata.c, 95
- pf_printpeak
 - common.h, 79
 - printdata.c, 96
- pf_printpeaks
 - common.h, 79
 - printdata.c, 96
- pf_prunepeaks
 - common.h, 79
 - peakfit.c, 90
- pf_set_command_parse_buffer
 - commands.h, 71
- pk
 - cal_fit_info, 7
- PLOT
 - cmdparse.c, 51, 55
 - cmdparse.h, 64
- plot_range
 - cmdparse.c, 59
- plotdata.c, 91
 - CHANPTS, 92
 - def_file_devs, 93
 - def_file_devsn, 94
 - DEFDEV, 92
 - finddomain, 92
 - MAXTITLE, 92
 - pf_plotdata, 93
 - pf_plotpeak, 93
- PRINT
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- printdata.c, 95
 - pf_printcal, 95
 - pf_printdata, 95
 - pf_printpeak, 96
 - pf_printpeaks, 96
- pt
 - cal_fit_info, 7
 - pf_cal_pt, 13
- pts
 - bg_poly_data, 6
 - pf_cal_pts, 14
- pts_by_ri
 - pf_cal_pts, 14
- pts_by_value
 - pf_cal_pts, 14
- QUIT
 - cmdparse.c, 51, 55
 - cmdparse.h, 64
- rbin
 - pf_peak, 21
- rbound
 - pf_peak, 21
- REJECT
 - cmdlex.c, 39
- RESCALE
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- rescale
 - pf_plot_params, 27
- ri
 - pf_cal_pt, 13
- rot
 - pf_plot_params, 27

- rot_angle
 - cmdparse.c, 59
- ROTATED
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- SEARCHSTEP
 - peakfit.c, 89
- should_annot
 - cmdparse.c, 59
- should_mark
 - cmdparse.c, 59
- should_quit
 - main.c, 86
- size
 - cmdlex.c, 44
- SMOOTH
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- sort_area
 - common.h, 74
- sort_height
 - common.h, 74
- sort_number
 - common.h, 74
- SORTED
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- str
 - YYSTYPE, 32
- STRING
 - cmdparse.c, 51, 56
 - cmdparse.h, 64, 65
- TO
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- total_chisq
 - pf_data, 18
- total_counts
 - pf_data, 18
- unit
 - pf_cal_fit, 12
 - pf_cal_pts, 15
- unput
 - cmdlex.c, 39
- use_norm
 - cmdparse.c, 59
- USING
 - cmdparse.c, 51, 55
 - cmdparse.h, 64, 65
- valid
 - pf_cal_fit, 12
- width
 - pf_peak, 21
- x
 - cal_fit_info, 7
- x0
 - pf_peak_fit, 25
- x0_err
 - pf_peak_fit, 25
- x_err
 - cal_fit_info, 7
- xend
 - pf_plot_params, 27
 - pf_plot_range, 29
- xendcal
 - pf_plot_params, 27
 - pf_plot_range, 29
- xstart
 - pf_plot_params, 27
 - pf_plot_range, 29
- xstartcal
 - pf_plot_params, 28
 - pf_plot_range, 29
- y
 - cal_fit_info, 7
- YY_AT_BOL
 - cmdlex.c, 39
- yy_at_bol
 - yy_buffer_state, 30
- YY_BREAK
 - cmdlex.c, 39
- yy_buf_pos
 - yy_buffer_state, 30
- YY_BUF_SIZE
 - cmdlex.c, 39
- yy_buf_size
 - yy_buffer_state, 30
- YY_BUFFER_EOF_PENDING
 - cmdlex.c, 39
- YY_BUFFER_NEW
 - cmdlex.c, 39
- YY_BUFFER_NORMAL
 - cmdlex.c, 39
- YY_BUFFER_STATE
 - cmdlex.c, 43
- yy_buffer_state, 30
 - yy_at_bol, 30
 - yy_buf_pos, 30
 - yy_buf_size, 30
 - yy_buffer_status, 30
 - yy_ch_buf, 30

- yy_fill_buffer, 30
- yy_input_file, 30
- yy_is_interactive, 30
- yy_is_our_buffer, 30
- yy_n_chars, 30
- yy_buffer_status
 - yy_buffer_state, 30
- yy_ch_buf
 - yy_buffer_state, 30
- YY_CHAR
 - cmdlex.c, 43
- YY_CURRENT_BUFFER
 - cmdlex.c, 39
- YY_DECL
 - cmdlex.c, 39
- YY_DO_BEFORE_ACTION
 - cmdlex.c, 39
- YY_END_OF_BUFFER
 - cmdlex.c, 39
- YY_END_OF_BUFFER_CHAR
 - cmdlex.c, 40
- YY_EXIT_FAILURE
 - cmdlex.c, 40
- YY_FATAL_ERROR
 - cmdlex.c, 40
- yy_fill_buffer
 - yy_buffer_state, 30
- YY_FLEX_MAJOR_VERSION
 - cmdlex.c, 40
- YY_FLEX_MINOR_VERSION
 - cmdlex.c, 40
- YY_FLUSH_BUFFER
 - cmdlex.c, 40
- YY_INPUT
 - cmdlex.c, 40
- yy_input_file
 - yy_buffer_state, 30
- yy_is_interactive
 - yy_buffer_state, 30
- yy_is_our_buffer
 - yy_buffer_state, 30
- YY_MORE_ADJ
 - cmdlex.c, 40
- yy_n_chars
 - yy_buffer_state, 30
- yy_new_buffer
 - cmdlex.c, 41
- YY_NEW_FILE
 - cmdlex.c, 41
- YY_NO_POP_STATE
 - cmdlex.c, 41
- YY_NO_PUSH_STATE
 - cmdlex.c, 41
- YY_NO_TOP_STATE
 - cmdlex.c, 41
- YY_NO_UNPUT
 - cmdlex.c, 41
- YY_NULL
 - cmdlex.c, 41
- YY_NUM_RULES
 - cmdlex.c, 41
- YY_PROTO
 - cmdlex.c, 41, 43
- YY_READ_BUF_SIZE
 - cmdlex.c, 41
- YY_REDUCE_PRINT
 - cmdparse.c, 51
- YY_RESTORE_YY_MORE_OFFSET
 - cmdlex.c, 41
- YY_RULE_SETUP
 - cmdlex.c, 41
- YY_SC_TO_UI
 - cmdlex.c, 41
- yy_set_bol
 - cmdlex.c, 41
- yy_set_interactive
 - cmdlex.c, 41
- yy_size_t
 - cmdlex.c, 43
- YY_SKIP_YYWRAP
 - cmdlex.c, 41
- YY_STACK_PRINT
 - cmdparse.c, 51
- YY_START
 - cmdlex.c, 42
- YY_START_STACK_INCR
 - cmdlex.c, 42
- YY_STATE_EOF
 - cmdlex.c, 42
- yy_state_type
 - cmdlex.c, 43
- YYABORT
 - cmdparse.c, 51
- YYACCEPT
 - cmdparse.c, 51
- yyalloc, 31
 - yyss, 31
 - yyvs, 31
- YYBACKUP
 - cmdparse.c, 51
- YYBISON
 - cmdparse.c, 52
- yychar
 - cmdparse.c, 59
- yyclearin
 - cmdparse.c, 52
- yyconst
 - cmdlex.c, 42

- YYCOPY
 - cmdparse.c, 52
- YYDEBUG
 - cmdparse.c, 52
- YYDPRINTF
 - cmdparse.c, 53
- YYDSYMPRINT
 - cmdparse.c, 53
- YYDSYMPRINTF
 - cmdparse.c, 53
- YYEMPTY
 - cmdparse.c, 53
- YYEOF
 - cmdparse.c, 53
- YYERRCODE
 - cmdparse.c, 53
- yyerrok
 - cmdparse.c, 53
- YYERROR
 - cmdparse.c, 53
- yyerror
 - cmdparse.c, 58
- YYERROR_VERBOSE
 - cmdparse.c, 53
- YYFAIL
 - cmdparse.c, 53
- YYFINAL
 - cmdparse.c, 53
- YYFREE
 - cmdparse.c, 53
- yyin
 - cmdlex.c, 44
- YYINITDEPTH
 - cmdparse.c, 53
- YYLAST
 - cmdparse.c, 53
- yyleng
 - cmdlex.c, 44
- yyless
 - cmdlex.c, 42
- YYLEX
 - cmdparse.c, 53
- yylex
 - cmdparse.c, 58
- YYLOC_DEFAULT
 - cmdparse.c, 53
- YYLSP_NEEDED
 - cmdparse.c, 53
- yylval
 - cmdparse.c, 60
 - cmdparse.h, 65
- YYMALLOC
 - cmdparse.c, 54
- YYMAXDEPTH
 - cmdparse.c, 54
- YYMAXUTOK
 - cmdparse.c, 54
- yymore
 - cmdlex.c, 42
- yynerrs
 - cmdparse.c, 60
- YYNNTS
 - cmdparse.c, 54
- YYNRULES
 - cmdparse.c, 54
- YYNSTATES
 - cmdparse.c, 54
- YYNTOKENS
 - cmdparse.c, 54
- yyout
 - cmdlex.c, 44
- YYPACT_NINF
 - cmdparse.c, 54
- yyparse
 - cmdparse.c, 58
 - main.c, 86
- YYPOPSTACK
 - cmdparse.c, 54
- YYPURE
 - cmdparse.c, 54
- YYRECOVERING
 - cmdparse.c, 54
- yysigned_char
 - cmdparse.c, 55
- YYSIZE_T
 - cmdparse.c, 54
- YYSKELETON_NAME
 - cmdparse.c, 54
- yyss
 - yyalloc, 31
- YYSTACK_ALLOC
 - cmdparse.c, 54
- YYSTACK_BYTES
 - cmdparse.c, 54
- YYSTACK_FREE
 - cmdparse.c, 54
- YYSTACK_GAP_MAXIMUM
 - cmdparse.c, 54
- YYSTACK_RELOCATE
 - cmdparse.c, 54
- YYSTATE
 - cmdlex.c, 43
- YYSTYPE, 32
 - cmdparse.c, 55
 - cmdparse.h, 64
 - num, 32
 - str, 32
- yystate

cmdparse.c, [54](#)
cmdparse.h, [64](#)
YYSTYPE_IS_DECLARED
cmdparse.c, [55](#)
cmdparse.h, [64](#)
YYSTYPE_IS_TRIVIAL
cmdparse.c, [55](#)
cmdparse.h, [64](#)
YYTABLE_NINF
cmdparse.c, [55](#)
yyterminate
cmdlex.c, [43](#)
YYTERROR
cmdparse.c, [55](#)
yytext
cmdlex.c, [44](#)
yytext_ptr
cmdlex.c, [43](#)
yytokentype
cmdparse.c, [55](#)
cmdparse.h, [64](#)
YYTRANSLATE
cmdparse.c, [55](#)
YYUNDEFTOK
cmdparse.c, [55](#)
yyvs
yyalloc, [31](#)
yywrap
cmdlex.c, [43](#)