sac-format

0.6.0

Generated by Doxygen 1.9.8

Chapter 1

Introduction

sac-format is a single-header statically linked library designed to make working with binary `SAC`-files as easy as possible. Written in C++20, it follows a modern and easy to read programming-style while providing the high performance brought by C++.

sac-format's developed on `GitHub`!

Download `sac-format` from the GitHub release page.

`Download` an offline version of the documentation (PDF).

Get `help` from the community forum.

1.1 Why sac-format

sac-format is Free and Open Source Software (FOSS) released under the MIT license. Anyone can use it, for any purpose (including proprietary software), anywhere in the world. sac-format is operating system agnostic and confirmed working on Windows, macOS, and Linux systems.

1.1.1 Safe

sac-format is **safe** it conforms to a strict set of C++ programming guidelines, chosen to ensure safe code-execution. The guideline conformance list is in `cpp-linter.yml` and can be cross-referenced against this `master list`. Results of conformance checking are `here`.

Testing is an important part of software development; the sac-format library is extensively tested using the `Catch2` testing framework. Everything from low-level binary conversions to high-level `Trace` reading/writing are tested and confirmed working. Check and run the tests yourself. See the Testing section for more information.

1.1.2 Fast

sac-format is **fast** it's written in C++, carefully optimized, and extensively benchmarked. You can run the benchmarks yourself to find out how sac-format performs on your system. See the Benchmarking section for more information.

1.1.3 Easy

sac-format is **easy** single-header makes integration in any project simple. Installation is easy with our automatic installers. Building is a breeze with `CMake`, even on different platforms. Object-oriented design makes use easy and intuitive. See the Quickstart section to get up and running.

1.1.4 Small

sac-format is **small** in total (header + implementation; excluding comments) the library is under 2100∗ lines of code. Small size opens the door to using on any sort of hardware (old or new) and makes it easy to expand upon.

∗ This value includes only the library, excluding all testing/benchmarking and example codes. Including `utests.cpp`, `benchmark.cpp`, `util.hpp`, the example program (`list_sac`), and sac-format totals just over 5100 lines of code.

1.1.5 Documented

sac-format is extensively **documented** both online and in the code. Nothing's hidden, nothing's obscured. Curious how something works? Check the documentation and in-code comments.

1.1.6 Transparent

sac-format is **transparent** all analysis and coverage information is publicly available online.

- `CodeFactor`
- `Codacy`
- `CodeCov`
- `Coverity Scan`

1.1.7 Trace Class

sac-format includes the `Trace` class for seismic traces, providing high-level object-oriented abstraction to seismic data. With the `Trace` class, you don't need to worry about manually reading SAC-files word-by-word. It's compatible with `v6` and `v7` SAC-files and can automatically detect the version upon reading. File output defaults to `v7` SAC-files and there is a `legacy_write` function for `v6` output.

1.1.8 Low-Level I/O

If you want to roll your own SAC-file processing workflow you can use the low-level I/O functionality built into sac-format. All functions tested and confirmed working they're used to build the `Trace` class!

Chapter 2

Installation

This section provides installation instructions.

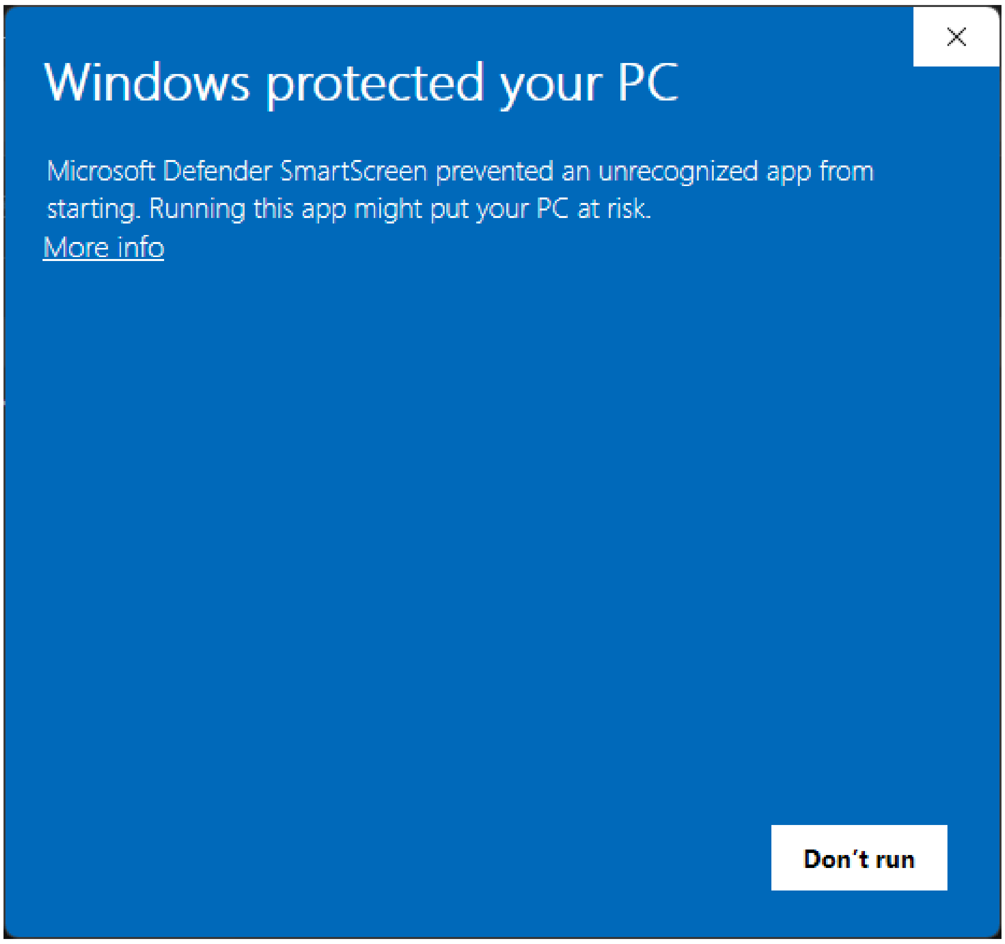
The easiest way to use sac-format is to install it via the automatic installers. Installers for the latest release are located `here`. Be sure to check the sha512 checksum of the installer against its correspondingly named `.sha512` file to ensure the file is safe (for example: `sac-format.pkg` corresponds to `sac-format.pkg.sha512`).

2.1 Windows

sac-format provides a graphical installer on Windows (`sac-format.exe`).

Always check the sha512 checksum value of the installer (`sac-format.exe`; `more info here`) against `sac-format.exe.sha512`.

By default, Microsoft Defender will block the installer with a pop-up like that one below:

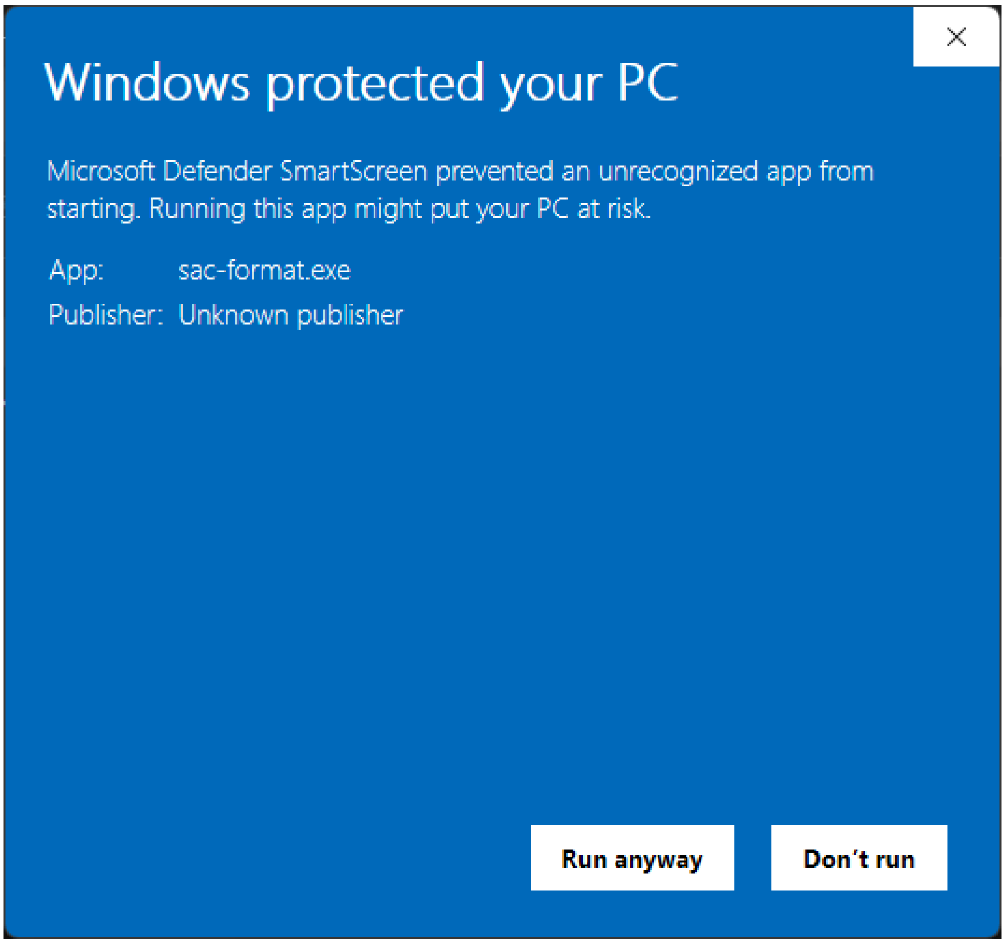

Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

More info

Figure 2.1 Windows Warning 1

To continue the install, click on the "More Info" link and then the "Run anyway" button as seen in the following image:



Windows protected your PC

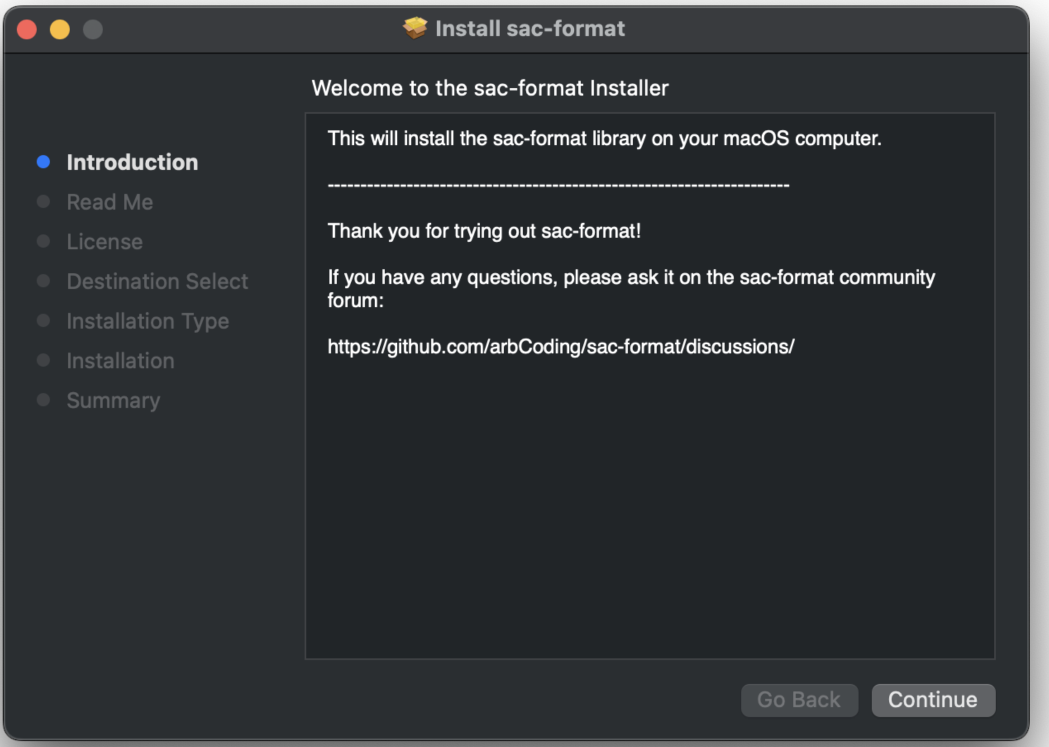
Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: sac-format.exe

Publisher: Unknown publisher

Figure 2.2 Windows Warning 2

Then the installer will open and present you with the welcome screen:



Welcome to the sac-format Installer

This will install the sac-format library on your macOS computer.

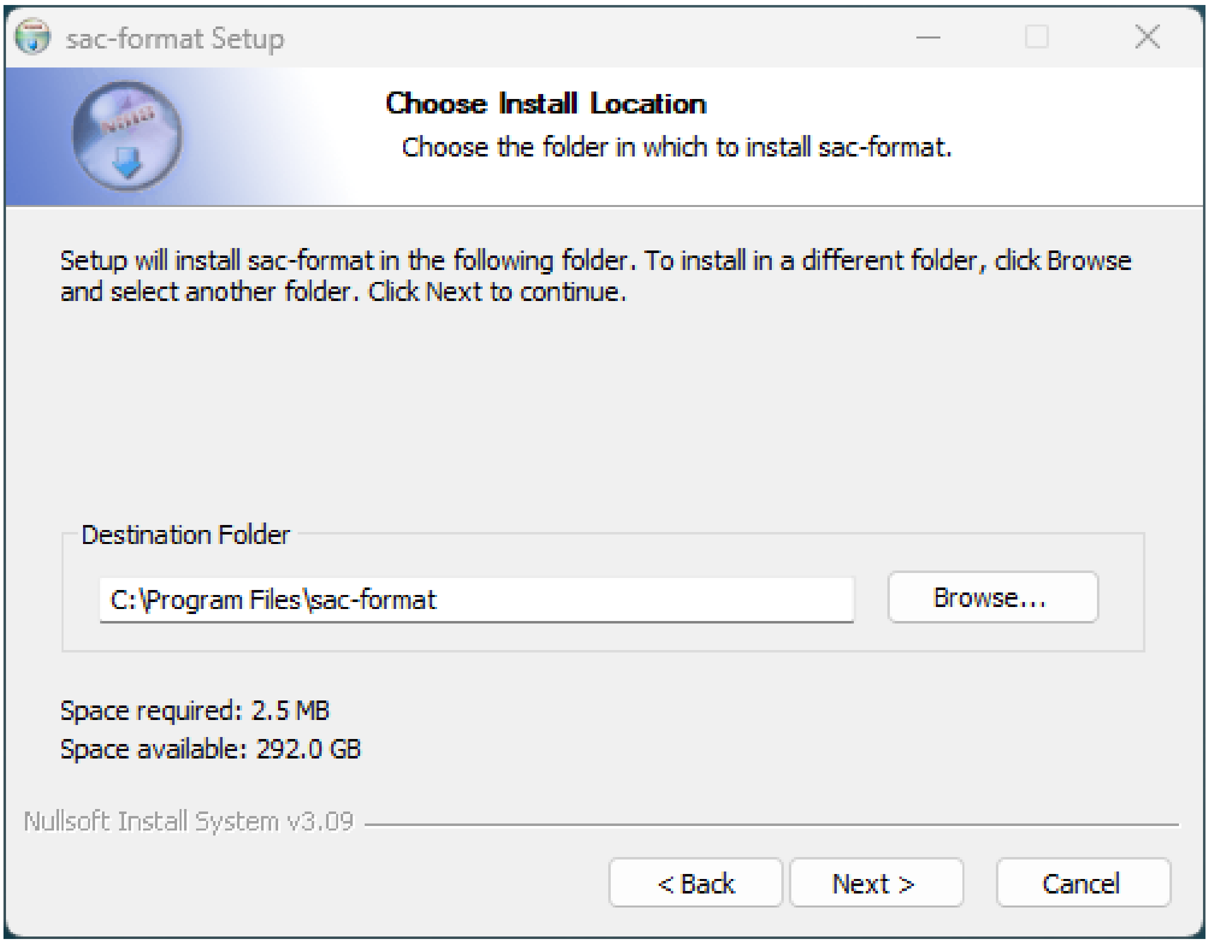
Thank you for trying out sac-format!

If you have any questions, please ask it on the sac-format community forum:

https://github.com/arbCoding/sac-format/discussions/

Figure 2.3 Windows Intro Install

By default, sac-format installs in `C:/Program Files/sac-format` as seen in the screen below:


sac-format Setup

Choose Install Location

Choose the folder in which to install sac-format.

Setup will install sac-format in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

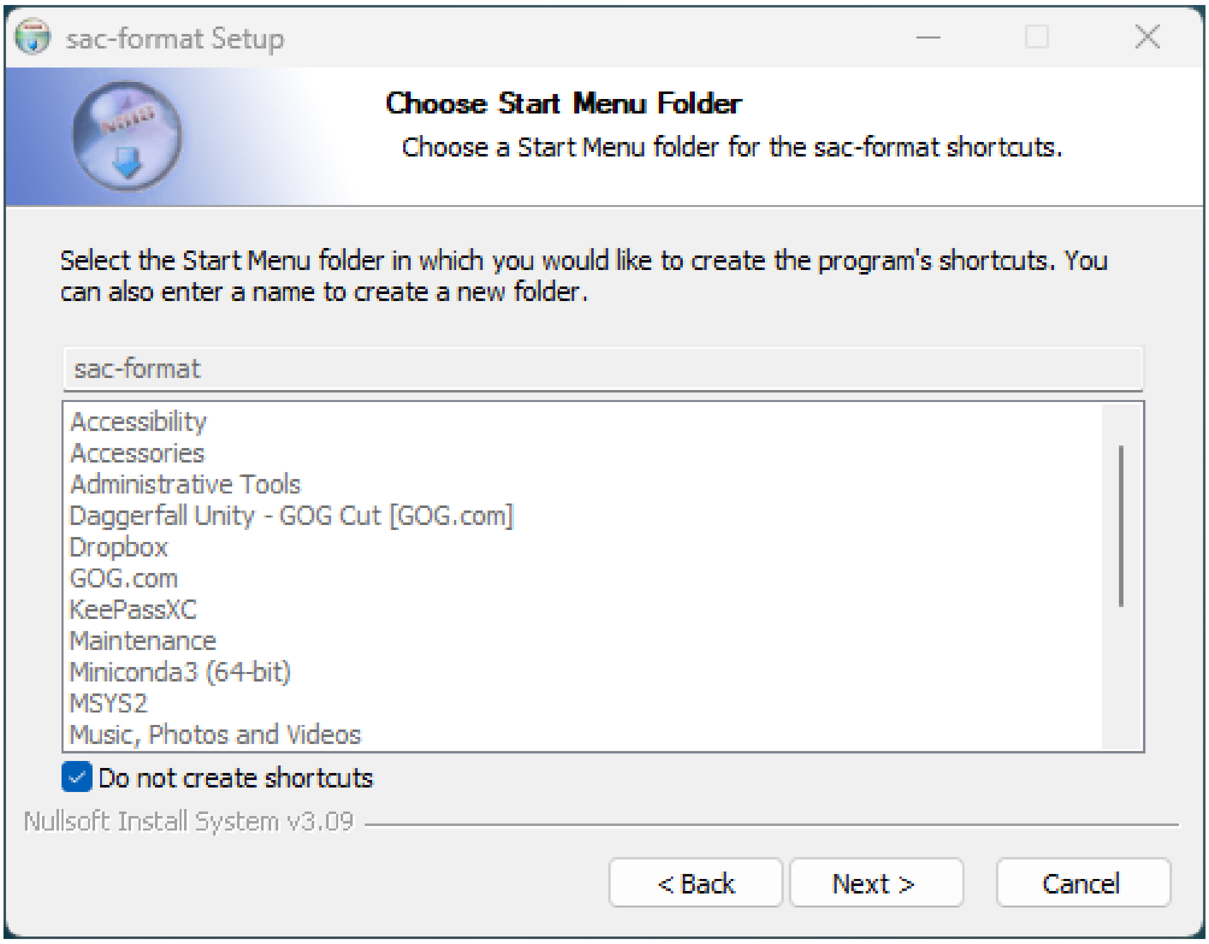
C:\Program Files\sac-format

Space required: 2.5 MB
Space available: 292.0 GB

Nullsoft Install System v3.09

Figure 2.4 Windows Location Install

Because all programs in sac-format are command-line based feel free to disable Start Menu shortcuts:


sac-format Setup

Choose Start Menu Folder

Choose a Start Menu folder for the sac-format shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

sac-format

Accessibility
Accessories
Administrative Tools
Daggerfall Unity - GOG Cut [GOG.com]
Dropbox
GOG.com
KeePassXC
Maintenance
Miniconda3 (64-bit)
MSYS2
Music, Photos and Videos

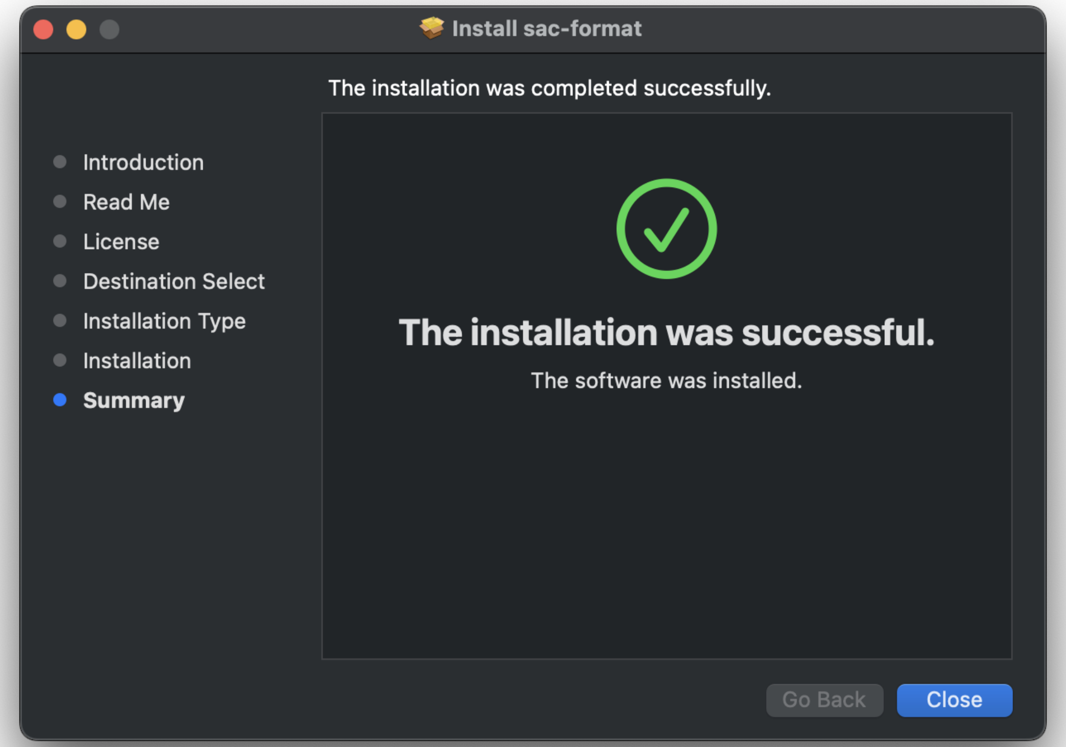
[✓] Do not create shortcuts

Nullsoft Install System v3.09

< Back | Next > | Cancel

Figure 2.5 Windows No Shortcuts

Upon successful install of sac-format you will see this window:



The installation was completed successfully.

The installation was successful.

The software was installed.

Figure 2.6 Windows Install Success

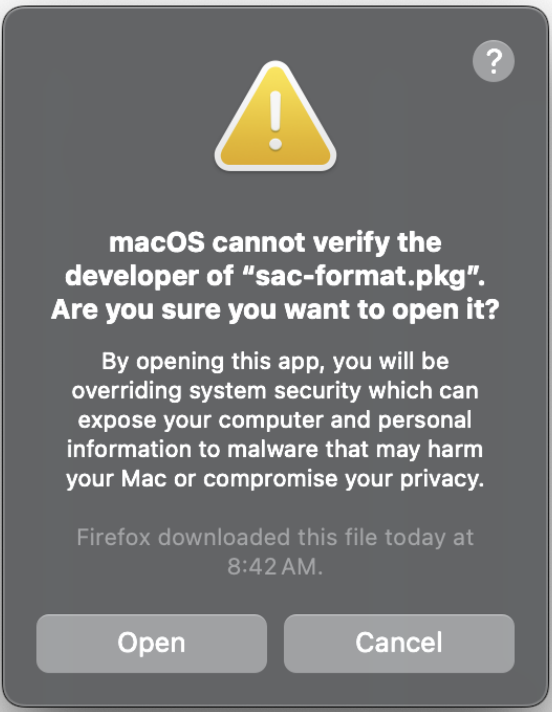
2.2 macOS

sac-format provides both command line and graphical installers on macOS.

2.2.1 Graphical

The graphical installer is `sac-format.pkg` and will walk you through the installation process. **NOTE**: the default installation location is `/opt/sac-format`.

By default, macOS will block the installer. To install, right-click on `sac-format.pkg` and select open. A warning will pop up that looks like:



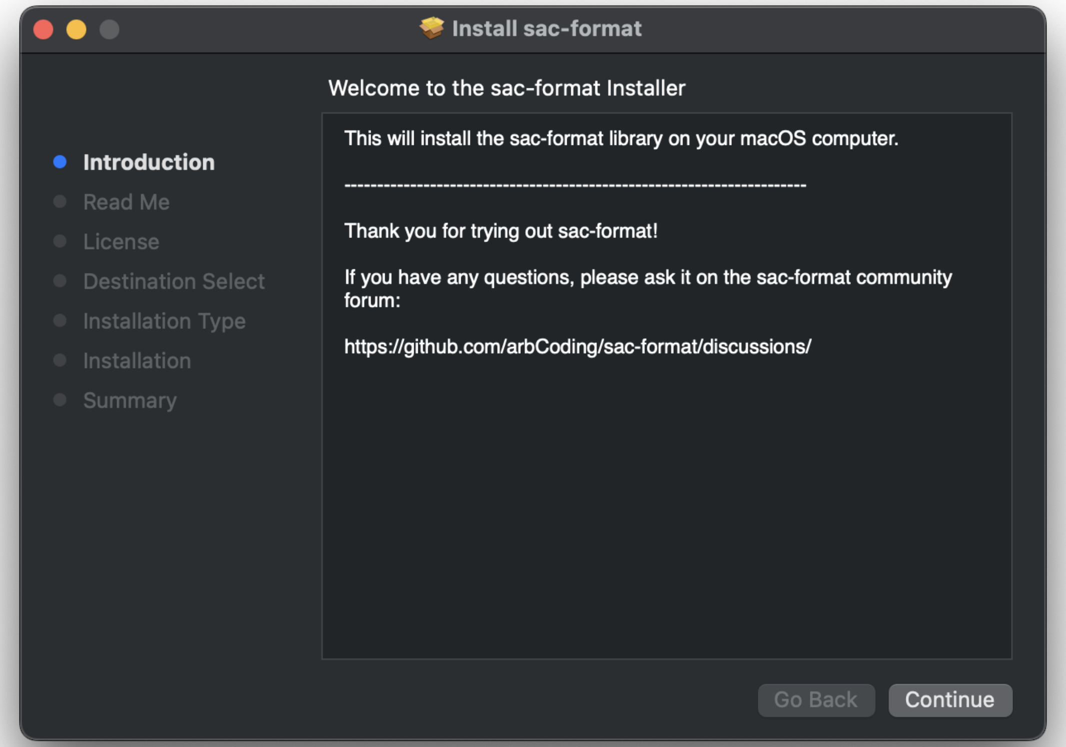
macOS cannot verify the developer of "sac-format.pkg". Are you sure you want to open it?

By opening this app, you will be overriding system security which can expose your computer and personal information to malware that may harm your Mac or compromise your privacy.

Firefox downloaded this file today at 8:42 AM.

Figure 2.7 macOS Warning

Simply click "Open" and the installer will begin from the first screen:



Welcome to the sac-format Installer

This will install the sac-format library on your macOS computer.

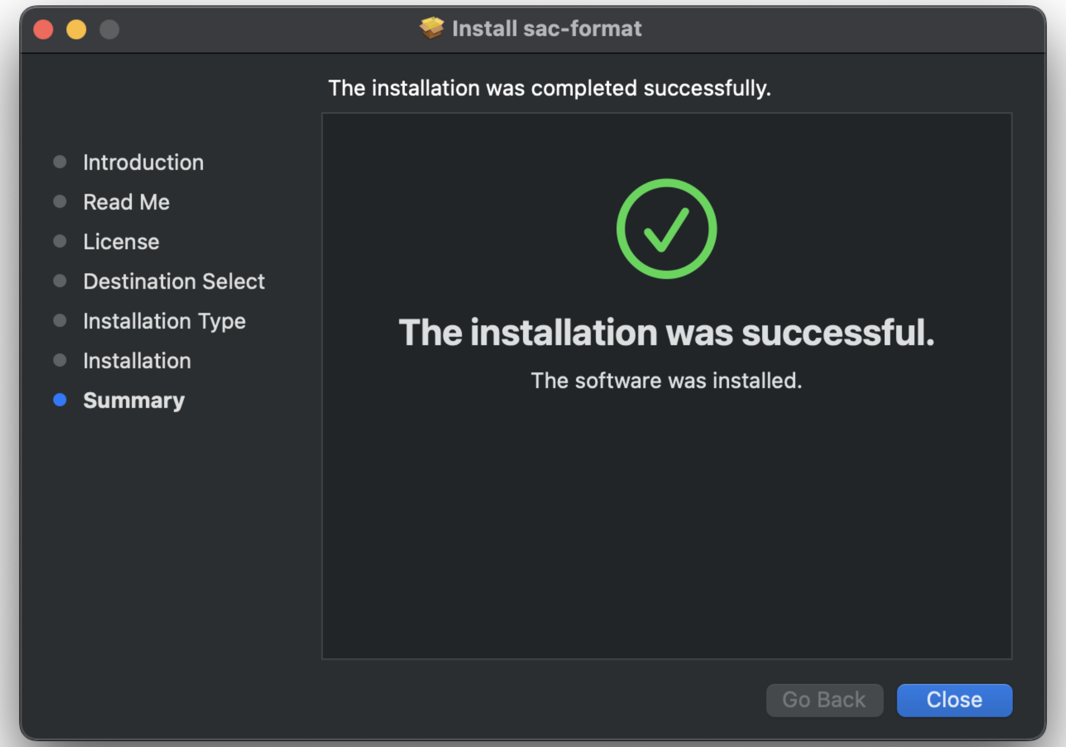
Thank you for trying out sac-format!

If you have any questions, please ask it on the sac-format community forum:

https://github.com/arbCoding/sac-format/discussions/

Figure 2.8 macOS Intro Install

Upon successful installation you will see:



The installation was completed successfully.

The installation was successful.

The software was installed.

Figure 2.9 macOS Install Success

2.2.2 Command line

Command line installation is performed either using the self-extrating archive or by manually extracting the gzipped tar archive.

2.2.2.1 Self-Extracting Archive

```
# Check the sha512 checksum
sha512sum -c sac-format-<version>-Darwin-<arch>.sh.sha512
# Run self-extracting archive
bash sac-format-<version>-Darwin-<arch>.sh
```

Be sure to replace `<version>` and `<arch>` with the correct versions and architectures, respectively (for example: `sac-format-0.4.0-Darwin-x86_64.sh`).

2.2.2.2 Gzipped Tar Archive

```
# Check the sha512 checksum
sha512sum -c sac-format-<version>-Darwin-<arch>.tar.gz.sha512
# Extract Gzipped tar archive
tar -xzf sac-format-<version>-Darwin-<arch>.tar.gz
```

2.3 Linux

sac-format provides four different command line installation methods on Linux.

`Debian` based distributions (for example: Debian, Ubuntu, Linux Mint) can use the Debian Archive.

`RedHat` based distributions (for example: RedHat, Fedora, CentOS) can use the RPM Archive.

All distributions can use the Self-Extracting Archive.

All distributions can use the Gzipped Tar Archive.

2.3.1 Debian Archive

```
# Check the sha512 checksum
sha512sum -c sac-format.deb.sha512
# Install using apt
sudo apt install ./sac-format.deb
```

2.3.2 RPM Archive

```
# Check the sha512 checksum
sha512sum -c sac-format.rpm.sha512
# Install using rpm
sudo rpm -i sac-format.rpm
```

2.3.3 Self-Extrating Archive

```
# Check the sha512 checksum
sha512sum -c sac-format-<version>-Linux-<arch>.sh.sha512
# Run self-extrating archive
bash sac-format-<version>-Linux-<arch>.sh
```

2.3.4 Gzipped Tar Archive

```
# Check the sha512 checksum
sha512sum -c sac-format-<version>-Linux-<arch>.tar.gz.sha512
# Extract gzipped tar archive
tar -xzf sac-format-<version>-Linux-<arch>.tar.gz
```

Chapter 3

Quickstart

This section provides information to incorporate into a project.

To use link to the library (`libsac-format.a` on Linux/macOS, `sac-format.lib` on Windows) and include `sac_format.hpp`.

3.1 Example Programs

3.1.1 list_sac

`list_sac` is a command line program that takes a single SAC-file as its input argument. It reads the SAC-file and outputs the header/footer information, as well as the true size of the `data1` and `data2` vectors.

3.2 CMake Integration

To integrate sac-format into your CMake project, add it to your `CMakeLists.txt`.

```
include(FetchContent)
set(FETCHCONTENT_UPDATES_DISCONNECTED TRUE)
FetchContent_Declare(sac-format
    GIT_REPOSITORY https://github.com/arbCoding/sac-format
    GIT_TAG vX.X.X)
FetchContent_MakeAvailable(sac-format)
include_directory(${sacformat_SOURCE_DIR/src})

project (your_project
    LANGUAGES CXX)

add_executable(your_executable
    your_sources
    sac_format.hpp)

target_link_libraries_library(your_executable
    PRIVATE sac-format)
```

3.3 Example

3.3.1 Reading and Writing

```
#include <sac_format.hpp>
#include <filesystem>
#include <iostream>

using namespace sacfmt;
namespace fs = std::filesystem;

int main() {
    Trace trace1{};
    // Change header variable
    trace1.kstnm("Station1");
    fs::path file{"./test.SAC"};
    // Write
    trace1.write(file);
    // Read
    Trace trace2{file};
    // Confirm equality
    std::cout « (trace1 == trace2) « '\n';
    fs::remove(file);
    return EXIT_SUCCESS;
}
```

Chapter 4

Basic Documentation

This section provides a brief overview of functionality and usage.

4.1 Trace class

The `Trace` class provides easy access to SAC-files in C++. Each SAC-file is a `Trace`; therefore, each `Trace` object is a seismic trace (seismogram).

4.1.1 Reading SAC

SAC-files can be read in by using the parameterized constructor with a `std::filesystem::path` (`<filesystem>`) or a `std::string` (`<string>`) variable that corresponds to the location of the SAC-file.

For example:

```
#include <sac_foramt.hpp>
#include <filesystem>

int main() {
  std::filesystem::path my_file{"/home/user/data/ANMO.SAC"};
  sacfmt::Trace anmo{my_file};
  return EXIT_SUCCESS;
}
```

4.1.2 Writing SAC

Writing SAC files can be done using one of two write functions.

4.1.2.1 v7 files

Use `write` (for example `trace.write(filename)`).

4.1.2.2 v6 files

Use `legacy_write` (for example `trace.legacy_write(filename)`).

4.1.3 Getters and Setters

Every SAC variable is accessed via getters and setters of the same name.

4.1.3.1 Example Getters

- `trace.npts()`
- `trace.data1()`
- `trace.kstnm()`

4.1.3.2 Example Setters

- `trace.kevnm("Event 1")`
- `trace.evla(32.89)`
- `trace.mag(3.21)`

4.1.3.3 Setter rules

Most of the setters are only constrained by the parameter type (single-precision, double-precision, boolean, etc.). **Some** setters are constrained by additional rules.

Required for sanity

Rules here are required because the sac-format library assumes them (not strictly required by the SAC format standard). For instance, the geometric functions assume certain bounds on latitudes and longitudes.

sac-format automatically imposes these rules.

`stla(input)`

Limited to [-90, 90] degrees, input that is outside that range is reduced using circular symmetry.

`stlo(input)`

Limited to [-180, 180] degrees, input that is outside that range is reduced using circular symmetry.

`evla(input)`

Limited to [-90, 90] degrees, input that is outside that range is reduced using circular symmetry.

`evlo(input)`

Limited to [-180, 180] degrees, input that is outside that range is reduced using circular symmetry.

Required for safety

Rules here are required by the SAC format standard. sac-format automatically imposes these rules to prevent the creation of corrupt sac-files.

`npts(input)`

Because `npts` defines the size of the data vectors, changing this value will change the size of `data1` and `data2`$*$. Increasing npts resizes the vectors (`std::vector::resize`) by placing zeros at the **end** of the vectors. Reducing npts resizes the vectors down to the **first npts** values.

Therefore, care must be taken to maintain separate copies of `data1` and `data2`$*$ if you plan to manipulate the original data **after** resizing.

$*$ data2 has `npts` only if it is legal, otherwise it is of size 0.

`leven(input)`

Changing the value of `leven` potentially changes the legality of `data2`, it also potentially affects the value of `iftype`.

If iftype$>$1, then leven must be `true` (evenly sampled data). Therefore, if leven is made `false` in this scenario (unevenly sampled data) then iftype becomes unset$*$.

If changing leven makes data2 legal$**$, then data2 is qresized to have `npts` zeros.

$*$ The SAC format defines the unset values for all data-types. For integers (like iftype) it is the integer value -12345.

$**$ If data2 was already legal, then it is unaffected.

`iftype(input)`

Changing the value of `iftype` poentially changes the legality of `data2`, it also potentially affects the value of `leven`.

If leven is `false`, then iftype must be either 1 or unset. Therefore, changing iftype to have a value $>$1 requires that leven becomes `true` (evenly sampled data).

If changing iftype makes data2 legal$*$, then data2 is resized to have `npts` zeros.

$*$ If data2 was already legal, then it is unaffected.

`data1(input)`

If the size of `data1` is changed, then `npts` must change to reflect the new size. If `data2` is legal, this adjusts its size to match as well.

`data2(input)`

If the size of `data2` is changed to be larger than 0 and it is illegal, it is made legal by setting `iftype(2)` (spectral-data).

When the size of data2 changes, `npts` is updated to the new size and `data1` is resized to match.

If `data2` is made illegal, its size is reduced to 0 while `npts` and `data1` are unaffected.

4.1.4 Convenience Methods

4.1.4.1 calc_geometry

Calculate `gcarc`, `dist`, `az`, and `baz` assuming spherical Earth.

```
trace.stla(45.3);
trace.stlo(34.5);
trace.evla(18.5);
trace.evlo(-34);
trace.calc_geometry();
std::cout « "GcArc: " « trace.gcarc() « '\n';
std::cout « "Dist: " « trace.dist() « '\n';
std::cout « "Azimuth: " « trace.az() « '\n';
std::cout « "BAzimuth: " « trace.baz() « '\n';
```

4.1.4.2 frequency

Calculate frequency from `delta`.

```
double frequency{trace.frequency()};
```

4.1.4.3 date

Return `std::string` formatted as `YYYY-JJJ` from `nzyear` and `nzjday`.

```
std::string date{trace.date()};
```

4.1.4.4 time

Return `std::string` formatted as `HH:MM:SS.xxx` from `nzhour`, `nzmin`, `nzsec`, and `nzmsec`.

```
std::string time{trace.time()};
```

4.1.5 Exceptions

sac-format throws exceptions of type `sacfmt::io_error` (inherits `std::exception`) in the event of a failure to read/write a SAC-file.

4.2 Convenience Functions

4.2.1 degrees_to_radians

Convert decimal degrees to radians.

```
double radians{sacfmt::degrees_to_radians(degrees)};
```

4.2.2 radians_to_degrees

Convert radians to decimal degrees.
```
double degrees{sacfmt::radians_to_degrees(radians)};
```

4.2.3 gcarc

Calculate great-circle arc distance (spherical planet).
```
const point location1{coord{latitude1}, coord{longitude1}};
const point location2{coord{latitude2}, coord{longitude2}};
double gcarc{sacfmt::gcarc(location1, location2)};
```

4.2.4 azimuth

Calculate azimuth between two points (spherical planet).
```
const point location1{coord{latitude1}, coord{longitude1}};
const point location2{coord{latitude2}, coord{longitude2}};
double azimuth{sacfmt::azimuth(location2, location1)};
double back_azimuth{sacfmt::azimuth(location1, location2)};
```

4.2.5 limit_360

Take arbitrary value of degrees and unwrap to [0, 360].
```
double degrees_limited{sacfmt::limit_360(degrees)};
```

4.2.6 limit_180

Take arbitrary value of degrees and unwrap to [-180, 180]. Useful for longitude.
```
double degrees_limited{sacfmt::limit_180(degrees)};
```

4.2.7 limit_90

Take arbitrary value of degrees and unwrap to [-90, 90]. Useful for latitude.
```
double degrees_limited{sacfmt::limit_90(degrees)};
```

4.3 Low-Level I/O

Low-level I/O functions are discussed below.

4.3.1 Binary conversion

4.3.1.1 int_to_binary and binary_to_int

Conversion pair for binary representation of integer values.
```
const int input{10};
// sacfmt::word_one is alias for std::bitset<32> (one word)
sacfmt::word_one binary{sacfmt::int_to_binary(input)};
const int output{sacfmt::binary_to_int(binary)};
std::cout « (input == output) « '\n';
```

4.3.1.2 float_to_binary and binary_to_float

Conversion pair for binary representation of floating-point values.
```
const float input{5F};
sacfmt::word_one binary{sacfmt::float_to_binary(input)};
const float output{sacfmt::binary_to_float(binary)};
std::cout « (input == output) « '\n';
```

4.3.1.3 double_to_binary and binary_to_double

Conversion pair for binary representation of double-precision values.
```
const double input{1e5};
// sacfmt::word_two is alias for std::bitset<64> (two words)
sacfmt::word_two binary{sacfmt::double_to_binary(input)};
const double output{sacfmt::binary_to_double(binary)};
std::cout « (input == output) « '\n';
```

4.3.1.4 string_to_binary and binary_to_string

Conversion pair for binary representation of two-word (regular) string values.
```
const std::string input{"NmlStrng"};
sacfmt::word_two binary{sacfmt::string_to_binary(input)};
const std::string output{sacfmt::binary_to_string(binary)};
std::cout « (input == output) « '\n';
```

4.3.1.5 long_string_to_binary and binary_to_long_string

Conversion pair for binary representation of four-word (only `kstnm` string values.
```
const std::string input{"The Long String"};
// sacfmt::word_four is alias for std::bitset<128> (four words)
sacfmt::word_four binary{sacfmt::long_string_to_binary(input)};
const std::string output{sacfmt::binary_to_long_string(binary)};
std::cout « (input == output) « '\n';
```

4.3.2 Reading/Writing

NOTE that care must be taken when using them to ensure that safe input is provided; the `Trace` class ensures safe I/O, low-level I/O functions do not necessarily ensure safety.

4.3.2.1 read_word, read_two_words, read_four_words, and read_data

Functions to read one-, two-, and four-word variables (depending on the header) and an arbitrary amount of binary data (exclusive to `data1` and `data2`).

4.3.2.2 convert_to_word, convert_to_words, and bool_to_word

Takes objects and converts them into `std::vector<char>` (`convert_to_word` and `bool_to_word`) or `std::array<char, N>` (`convert_to_words`, N = # of words).

4.3.2.3 write_words

Writes input words (as `std::vector<char>`) to a binary SAC-file.

4.3.3 Utility

4.3.3.1 concat_words

Concatenates words taking into account the system endianness.

4.3.3.2 bits_string and string_bits

Template function that performs conversion of binary strings of arbitrary length to an arbitrary number of words.

4.3.3.3 remove_leading_spaces and remove_trailing_spaces

Remove leading and trailing blank spaces from strings assuming ASCII convention (space character is integer 32, below that value are control characters that also appear as blank spaces).

4.3.3.4 string_cleaning

Ensures string does not contain an internal termination character (`\0`) and removes it if present, then removes blank spaces.

4.3.3.5 prep_string

Performs `string_cleaning` followed by string truncation/padding to the necessary length.

4.3.3.6 equal_within_tolerance

Floating-point/double-precision equality within a provided tolerance (default is `f_eps`, defined in `sac_format.↩ hpp`).

4.4 Testing

Unit- and integration-tests (using Catch2) are contained in the `tests` folder. They include:

- `binary_conversions.cpp` confirms that conversion to/from binary functions correctly.
- `constants.cpp` confirms constant values (e.g. SAC magic numbers) are correct.
- `datetime.cpp` confirms date and time functions work correctly.
- `geometry.cpp` confirms that geometric calculations are correct (azimuth, greater-circle arc-length, etc.).
- `trace.cpp` confirms that the trace class is functioning correctly (I/O, exceptions, bounded headers, etc.).

The tests compile to the following programs:

- `basic_tests` (binary conversions and constants).
- `datetime_tests`
- `geometry_tests`
- `trace_tests`

Test coverage details are visible on `CodeCov.io` and `Codacy.com`. All tests can be locally-run to ensure full functionality and compliance.

4.4.1 Errors only

By default each test prints out a pass summary, without details unless an error is encountered.

4.4.2 Full output

By passing the `--success` flag you can see the full results of all tests.

4.4.3 Compact output

The full output is verbose, using the compact reporter will condense the test results (`--reporter=compact`).

4.4.4 Additional options

To see additional options, run `-?`.

4.4.5 Using ctest

If you have CMake install, you can run the tests using `ctest`.

4.5 Benchmarking

`benchmark.cpp` contains the benchmarks. Running it locally will provide information on how long each function takes; benchmarks start with the low-level I/O function and build up to Trace reading, writing, and equality comparison.

To view available optional flags, run `becnhmark -?`.

4.6 Source File List

4.6.1 Core

The two core files are split in the standard interface (hpp)/implementation (cpp) format.

4.6.1.1 sac_format.hpp

Interface: function declarations and constants.

4.6.1.2 sac_format.cpp

Implementation: function details.

4.6.2 Testing and Benchmarking

4.6.2.1 util.hpp

Utility functions and constants exclusive to testing and benchmarking. Not split into interface/implementation.

4.6.2.2 utests.cpp

4.6.2.3 benchmark.cpp

4.6.3 Example programs

4.6.3.1 list_sac.cpp

Chapter 5

SAC-file format

This section provides a centralized description of the SAC file format.

The official and up-to-date documentation for the SAC-file format is available from the EarthScope Consortium (formerly IRIS/UNAVCO) `here`. The following subsections constitute my notes on the format. Below is a quick guide: all credit for the creation of, and documentation for, the SAC file-format belongs to its developers and maintainers (details `here`).

5.1 Floating-point (39)

32-bit (1 word, 4 bytes)

5.1.1 depmin

Minimum value of the dependent variable (displacement/velocity/acceleration/volts/counts).

5.1.2 depmen

Mean value of the dependent variable.

5.1.3 depmax

Maximum value of the dependent variable.

5.1.4 odelta

Modified (*observational*) value of `delta`.

5.1.5 resp(0–9)

Instrument response parameters (poles, zeros, and a constant).

Not used by SAC they're free for other purposes.

5.1.6 stel

Station elevation in meters above sea level (*m.a.s.l*).

Not used by SAC free for other purposes.

5.1.7 stdp

Station depth in meters below surface (borehole/buried vault).

Not used by SAC free for other purposes.

5.1.8 evel

Event elevation *m.a.s.l.*

Not used by SAC free for other purposes.

5.1.9 evdp

Event depth in kilometers (*previously meters*) below surface.

5.1.10 mag

Event magnitude.

5.1.11 user(0–9)

Storage for user-defined values.

5.1.12 dist

Station-Event distance in kilometers.

5.1.13 az

Azimuth $(\text{Event} \rightarrow \text{Station})$, decimal degrees from North.

5.1.14 baz

Back-azimuth $(\text{Station} \rightarrow \text{Event})$, decimal degrees from North.

5.1.15 gcarc

Station-Event great circle arc-length, decimal degrees.

5.1.16 cmpaz

Instrument measurement azimuth, decimal degrees from North.

Value	Direction
0 °	North
90 °	East
180 °	South
270 °	West
Other	1/2/3

5.1.17 cmpinc

Instrument measurement incident angle, decimal degrees from upward vertical (incident 0 ° = dip -90 °).

Value	Direction
0 °	Up
90 °	Horizontal
180 °	Down
270 °	Horizontal

NOTE: SEED/MINISEED use dip angle, decimal degrees down from horizontal (dip 0 ° = incident 90 °).

5.1.18 xminimum

Spectral-only equivalent of `depmin` (f_0 or ω_0).

5.1.19 xmaximum

Spectral-only equivalent of `depmax` (f_{max} or ω_{max}).

5.1.20 yminimum

Spectral-only equivalent of `b`.

5.1.21 ymaximum

Spectral-only equivalent of `e`.

5.2 Double (22)

64-bit (2 words, 8 bytes)

NOTE: in the header section these are floats; they're doubles in the footer section of `v7` SAC-files. In memory they're stored as doubles regardless of the SAC-file version.

5.2.1 delta

Increment between evenly spaced samples (Δt for timeseries, Δf or $\Delta \omega$ for spectra).

5.2.2 b

First value (*begin*) of independent variable (t_0).

5.2.3 e

Final value (*end*) of independent variable (t_{max}).

5.2.4 o

Event *origin* time, in seconds relative to the reference time.

5.2.5 a

Event first *arrival* time, in seconds relative to the reference time.

5.2.6 t(0–9)

User defined *time* values, in seconds relative to the reference time.

5.2.7 f

Event end (*fini*) time, in seconds relative to the reference time.

5.2.8 stla

Station latitude in decimal degrees, N/S - positive/negative.

sac-format automatically enforces $\mathrm{stla} \in [-90, 90]$.

5.2.9 stlo

Station longitude in decimal degrees, E/W - positive/negative.

sac-format automatically enforces $\mathrm{stlo} \in [-180, 180]$.

5.2.10 evla

Event latitude in decimal degrees, N/S - positive/negative.

sac-format automatically enforces $\mathrm{evla} \in [-90, 90]$.

5.2.11 evlo

Event longitude in decimal degrees, E/W - positive/negative.

sac-format automatically enforces $\mathrm{evlo} \in [-180, 180]$.

5.2.12 sb

Original (*saved*) `b` value.

5.2.13 sdelta

Original (*saved*) `delta` value.

5.3 Integer (26)

32-bit (1 word, 4 bytes)

5.3.1 nzyear

Reference time GMT year.

5.3.2 nzjday

Reference time GMT day-of-year (often called `Julian Date`) (1–366).

5.3.3 nzhour

Reference time GMT hour (0–23).

5.3.4 nzmin

Reference time GMT minute (0–59).

5.3.5 nzsec

Reference time GMT second (0–59).

5.3.6 nzmsec

Reference time GMT Millisecond (0–999).

5.3.7 nvhdr

SAC-file version.

Version	Description
`v7`	Footer (2020+, sac 102.0+)
`v6`	No footer (pre-2020, sac 101.6a-)

5.3.8 norid

Origin ID.

5.3.9 nevid

Event ID.

5.3.10 npts

Number of points in data.

5.3.11 nsnpts

Original (*saved*) `npts`.

5.3.12 nwfid

Waveform ID.

5.3.13 nxsize

Spectral-only equivalent of `npts` (length of spectrum).

5.3.14 nysize

Spectral-only, width of spectrum.

5.3.15 iftype

File type.

Value	Type	Description
01	ITIME	Time-series
02	IRLIM	Spectral (real/imaginary)
03	IAMPH	Spectral (amplitude/phase)
04	IXY	General XY file
??	IXYZ$*$	General XYZ file

$*$Value not listed in the standard.

5.3.16 idep

Dependent variable type.

Value	Type	Description
05	IUNKN	Unknown
06	IDISP	Displacement (nm)
07	IVEL	Velocity $\left(\frac{\mathrm{nm}}{\mathrm{s}}\right)$
08	IACC	Acceleration $\left(\frac{\mathrm{nm}}{\mathrm{s}^2}\right)$
50	IVOLTS	Velocity (volts)

5.3.17 iztype

Reference time equivalent.

Value	Type	Description
05	IUNKN	Unknown
09	IB	Recording start time
10	IDAY	Midnight reference GMT day
11	IO	Event origin time
12	IA	First arrival time
13-22	IT(0-9)	User defined time (t) pick

5.3.18 iinst

Recording instrument type.

Not used by SAC: free for other purposes.

5.3.19 istreg

Station geographic region.

Not used by SAC: free for other purposes.

5.3.20 ievreg

Event geographic region.

Not used by SAC: free for other purposes.

5.3.21 ievtyp

Event type.

Value	Type	Description
05	IUNKN	Unknown
11	IO	Other source of known origin
37	INUCL	Nuclear
38	IPREN	Nuclear pre-shot
39	IPOSTN	Nuclear post-shot
40	IQUAKE	Earthquake
41	IPREQ	Foreshock
42	IPOSTQ	Aftershock
43	ICHEM	Chemical explosion
44	IOTHER	Other
72	IQB	Quarry/mine blast: confirmed by quarry/mine
73	IQB1	Quarry/mine blast: designed shot info-ripple fired

Value	Type	Description
74	IQB2	Quarry/mine blast: observed shot info-ripple fired
75	IQBX	Quarry/mine blast: single shot
76	IQMT	Quarry/mining induced events: tremor and rockbursts
77	IEQ	Earthquake
78	IEQ1	Earthquake in a swarm or in an aftershock sequence
79	IEQ2	Felt earthquake
80	IME	Marine explosion
81	IEX	Other explosion
82	INU	Nuclear explosion
83	INC	Nuclear cavity collapse
85	IL	Local event of unknown origin
86	IR	Region event of unknown origin
87	IT	Teleseismic event of unknown origin
88	IU	Undetermined/conflicting information

5.3.22 iqual

Quality of data.

Value	Type	Description
44	IOTHER	Other
45	IGOOD	Good
46	IGLCH	Glitches
47	IDROP	Dropouts
48	ILOWSN	Low signal-to-noise ratio

Not used by SAC: free for other purposes.

5.3.23 isynth

Synthetic data flag.

Value	Type	Description
49	IRLDATA	Real data
XX	$*$	Synthetic

$*$Values and types not listed in the standard.

5.3.24 imagtyp

Magnitude type.

Value	Type	Description
52	IMB	Body-wave magnitude (M_b)

Value	Type	Description
53	IMS	Surface-wave magnitude (M_s)
54	IML	Local magnitude (M_l)
55	IMW	Moment magnitude (M_w)
56	IMD	Duration magnitude (M_d)
57	IMX	User-defined magnitude (M_x)

5.3.25 imagsrc

Source of magnitude information.

Value	Type	Description
58	INEIC	National Earthquake Information Center
61	IPDE	Preliminary Determination of Epicenter
62	IISC	Internation Seismological Centre
63	IREB	Reviewed Event Bulletin
64	IUSGS	U.S. Geological Survey
65	IBRK	UC Berkeley
66	ICALTECH	California Institute of Technology
67	ILLNL	Lawrence Livermore National Laboratory
68	IEVLOC	Event location (computer program)
69	IJSOP	Joint Seismic Observation Program
70	IUSER	The user
71	IUNKNOWN	Unknown

5.3.26 ibody

Body/spheroid definition used to calculate distances.

Value	Type	Name	Semi-major axis (a [m])	Inverse Flattening (*f*)
-12345	UNDEF	Earth (*Historic*)	6378160.0	0.00335293
98	ISUN	Sun	696000000.0	8.189e-6
99	IMERCURY	Mercury	2439700.0	0.0
100	IVENUS	Venus	6051800.0	0.0
101	IEARTH	Earth (*WGS84*)	6378137.0	0.0033528106647474805
102	IMOON	Moon	1737400.0	0.0
103	IMARS	Mars	3396190.0	0.005886007555525457

5.4 Boolean (4)

32-bit (1 word, 4 bytes) in-file/8-bit (1 byte) in-memory

5.4.1 leven

REQUIRED Evenly-spaced data flag.

If true, then data is evenly spaced.

5.4.2 lpspol

Station polarity flag.

If true, then station has positive-polarity; it follows the left-hand convention (for example, North-East-Up [NEZ]).

5.4.3 lovrok

File overwrite flag.

If true, then it's okay to overwrite the file.

5.4.4 lcalda

Calculate geometry flag.

If true, then calculate `dist`, `az`, `baz`, and `gcarc` from `stla`, `stlo`, `evla`, and `evlo`.

5.5 String (23)

32/64-bit (2/4 words, 8/16 bytes, 8/16 characters)

5.5.1 kstnm

Station name.

5.5.2 kevnm

Event name.

$*$This is the **only** four word (16 character) string.

5.5.3 khole

Nuclear: Hole identifier.

Other: Location identifier (LOCID).

5.5.4 ko

Text for `o`.

5.5.5 ka

Text for `a`.

5.5.6 kt(0–9)

Text for `t(0--9)`.

5.5.7 kf

Text for `f`.

5.5.8 kuser(0–2)

Text for the first three of `user(0--9)`.

5.5.9 kdatrd

Date the data was read onto a computer.

5.5.10 kinst

Text for `iinst`.

5.6 Data (2)

32-bit (2 words, 8 bytes) in-file/64-bit (4 words, 16 bytes) in-memory

Stored as floating-point (32-bit) values in SAC-files; stored as double-precision in memory.

5.6.1 data1

The first data vector—**always** present in a SAC-file and begins at word 158.

5.6.2 data2

The second data vector—**conditionally** present and begins after `data1`.

Required if `leven` is false, or if `iftype` is spectral/XY/XYZ.

Chapter 6

Build Instructions

This section provides instructions to build from source.

6.1 Dependencies

6.1.1 Automatic (CMake)

`Xoshiro-cpp v1.12.0` (testing and benchmarking).

6.1.2 Manual

`Catch2 v3.4.0` (testing and benchmarking). Note that this is automatic on Windows (not Linux nor macOS).

6.1.2.1 macOS and Linux

```
git clone https://github.com/catchorg/Catch2.git
cd Catch2
git checkout v3.5.2
cmake -Bbuild -S. -DBUILD_TESTING=OFF
sudo cmake --build ./build/ --target install
```

6.2 Building

Building is as easy as cloning the repository, running CMake for your preferred build tool, and then building.

6.2.1 GCC

```
git clone https://github.com/arbCoding/sac-format.git
cd sac-format
cmake --preset gcc-hard-release
cmake --build ./build/hard/release/gcc
```

6.2.2 Clang

```
git clone https://github.com/arbCoding/sac-format.git
cd sac-format
cmake --preset clang-hard-release
cmake --build ./build/hard/release/clang
```

6.2.3 MSVC

```
git clone https://github.com/arbCoding/sac-format.git
cd sac-format
cmake -B ./build -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_STANDARD=20 `
-DCMAKE_CXX_STANDARD_REQUIRED=ON -DCMAKE_CXX_EXTENSIONS=OFF `
-DCMAKE_CXX_FLAGS="/O2 /EHsc /Gs /guard:cf"
```

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

Chapter 9

Class Index

9.1 Class List

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

Chapter 10

Namespace Documentation

10.1 sacfmt Namespace Reference

sac-format namespace

Namespaces

- namespace bitset_type

  bitset type-safety namespace.

Classes

- class coord

  Defines a geographic coordinant (degrees/radians)
- class io_error

  Class for generic I/O exceptions.
- struct point

  Defines a geographic point (latitude, longitude)
- struct read_spec

  Struct that specifies parameters for reading.
- class Trace

  The Trace class.
- struct word_pair

  Struct containing a pair of words.

Typedefs

- using char_bit = std::bitset< bits_per_byte >

  One binary character (useful for building strings).
- using word_one = std::bitset< binary_word_size >

  One binary word (useful for non-strings).
- using word_two = std::bitset< static_cast< size_t >(2) ∗binary_word_size >

  Two binary words (useful for strings).
- using word_four = std::bitset< static_cast< size_t >(4) ∗binary_word_size >

  Four binary words (kEvNm only).
- template<class T >
  using unsigned_int = typename bitset_type::uint< sizeof(T) ∗bits_per_byte >::type

  Convert variable to unsigned-integer using type-safe conversions.

Enumerations

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

  Enumeration of all SAC fields.

Functions

- std::streamoff word_position (const size_t word_number) noexcept

  Calculates position of word in SAC-file.
- word_one uint_to_binary (uint num) noexcept

  Convert unsigned integer to 32-bit (one word) binary bitset.
- word_one int_to_binary (int num) noexcept

  Convert integer to 32-bit (one word) binary bitset.
- int binary_to_int (word_one bin) noexcept

  Convert 32-bit (one word) binary bitset to integer.
- word_one float_to_binary (const float num) noexcept

  Convert floating-point value to 32-bit (one word) binary bitset.
- float binary_to_float (const word_one &bin) noexcept

  Convert 32-bit (one word) binary bitset to a floating-point value.
- word_two double_to_binary (const double num) noexcept

  Convert double-precision value to 64-bit (two words) binary bitset.
- double binary_to_double (const word_two &bin) noexcept

  Convert 64-bit (two words) binary bitset to double-precision value.
- void remove_leading_spaces (std::string ∗str) noexcept

  Remove all leading spaces from a string.

- void remove_trailing_spaces (std::string ∗str) noexcept

  Remove all trailing spaces from a string.
- std::string string_cleaning (const std::string &str) noexcept

  Remove leading/trailing spaces and control characters from a string.
- void prep_string (std::string ∗str, const size_t str_size) noexcept

  Cleans string and then truncates/pads as necessary.
- template< typename T >
  void string_bits (T ∗bits, const std::string &str, const size_t str_size) noexcept

  Template function to convert string into binary bitset.
- template< typename T >
  std::string bits_string (const T &bits, const size_t num_words) noexcept

  Template function to convert binary bitset to string.
- word_two string_to_binary (std::string str) noexcept

  Convert string to a 64-bit (two word) binary bitset.
- std::string binary_to_string (const word_two &str) noexcept

  Convert a 64-bit (two word) binary bitset to a string.
- word_four long_string_to_binary (std::string str) noexcept

  Convert a string to a 128-bit (four word) binary bitset.
- std::string binary_to_long_string (const word_four &str) noexcept

  Convert a 128-bit (four word) binary bitset to a string.
- word_one bool_to_binary (const bool flag) noexcept

  Convert a boolean to a 32-bit (one word) binary bitset.
- bool binary_to_bool (const word_one &flag) noexcept

  Convert a 32-bit (one word) binary bitset to a boolean.
- word_two concat_words (const word_pair< word_one > &pair_words) noexcept

  Concatenate two word_one binary strings into a single word_two string.
- word_four concat_words (const word_pair< word_two > &pair_words) noexcept

  Concatenate two word_two binary strings into a single word_four string.
- bool nwords_after_current (std::ifstream ∗sac, const read_spec &spec) noexcept

  Determine if the SAC-file has enough remaining data to read the requested amount of data.
- void safe_to_read_header (std::ifstream ∗sac)

  Determine if the SAC-file is large enough to contain a complete header.
- void safe_to_read_footer (std::ifstream ∗sac)

  Determines if the SAC-file has enough space remaining to contain a complete footer.
- void safe_to_read_data (std::ifstream ∗sac, const size_t n_words, const bool data2)

  Determines if the SAC-file has enough space remaining to contain a complete data vector.
- void safe_to_finish_reading (std::ifstream ∗sac)

  Determines if the SAC-file is finished.
- word_one read_word (std::ifstream ∗sac)

  Read one word (32 bits, useful for non-strings) from a binary SAC-File.
- word_two read_two_words (std::ifstream ∗sac)

  Read two words (64 bits, useful for most strings) from a binary SAC-file.
- word_four read_four_words (std::ifstream ∗sac)

  Read four words (128 bits, kEvNm only) from a binary SAC-file.
- std::vector< double > read_data (std::ifstream ∗sac, const read_spec &spec)

  Reader arbitrary number of words (useful for vectors) from a binary SAC-file.
- void write_words (std::ofstream ∗sac_file, const std::vector< char > &input)

  Write arbitrary number of words (useful for vectors) to a binary SAC-file.
- template< typename T >
  std::vector< char > convert_to_word (const T input) noexcept

  Template function to convert input value into a std::vector<char> for writing.

- std::vector< char > convert_to_word (const double input) noexcept

    Convert double value into a std::vector<char> for writing.
- template<size_t N>
  std::array< char, N > convert_to_words (const std::string &str, int n_words) noexcept

    Template function to convert input string value into a std::array<char> for writing.
- std::vector< char > bool_to_word (const bool flag) noexcept

    Convert boolean to a word for writing.
- bool equal_within_tolerance (const std::vector< double > &vector1, const std::vector< double > &vector2, const double tolerance) noexcept

    Check if two std::vector<double> are equal within a tolerance limit.
- bool equal_within_tolerance (const double val1, const double val2, const double tolerance) noexcept

    Check if two double values are equal within a tolerance limit.
- double degrees_to_radians (const double degrees) noexcept

    Convert decimal degrees to radians.
- double radians_to_degrees (const double radians) noexcept

    Convert radians to decimal degrees.
- double gcarc (const point location1, const point location2) noexcept

    Calculate great circle arc distance in decimal degrees between two points.
- double azimuth (const point location1, const point location2) noexcept

    Calculate azimuth between two points.
- double limit_360 (const double degrees) noexcept

    Takes a decimal degree value and constrains it to full circle using symmetry.
- double limit_180 (const double degrees) noexcept

    Takes a decimal degree value and constrains it to a half circle using symmetry.
- double limit_90 (const double degrees) noexcept

    Takes a decimal degree value and constrains it to a quarter circle using symmetry.
- template std::vector< char > convert_to_word (const float input) noexcept
- template std::vector< char > convert_to_word (const int x) noexcept
- template std::array< char, word_length > convert_to_words (const std::string &str, const int n_words) noexcept

Variables

- constexpr size_t word_length {4}

    Size (bytes) of fundamental data-chunk.
- constexpr size_t bits_per_byte {8}

    Size (bits) of binary character.
- constexpr size_t binary_word_size {word_length * bits_per_byte}

    Size (bits) of funamental data-chunk.
- constexpr std::streamoff data_word {158}

    First word of (first) data-section (stream offset).
- constexpr int unset_int {-12345}

    Integer unset value (SAC Magic).
- constexpr float unset_float {-12345.0F}

    Float-point unset value (SAC Magic).
- constexpr double unset_double {-12345.0}

    Double-precision unset value (SAC Magic).
- constexpr bool unset_bool {false}

    Boolean unset value (SAC Magic).
- const std::string unset_word {"-12345"}

    String unset value (SAC Magic).

- constexpr float f_eps {2.75e-6F}

  Accuracy precision expected of SAC floating-point values.
- constexpr int ascii_space {32}

  ASCII-code of 'space' character.
- constexpr int num_float {39}

  Number of float-poing header values in SAC format.
- constexpr int num_double {22}

  Number of double-precision header values in SAC format.
- constexpr int num_int {26}

  Number of integer header values in SAC format.
- constexpr int num_bool {4}

  Number of boolean header values in SAC format.
- constexpr int num_string {23}

  Number of string header values in SAC format.
- constexpr int num_data {2}

  Number of data arrays in SAC format.
- constexpr int num_footer {22}

  Number of double-precision footer values in SAC format (version 7).
- constexpr int modern_hdr_version {7}

  nVHdr value for newest SAC format (2020+).
- constexpr int old_hdr_version {6}

  nVHdr value for historic SAC format (pre-2020).
- constexpr int common_skip_num {7}

  Extremely common number of 'internal use' headers in SAC format.
- constexpr double rad_per_deg {std::numbers::pi_v< double > / 180.0}

  Radians per degree.
- constexpr double deg_per_rad {1.0 / rad_per_deg}

  Degrees per radian.
- constexpr double circle_deg {360.0}

  Degrees in a circle.
- constexpr double earth_radius {6378.14}

  Average radius of Earth (kilometers).
- const std::unordered_map< name, const size_t > sac_map

  Lookup table for variable locations.

10.1.1 Detailed Description

sac-format namespace

10.1.2 Typedef Documentation

10.1.2.1 char_bit

```
using sacfmt::char_bit = typedef std::bitset<bits_per_byte>
```

One binary character (useful for building strings).

10.1.2.2 unsigned_int

```
template<class T >
using sacfmt::unsigned_int = typedef typename bitset_type::uint<sizeof(T) * bits_per_byte>↩
::type
```

Convert variable to unsigned-integer using type-safe conversions.

10.1.2.3 word_four

```
using sacfmt::word_four = typedef std::bitset<static_cast<size_t>(4) * binary_word_size>
```

Four binary words (kEvNm only).

10.1.2.4 word_one

```
using sacfmt::word_one = typedef std::bitset<binary_word_size>
```

One binary word (useful for non-strings).

10.1.2.5 word_two

```
using sacfmt::word_two = typedef std::bitset<static_cast<size_t>(2) * binary_word_size>
```

Two binary words (useful for strings).

10.1.3 Enumeration Type Documentation

10.1.3.1 name

```
enum class sacfmt::name  [strong]
```

Enumeration of all SAC fields.

Additional information can be found at SAC-file format

Enumerator

depmin	Float Minimum value of the dependent variable (displacement/velocity/acceleration/volts/counts).
depmax	Float Maximum value of the dependent variable.
odelta	Float Modified (observational) value of delta.
resp0	Float Instrument response parameter (poles, zeros, and a constant). Not used by SAC - free for other purposes.
resp1	See resp0.
resp2	See resp0.

Enumerator

resp3	See resp0.
resp4	See resp0.
resp5	See resp0.
resp6	See resp0.
resp7	See resp0.
resp8	See resp0.
resp9	See resp0.
stel	Float Station elevation in meters above sea level (m.a.s.l.). Not used by SAC - free for other purposes.
stdp	Float Station depth in meters below surface (borehole/buried vault). Not used by SAC - free for other purposes.
evel	Float Event elevation m.a.s.l. Not used by SAC - free for other purposes.
evdp	Float Event depth in kilometers (previous meters) below surface.
mag	Float Event magnitude.
user0	Float Storage for user-defined values.
user1	See user0.
user2	See user0.
user3	See user0.
user4	See user0.
user5	See user0.
user6	See user0.
user7	See user0.
user8	See user0.
user9	See user0.
dist	Float Station-Event distance in kilometers.
az	Float Azimuth $Station \rightarrow Event$ in decimal degrees from North.
baz	Float Back-Azimuth $Event \rightarrow Station$ in decimal degrees from North.
gcarc	Float Great-circle arc-distance between station and event in decimal degrees.
depmen	Float Mean value of dependent variable.
cmpaz	Float Instrument measurement azimuth, decimal degrees from North.
cmpinc	Float Instrument measurement incidence angle, decimal degrees from upward vertical (incident 0 = dip -90). Note: SEED/MINISEED use dip angle, decimal degrees from horizontal (dip 0 = incident 90).
xminimum	Float Spectral-only equivalent of depmin (f_0 or ω_0).
xmaximum	Float Spectral-only equivalent of depman (f_{max} or ω_{max}).

Enumerator

yminimum	Float Spectral-only equivalent of b.
ymaximum	Float Spectral-only equivalent of e.
delta	Double Increment between evenly-spaced samples (Δt for timeseries, Δf or $\Delta\omega$ for spectral).
b	Double First value (beginning) of independent variable (t_0).
e	Double Final value (ending) of the independent variable (t_{max}).
o	Double Event origin time, in seconds relative to the reference time.
a	Double Event first arrival time, in seconds relative to the reference time.
t0	Double User defined time value, in seconds relative to the reference time.
t1	See t0.
t2	See t0.
t3	See t0.
t4	See t0.
t5	See t0.
t6	See t0.
t7	See t0.
t8	See t0.
t9	See t0.
f	Double Event end (fini) time, in seconds relative to the reference time.
stla	Double Station latitude in decimal degrees, N/S is positive/negative. sac-format automatically enforces $\phi \in [-90, 90]$.
stlo	Double Station longitude in decimal degrees, E/W is positive/negative. sac-format automaticall enforces $\lambda \in [-180, 180]$.
evla	Double Event latitude in decimal degrees, N/S is positive/negative. sac-format automatically enforces $\phi \in [-90, 90]$.
evlo	Double Event longitude in decimal degrees, E/W is positive/negative. sac-format automatically enforces $\lambda \in [-180, 180]$.
sb	Double Original (saved) value of b (beginning).
sdelta	Double Original (saved) value of delta (sample-spacing).
nzyear	Integer Reference time GMT year.
nzjday	Integer Reference time GMT day-of-year (often called Julian Date). 1-366 Not enforced.
nzhour	Integer Reference time GMT hour. 00-23 Not enforced.

Enumerator

nzmin	Integer Reference time GMT minute. 00-59 Not enforced.
nzsec	Integer Reference time GMT second. 00-59 Not enforced.
nzmsec	Integer Reference time GMT millisecond. 0-999 not enforced.
nvhdr	Integer SAC-file version. 7 = 2020+, sac 102.0+, has a Footer. 6 = pre-2020, sac 101.6a-, no Footer.
norid	Integer Origin ID.
nevid	Integer Event ID.
npts	Integer Number of points in data.
nsnpts	Integer Original (saved) npts.
nwfid	Integer Waveform ID.
nxsize	Integer Spectral-only equivalent of npts (length of spectrum).
nysize	Integer Spectral-only; width of spectrum.
iftype	Integer File type.
idep	Integer Dependent variable type.
iztype	Integer Reference time equivalent.
iinst	Integer Recording instrument type. Not used by SAC - free for other purposes.
istreg	Integer Station geographic region. Not used by SAC - free for other purposes.
ievreg	Integer Event geographic region. Not used by SAC - free for other purposes.
ievtyp	Integer Event type. Not used by SAC - free for other purposes.
iqual	Integer Quality of data. Not used by SAC - free for other purposes.
isynth	Integer Synthetic data flag. Not used by SAC - free for other purposes.
imagtyp	Integer Magnitude type.

Enumerator

imagsrc	Integer Magnitude information source.
ibody	Integer Body/spheroid definition used to calculate distances. Not currently-used by sac-format (SAC does used it).
leven	Boolean REQUIRED Evenly-spaced data flag. True = even.
lpspol	Boolean Station polarity flag. True = positive (left-handed, e.g. North-East-Up).
lovrok	Boolean File overwrite flag. If true, okay to overwrite file. Not used by sac-format.
lcalda	Boolean Calculate geometry flag. Not used by sac-format.
kstnm	String (2 words) Station name.
kevnm	String (4 words) Event name.
khole	String (2 words) Nuclear-Hole identifier. Other-Location identifier (LOCID).
ko	String (2 words) Text for o.
ka	String (2 words) Text for a.
kt0	String (2 words) Text for t0
kt1	See kt0.
kt2	See kt0.
kt3	See kt0.
kt4	See kt0.
kt5	See kt0.
kt6	See kt0.
kt7	See kt0.
kt8	See kt0.
kt9	See kt0.
kf	String (2 words) Text for f.
kuser0	String (2 words) Text for user0.
kuser1	See kuser0.
kuser2	See kuser0.
kcmpnm	String (2 words) Component name.
knetwk	String (2 words) Network name.
kdatrd	String (2 words) Date the data was read onto a computer.

Enumerator

kinst	String (2 words) Instrument name.
data1	std::vector<double> First data vector. ALWAYS present, ALWAYS begins at word 158.
data2	std::vector<double> Second data vector. CONDITIONAL present. IF PRESENT, begins at end of data1. Required if leven is false (uneven sampling), or if iftype is spectral/XY/XYZ.

```
             {
// Floats
depmin,
depmax,
odelta,
resp0,
resp1,
resp2,
resp3,
resp4,
resp5,
resp6,
resp7,
resp8,
resp9,
stel,
stdp,
evel,
evdp,
mag,
user0,
user1,
user2,
user3,
user4,
user5,
user6,
user7,
user8,
user9,
dist,
az,
baz,
gcarc,
depmen,
cmpaz,
cmpinc,
xminimum,
xmaximum,
yminimum,
ymaximum,
// Doubles
delta,
b,
e,
o,
a,
t0,
t1,
t2,
t3,
t4,
t5,
t6,
t7,
t8,
t9,
f,
stla,
stlo,
evla,
evlo,
sb,
sdelta,
// Ints
nzyear,
nzjday,
nzhour,
nzmin,
nzsec,
nzmsec,
nvhdr,
  norid,
  nevid,
  npts,
  nsnpts,
  nwfid,
  nxsize,
  nysize,
  iftype,
  idep,
  iztype,
  iinst,
  istreg,
  ievreg,
  ievtyp,
  iqual,
  isynth,
  imagtyp,
  imagsrc,
  ibody,
  // Bools
  leven,
  lpspol,
  lovrok,
  lcalda,
  // Strings
  kstnm,
  kevnm,
  khole,
  ko,
  ka,
  kt0,
  kt1,
  kt2,
  kt3,
  kt4,
  kt5,
  kt6,
  kt7,
  kt8,
  kt9,
  kf,
  kuser0,
  kuser1,
  kuser2,
  kcmpnm,  // missing in org documentation
  knetwk,  // missing in org documentation
  kdatrd,
  kinst,
  // Data
  data1,
  data2
};
```

10.1.4 Function Documentation

10.1.4.1 azimuth()

```
double sacfmt::azimuth (
              const point location1,
              const point location2 )  [noexcept]
```

Calculate azimuth between two points.

Assumes spherical Earth (in future may update to solve on a more general body).

ϕ is latitude. λ is longitude. θ is azimuth.

$$\theta = tan^{-1}\left(\frac{sin(\delta\lambda)cos(\phi_2)}{cos(\phi_1)sin(\phi_2) - sin(\phi_1)cos(\phi_2)cos(\delta\lambda)}\right)$$

Parameters

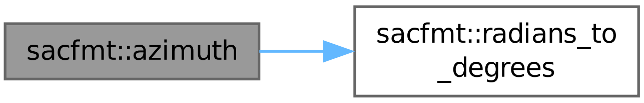
in	*location1*	point of first location.
in	*location2*	point of second location.

Returns

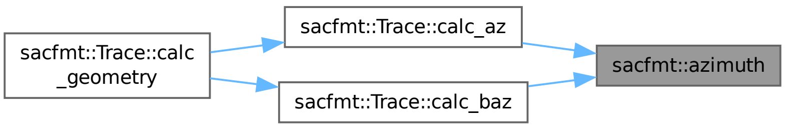
double The azimuth from the first location to the second location.

```
                                                                                              {
  const double numerator{
      std::sin(location2.longitude.radians() - location1.longitude.radians()) *
      std::cos(location2.latitude.radians())};
  const double denominator{(std::cos(location1.latitude.radians()) *
                             std::sin(location2.latitude.radians())) -
                            (std::sin(location1.latitude.radians()) *
                             std::cos(location2.latitude.radians()) *
                             std::cos(location2.longitude.radians() -
                                        location1.longitude.radians()))};
  double result{radians_to_degrees(std::atan2(numerator, denominator))};
  while (result < 0.0) {
    result += circle_deg;
  }
  return result;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.2 binary_to_bool()

```
bool sacfmt::binary_to_bool (
              const word_one & flag )   [noexcept]
```

Convert a 32-bit (one word) binary bitset to a boolean.

Parameters

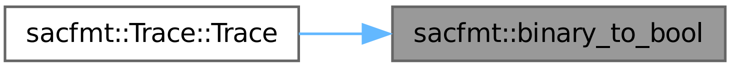
in	*flag*	word_one binary bitset to be converted (takes zeroth element).

Returns

boolean Converted boolean value.

```
{ return flag[0]; }
```

Here is the caller graph for this function:



10.1.4.3 binary_to_double()

```
double sacfmt::binary_to_double (
            const word_two & bin )  [noexcept]
```

Convert 64-bit (two words) binary bitset to double-precision value.

Converts bitset to unsigned long long then to double.

Parameters

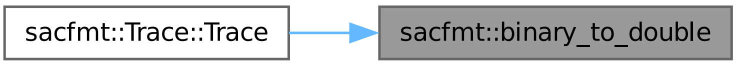
in	*bin*	word_two Binary value to be converted.

Returns

double Converted value.

```
                                                  {
  const auto val = bin.to_ullong();
  double result{};
  // flawfinder: ignore
  memcpy(&result, &val, sizeof(double));
  return result;
}
```

Here is the caller graph for this function:



10.1.4.4 binary_to_float()

```
float sacfmt::binary_to_float (
            const word_one & bin )  [noexcept]
```

Convert 32-bit (one word) binary bitset to a floating-point value.

Converts bitset to unsigned long then to float.

Parameters

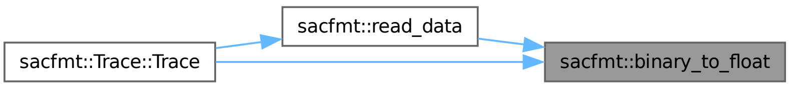
in	*bin*	word_one Binary value to be converted.

Returns

float Converted value.

```
                                                           {
  const auto val = bin.to_ulong();
  float result{};
  // flawfinder: ignore
  memcpy(&result, &val, sizeof(float));
  return result;
}
```

Here is the caller graph for this function:



10.1.4.5 binary_to_int()

```
int sacfmt::binary_to_int (
            word_one bin )  [noexcept]
```

Convert 32-bit (one word) binary bitset to integer.

Uses two's complement to convert a binary value into an integer.

Parameters

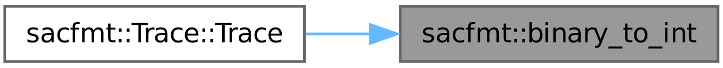
in	*bin*	Binary value to be converted.

Returns

int Converted value.

```
                                        {
  int result{};
  if (bin.test(binary_word_size - 1)) {
    // Complement
    bin.flip();
    result = static_cast<int>(bin.to_ulong());
    result += 1;
    // Change sign to make it negative
    result *= -1;
  } else {
    result = static_cast<int>(bin.to_ulong());
  }
  return result;
}
```

Here is the caller graph for this function:



10.1.4.6 binary_to_long_string()

```
std::string sacfmt::binary_to_long_string (
            const word_four & str ) [noexcept]
```

Convert a 128-bit (four word) binary bitset to a string.

Exclusively used to work with the kEvNm header.

Parameters

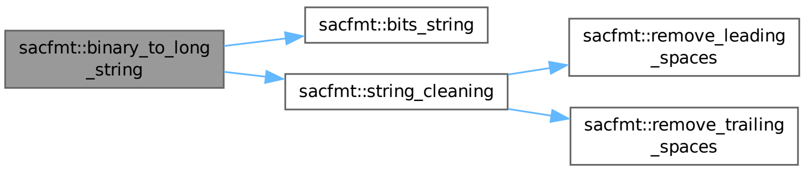
in	*str*	word_four to be converted to a string.

Returns

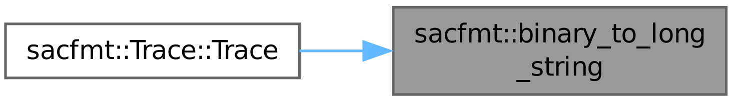
std::string Converted string.

```
                                                            {
  std::string result{bits_string(str, 4)};
  return string_cleaning(result);
}
```

Here is the call graph for this function:


sacfmt::binary_to_long _string
sacfmt::bits_string
sacfmt::string_cleaning
sacfmt::remove_leading _spaces
sacfmt::remove_trailing _spaces

Here is the caller graph for this function:



10.1.4.7 binary_to_string()

```
std::string sacfmt::binary_to_string (
            const word_two & str )  [noexcept]
```

Convert a 64-bit (two word) binary bitset to a string.

Parameters

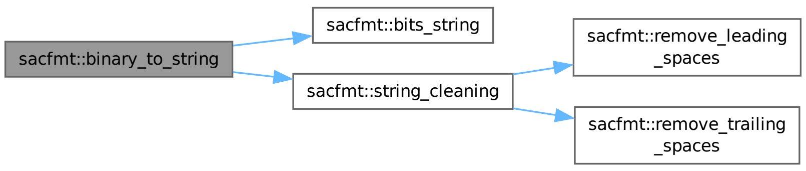
in	*str*	word_two to be converted to a string.

Returns

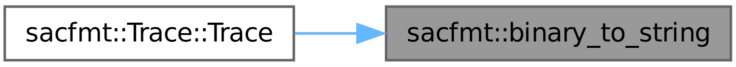
std::string Converted string.

```
                                                               {
      std::string result{bits_string(str, 2)};
      return string_cleaning(result);
  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.8 bits_string()

```
template<typename T >
std::string sacfmt::bits_string (
            const T & bits,
            const size_t num_words )  [noexcept]
```

Template function to convert binary bitset to string.

Parameters

in	*bits*	Source bitset for the string.
in	*num_words*	Length of string in words (4 chars = 1 word)

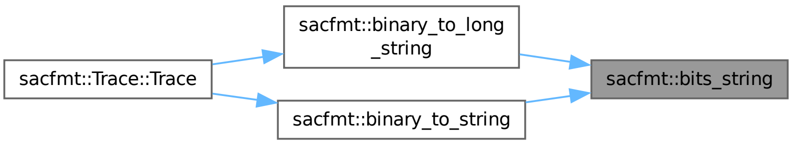
Returns

std::string String converted from bitset.

```
                                                                         {
      std::string result{};
      result.reserve(num_words * word_length);
      constexpr size_t char_size{bits_per_byte};
```

```
  char_bit byte{};
  for (size_t i{0}; i < num_words * binary_word_size; i += char_size) {
    for (size_t j{0}; j < char_size; ++j) [[likely]] {
      byte[j] = bits[i + j];
    }
    result.push_back(static_cast<char>(byte.to_ulong()));
  }
  return result;
}
```

Here is the caller graph for this function:



10.1.4.9 bool_to_binary()

```
word_one sacfmt::bool_to_binary (
              const bool flag )  [noexcept]
```

Convert a boolean to a 32-bit (one word) binary bitset.

Parameters

in	*flag*	Boolean value to be converted to a bitset (sets zeroth element).

Returns

word_one Converted binary bitset.

```
                                                {
  word_one result{};
  result[0] = flag;
  return result;
}
```

10.1.4.10 bool_to_word()

```
std::vector< char > sacfmt::bool_to_word (
              const bool flag )  [noexcept]
```

Convert boolean to a word for writing.

Parameters

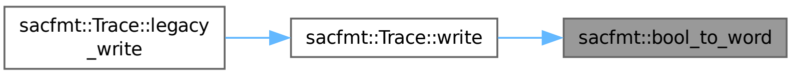
in	*flag*	Boolean to be converted.

Returns

std::vector<char> Prepared value for writing.

```
                                                    {
  std::vector<char> result;
  result.resize(word_length);
  result[0] = static_cast<char>(flag ? 1 : 0);
  for (size_t i{1}; i < word_length; ++i) {
    result[i] = 0;
  }
  return result;
}
```

Here is the caller graph for this function:



10.1.4.11 concat_words() [1/2]

```
word_two sacfmt::concat_words (
            const word_pair< word_one > & pair_words )  [noexcept]
```

Concatenate two word_one binary strings into a single word_two string.

Useful for reading strings from SAC-files.

Parameters

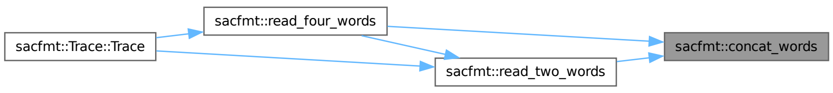
in	*pair_words*	word_pair Words to be concatenated.

Returns

word_two Concatenated words.

```
                                                                  {
  word_two result{};
  for (size_t i{0}; i < binary_word_size; ++i) [[likely]] {
    result[i] = pair_words.first[i];
    result[i + binary_word_size] = pair_words.second[i];
  }
  return result;
}
```

Here is the caller graph for this function:



10.1.4.12 concat_words() [2/2]

```
word_four sacfmt::concat_words (
            const word_pair< word_two > & pair_words )  [noexcept]
```

Concatenate two word_two binary strings into a single word_four string.

Exclusively used to read kEvNm header from SAC-file.

Parameters

in	*pair_words*	word_pair Words to be concatenated.

Returns

word_four Concatenated words.

```
                                                                    {
  word_four result{};
  constexpr size_t two_words{2 * binary_word_size};
  for (size_t i{0}; i < two_words; ++i) [[likely]] {
    result[i] = pair_words.first[i];
    result[i + two_words] = pair_words.second[i];
  }
  return result;
}
```

10.1.4.13 convert_to_word() [1/4]

```
std::vector< char > sacfmt::convert_to_word (
            const double input )  [noexcept]
```

Convert double value into a std::vector<char> for writing.

Parameters

in	*input*	Input value to convert (double).

Returns

std::vector<char> Prepared for writing to binary SAC-file.

```
                                                          {
  std::array<char, static_cast<size_t>(2) * word_length> tmp{};
  // Copy bytes from input into the tmp array
  // flawfinder: ignore
  std::memcpy(tmp.data(), &input, static_cast<size_t>(2) * word_length);
  std::vector<char> word{};
  word.resize(static_cast<size_t>(2) * word_length);
  for (size_t i{0}; i < 2 * word_length; ++i) {
    word[i] = tmp[i];
  }
  return word;
}
```

10.1.4.14 convert_to_word() [2/4]

```
template std::vector< char > sacfmt::convert_to_word (
            const float input )  [noexcept]
```

10.1.4.15 convert_to_word() [3/4]

```
template std::vector< char > sacfmt::convert_to_word (
            const int x )  [noexcept]
```

10.1.4.16 convert_to_word() [4/4]

```
template<typename T >
std::vector< char > sacfmt::convert_to_word (
            const T input )  [noexcept]
```

Template function to convert input value into a std::vector<char> for writing.

Parameters

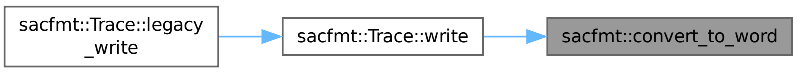
in	*input*	Input value (float or int) to convert.

Returns

std::vector<char> Prepared for writing to binary SAC-file.

```
                                                          {
        std::array<char, word_length> tmp{};
        // Copy bytes from input into the tmp array
        // flawfinder: ignore
        std::memcpy(tmp.data(), &input, word_length);
        std::vector<char> word{};
        word.resize(word_length);
        for (size_t i{0}; i < word_length; ++i) [[likely]] {
          word[i] = tmp[i];
        }
        return word;
      }
```

Here is the caller graph for this function:



10.1.4.17 convert_to_words() [1/2]

```
template std::array< char, word_length > sacfmt::convert_to_words (
            const std::string & str,
            const int n_words )  [noexcept]
```

10.1.4.18 convert_to_words() [2/2]

```
template<size_t N>
template std::array< char, 4 *word_length > sacfmt::convert_to_words (
            const std::string & str,
            int n_words )  [noexcept]
```

Template function to convert input string value into a std::array<char> for writing.

Parameters

in	*str*	Input string to convert.
in	*n_words*	Number of words

Returns

std::array<char, N> Prepared for writing to a binary SAC-file.

```
                                                          {
  std::array<char, N> all_words{};
  // String to null-terminated character array
  const char *c_str = str.c_str();
  for (size_t i{0}; i < static_cast<size_t>(n_words) * word_length; ++i) {
    all_words[i] = c_str[i];
  }
  return all_words;
}
```

10.1.4.19 degrees_to_radians()

```
double sacfmt::degrees_to_radians (
              const double degrees )  [noexcept]
```

Convert decimal degrees to radians.

$$r = d \cdot \frac{\pi}{180^\circ}$$

Parameters

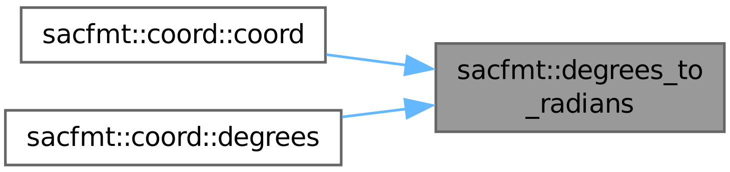
in	*degrees*	Angle in decimal degrees to be converted.

Returns

double Angle in radians.

```
                                                        {
  return rad_per_deg * degrees;
}
```

Here is the caller graph for this function:



10.1.4.20 double_to_binary()

```
word_two sacfmt::double_to_binary (
            const double num ) [noexcept]
```

Convert double-precision value to 64-bit (two words) binary bitset.

Converts double to unsigned-integer of same size for storage in bitset.

Parameters

in	*num*	Double value to be converted.

Returns

word_two Converted value.

```
                                                        {
  unsigned_int<double> num_as_uint{0};
  // flawfinder: ignore
  std::memcpy(&num_as_uint, &num, sizeof(double));
  word_two result{num_as_uint};
  return result;
}
```

10.1.4.21 equal_within_tolerance() [1/2]

```
bool sacfmt::equal_within_tolerance (
            const double val1,
            const double val2,
            const double tolerance ) [noexcept]
```

Check if two double values are equal within a tolerance limit.

Default tolerance is f_eps.

Parameters

in	*val1*	First double in comparison.
in	*val2*	Second double in comparison.
in	*tolerance*	Numerical equality tolerance (default f_eps).

Returns

bool Boolean equality value.

```
                                                                    {
  return std::abs(val1 - val2) < tolerance;
}
```

10.1.4.22 equal_within_tolerance() [2/2]

```
bool sacfmt::equal_within_tolerance (
            const std::vector< double > & vector1,
            const std::vector< double > & vector2,
            const double tolerance )  [noexcept]
```

Check if two std::vector<double> are equal within a tolerance limit.

Default tolerance is f_eps.

Parameters

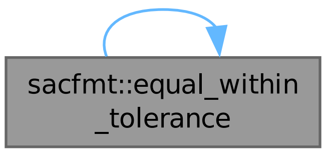
in	*vector1*	First data vector in comparison.
in	*vector2*	Second data vector in comparison.
in	*tolerance*	Numerical equality tolerance (default f_eps).

Returns

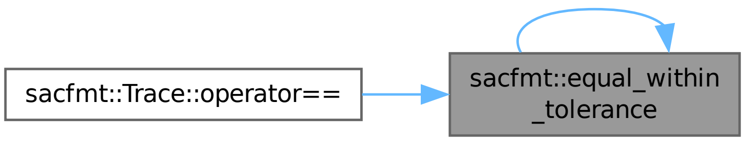
bool Boolean equality value.

```
                                                             {
  if (vector1.size() != vector2.size()) {
    return false;
  }
  for (size_t i{0}; i < vector1.size(); ++i)  [[likely]] {
    if (!equal_within_tolerance(vector1[i], vector2[i], tolerance)) {
      return false;
    }
  }
  return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.23 float_to_binary()

```
word_one sacfmt::float_to_binary (
            const float num ) [noexcept]
```

Convert floating-point value to 32-bit (one word) binary bitset.

Converts float to unsigned-integer of same size for storage in bitset.

Parameters

in	*num*	Float value to be converted.

Returns

word_one Converted value.

```
                                                    {
  unsigned_int<float> num_as_uint{0};
  // flawfinder: ignore
  std::memcpy(&num_as_uint, &num, sizeof(float));
  word_one result{num_as_uint};
  return result;
}
```

10.1.4.24 gcarc()

```
double sacfmt::gcarc (
            const point location1,
            const point location2 ) [noexcept]
```

Calculate great circle arc distance in decimal degrees between two points.

Assumes spherical Earth (in future will include flatenning and adjustable radius for other bodies/greater accuracy).

ϕ is latitude. λ is longitude. Δ is great circle arc distance (gcarc).

$$\Delta = cos^{-1}\left(sin(\phi_1)sin(\phi_2) + cos(\phi_1)cos(\phi_2)cos(\lambda_2 - \lambda_1)\right)$$

Parameters

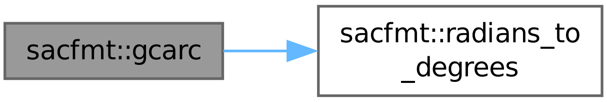
in	*location1*	point of first location.
in	*location2*	point of second location

Returns

double The great circle arc distance in decimal degrees.

```
                                                                     {
  return radians_to_degrees(
      std::acos(std::sin(location1.latitude.radians()) *
                    std::sin(location2.latitude.radians()) +
                std::cos(location1.latitude.radians()) *
                    std::cos(location2.latitude.radians()) *
                    std::cos(location2.longitude.radians() -
                                   location1.longitude.radians())));
}
```

Here is the call graph for this function:



10.1.4.25 int_to_binary()

```
word_one sacfmt::int_to_binary (
               int num )  [noexcept]
```

Convert integer to 32-bit (one word) binary bitset.

Uses two's complement to convert an integer into a binary value.

Parameters

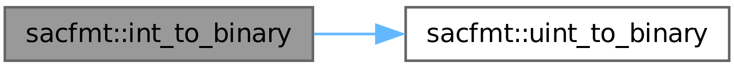
in	*num*	Number to be converted.

Returns

word_one Converted value.

```
                                        {
  word_one bits{};
  if (num >= 0) {
    bits = uint_to_binary(static_cast<uint>(num));
  } else {
    bits = uint_to_binary(static_cast<uint>(-num));
    // Complement
    bits.flip();
    bits = bits.to_ulong() + 1;
  }
  return bits;
}
```

Here is the call graph for this function:



10.1.4.26 limit_180()

```
double sacfmt::limit_180 (
            const double degrees )  [noexcept]
```

Takes a decimal degree value and constrains it to a half circle using symmetry.

$$[-\infty, \infty] \rightarrow (-180, 180]$$

Parameters

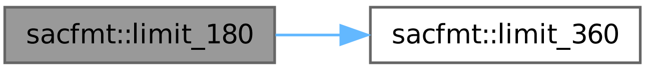
in	*degrees*	Decimal degrees to be constrained.

Returns

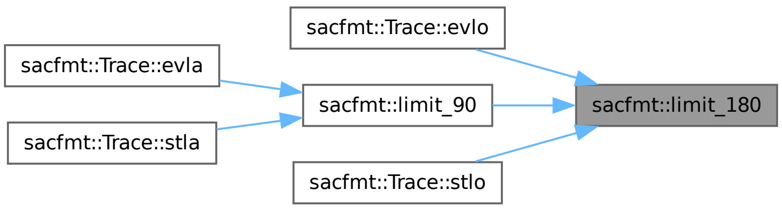
double Value within limits.

```
                                               {
  double result{limit_360(degrees)};
  constexpr double hemi{180.0};
  if (result > hemi) {
    result = result - circle_deg;
  }
  return result;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.27 limit_360()

```
double sacfmt::limit_360 (
            const double degrees )  [noexcept]
```

Takes a decimal degree value and constrains it to full circle using symmetry.

$$[-\infty, \infty] \rightarrow [0, 360]$$

Parameters

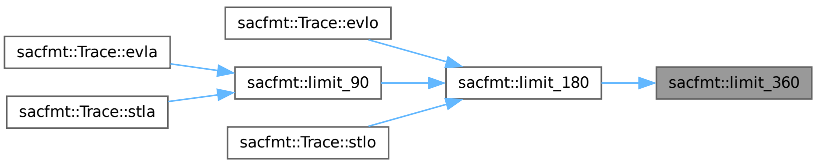
in	*degrees*	Decimal degrees to be constrained.

Returns

double Value within limits.

```
                                                     {
  double result{degrees};
  while (std::abs(result) > circle_deg) {
    if (result > circle_deg) {
      result -= circle_deg;
    } else {
      result += circle_deg;
    }
  }
  if (result < 0) {
    result += circle_deg;
  }
  return result;
}
```

Here is the caller graph for this function:



10.1.4.28 limit_90()

```
double sacfmt::limit_90 (
            const double degrees )  [noexcept]
```

Takes a decimal degree value and constrains it to a quarter circle using symmetry.

$$[-\infty, \infty] \rightarrow [-90, 90]$$

Parameters

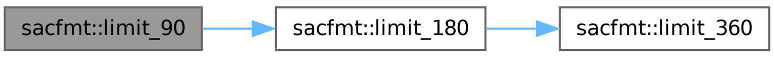
in	*degrees*	Decimal degrees to be constrained.

Returns

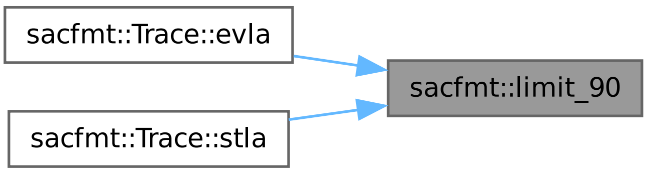
double Value within limits.

```
                                              {
  double result{limit_180(degrees)};
  constexpr double quarter{90.0};
  if (result > quarter) {
    result = (2 * quarter) - result;
  } else if (result < -quarter) {
    result = (-2 * quarter) - result;
  }
  return result;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.29 long_string_to_binary()

```
word_four sacfmt::long_string_to_binary (
              std::string str )  [noexcept]
```

Convert a string to a 128-bit (four word) binary bitset.

If the string is longer than 16 characters, then only the first 16 characters are kept. If the string is less than 16 characters long, it is right-padded with spaces.

Exclusively used to work with the kEvNm header.

Parameters

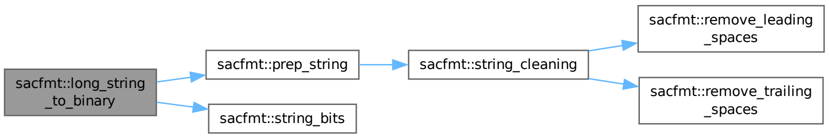
in	*str*	String to be converted to a bitset.

Returns

word_four Converted binary bitset.

```
                                                              {
  constexpr size_t string_size{4 * word_length};
  prep_string(&str, string_size);
  // Four words (16 characters)
  word_four bits{};
  string_bits(&bits, str, string_size);
  return bits;
}
```

Here is the call graph for this function:


sacfmt::long_string_to_binary
sacfmt::prep_string
sacfmt::string_cleaning
sacfmt::remove_leading_spaces
sacfmt::remove_trailing_spaces
sacfmt::string_bits

10.1.4.30 nwords_after_current()

```
bool sacfmt::nwords_after_current (
            std::ifstream * sac,
            const read_spec & spec )  [noexcept]
```

Determine if the SAC-file has enough remaining data to read the requested amount of data.

Parameters

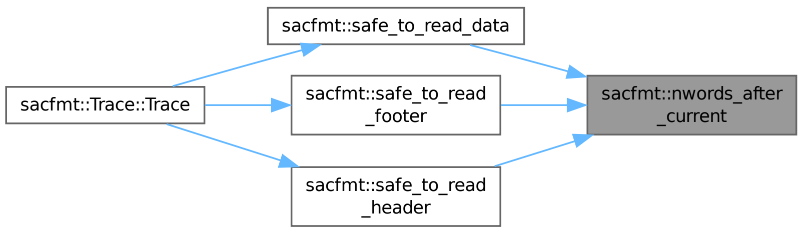
in	*sac*	std::ifstream∗ SAC-file to read.
in	*spec*	read_spec reading specification.

Returns

bool Truth value (true = safe to read).

```
                                                                                    {
  bool result{false};
  if (sac->good()) {
    sac->seekg(0, std::ios::end);
    const std::size_t final_pos{static_cast<size_t>(sac->tellg())};
    // Doesn't like size_t since it wants to allow
    // the possibility of negative offsets (not how I use it)
    sac->seekg(static_cast<std::streamoff>(spec.start_word));
    const std::size_t diff{final_pos - spec.start_word};
    result = (diff >= (spec.num_words * word_length));
  }
  return result;
}
```

Here is the caller graph for this function:


sacfmt::safe_to_read_data
sacfmt::Trace::Trace
sacfmt::safe_to_read_footer
sacfmt::nwords_after_current
sacfmt::safe_to_read_header

10.1.4.31 prep_string()

```
void sacfmt::prep_string (
             std::string * str,
             const size_t str_size )  [noexcept]
```

Cleans string and then truncates/pads as necessary.

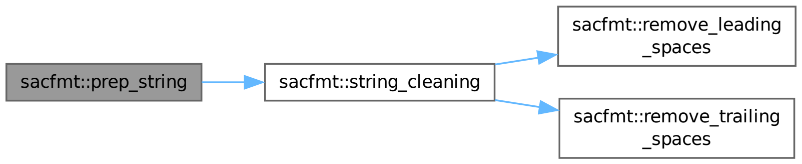
This edits the string in-place.

Parameters

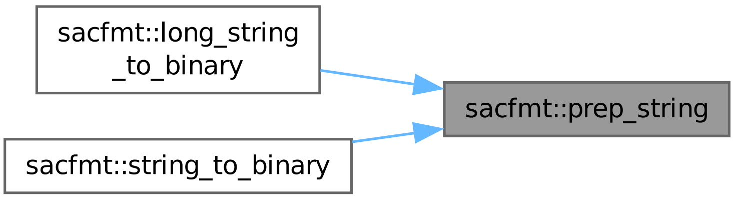
in,out	*str*	std::string* String to be prepared.
in	*str_size*	Desired string length.

```
                                                                    {
  *str = string_cleaning(*str);
  if (str->length() > str_size) {
    str->resize(str_size);
  } else if (str->length() < str_size) {
    *str = str->append(str_size - str->length(), ' ');
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.32 radians_to_degrees()

```
double sacfmt::radians_to_degrees (
            const double radians ) [noexcept]
```

Convert radians to decimal degrees.

$$d = r \cdot \frac{180^\circ}{\pi}$$

Parameters

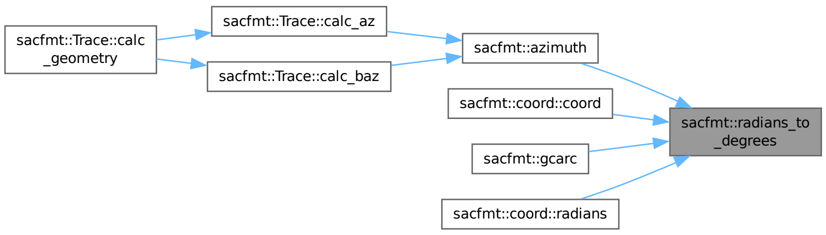
in	*radians*	Angle in radians to be converted.

Returns

double Angle in decimal degrees.

```
                                                              {
    return deg_per_rad * radians;
}
```

Here is the caller graph for this function:



10.1.4.33 read_data()

```
std::vector< double > sacfmt::read_data (
            std::ifstream * sac,
            const read_spec & spec )
```

Reader arbitrary number of words (useful for vectors) from a binary SAC-file.

Note that this modifies the position of the reader within the stream (to the end of the read words).

Parameters

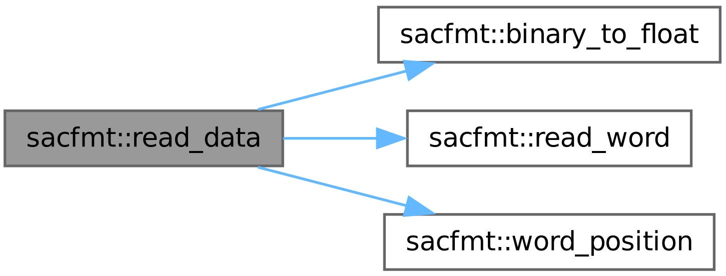
in,out	*sac*	std::ifstream∗ Input binary SAC-file.
in	*spec*	read_spec Reading specification.

Returns

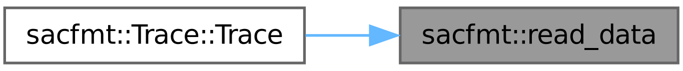
std::vector<double> Data vector read in.

```
                                                                       {
  sac->seekg(word_position(spec.start_word));
  std::vector<double> result{};
  result.resize(spec.num_words);
  for (size_t i{0}; i < spec.num_words; ++i) [[likely]] {
    result[i] = static_cast<double>(binary_to_float(read_word(sac)));
  }
  return result;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.34 read_four_words()

```
word_four sacfmt::read_four_words (
            std::ifstream * sac )
```

Read four words (128 bits, kEvNm only) from a binary SAC-file.

Note that this modifies the position of the reader within the stream (to the end of the read words).

Parameters

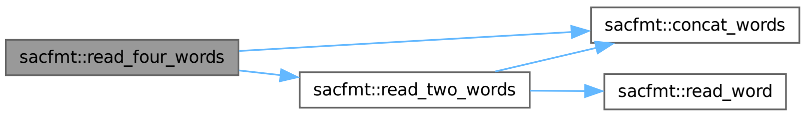
in,out	*sac*	std::ifstream* Input binary SAC-file.

Returns

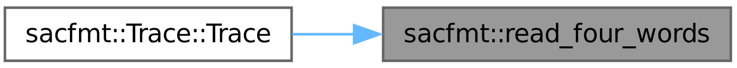
word_four Binary bitset representation of four words.

```
                                              {
  const word_two first_words{read_two_words(sac)};
  const word_two second_words{read_two_words(sac)};
  word_pair<word_two> pair_words{};
  if constexpr (std::endian::native == std::endian::little) {
    pair_words.first = first_words;
    pair_words.second = second_words;
  } else {
    pair_words.first = second_words;
    pair_words.second = first_words;
  }
  return concat_words(pair_words);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.35 read_two_words()

```
word_two sacfmt::read_two_words (
            std::ifstream * sac )
```

Read two words (64 bits, useful for most strings) from a binary SAC-file.

Note that this modifies the position of the reader within the stream (to the end of the read words).

Parameters

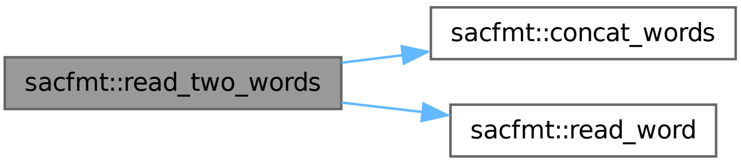
in,out	*sac*	std::ifstream∗ Input binary SAC-file.

Returns

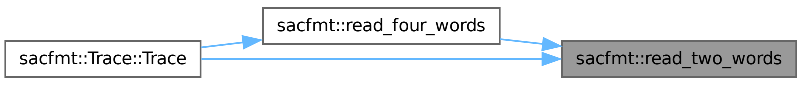
word_two Binary bitset representation of two words.

```
                                                      {
  const word_one first_word{read_word(sac)};
  const word_one second_word{read_word(sac)};
  word_pair<word_one> pair_words{};
  if constexpr (std::endian::native == std::endian::little) {
    pair_words.first = first_word;
    pair_words.second = second_word;
  } else {
    pair_words.first = second_word;
    pair_words.second = first_word;
  }
  return concat_words(pair_words);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.36 read_word()

```
word_one sacfmt::read_word (
            std::ifstream * sac )
```

Read one word (32 bits, useful for non-strings) from a binary SAC-File.

Note that this modifies the position of the reader within the stream (to the end of the read word).

Parameters

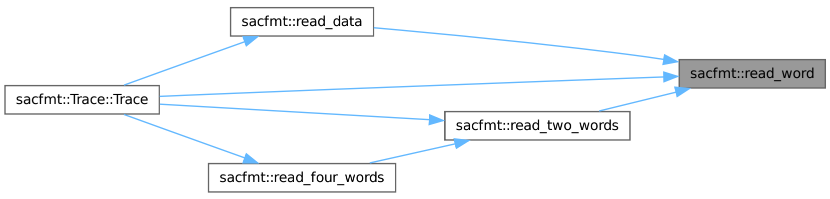
in,out	*sac*	std::ifstream* Input binary SAC-file.

Returns

word_one Binary bitset representation of single word.

```
                                          {
  word_one bits{};
  constexpr size_t char_size{bits_per_byte};
  // Where we will store the characters
  std::array<char, word_length> word{};
  // Read to our character array
  // This can always hold the source due to careful typing/sizing
  // flawfinder: ignore
  if (sac->read(word.data(), word_length)) {
    // Take each character
    for (size_t i{0}; i < word_length; ++i) [[likely]] {
      uint character{static_cast<uint>(word[i])};
      char_bit byte{character};
      // bit-by-bit
      for (size_t j{0}; j < char_size; ++j) [[likely]] {
        bits[(i * char_size) + j] = byte[j];
      }
    }
  }
  return bits;
}
```

Here is the caller graph for this function:



10.1.4.37 remove_leading_spaces()

```
void sacfmt::remove_leading_spaces (
             std::string * str ) [noexcept]
```

Remove all leading spaces from a string.

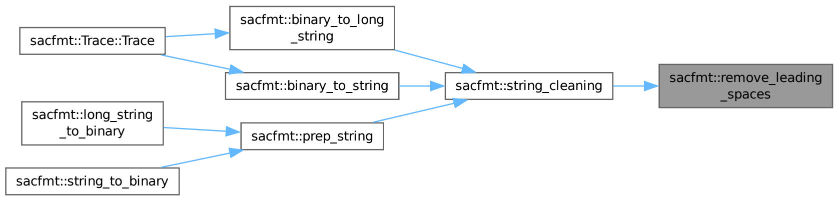
This edits the string in-place.

Parameters

in,out	*str*	std::string* String to have spaces removed.

```
                                                            {
  while ((static_cast<int>(str->front()) <= ascii_space) && (!str->empty())) {
    str->erase(0, 1);
  }
}
```

Here is the caller graph for this function:



10.1.4.38 remove_trailing_spaces()

```
void sacfmt::remove_trailing_spaces (
            std::string * str ) [noexcept]
```

Remove all trailing spaces from a string.

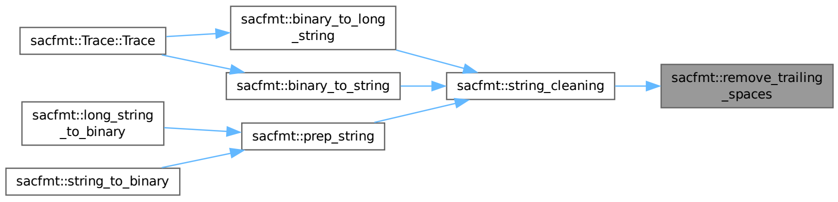
This edits the string in-place.

Parameters

in,out	*str*	std::string∗ String to have spaces removed.

```
                                                              {
  while ((static_cast<int>(str->back()) <= ascii_space) && (!str->empty())) {
    str->pop_back();
  }
}
```

Here is the caller graph for this function:



10.1.4.39 safe_to_finish_reading()

```
void sacfmt::safe_to_finish_reading (
            std::ifstream * sac )
```

Determines if the SAC-file is finished.

This must run after reading the header, data vector(s), and footer (if applicable). This checks to ensure there is no additional data in the SAC-file (there shouldn't be, and out of safety it throws an io_error to inform the user if there are shenanigans).

Parameters

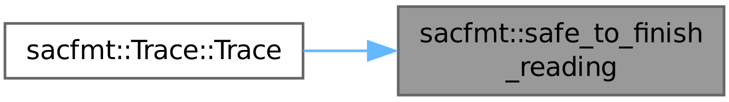
in	*sac*	std::ifstream∗ SAC-file to be checked.

Exceptions

io_error	If the file is not finished.

```
                                              {
  const std::streamoff current_pos{sac->tellg()};
  sac->seekg(0, std::ios::end);
  const std::streamoff end_pos{sac->tellg()};
  sac->seekg(current_pos, std::ios::beg);
  // How far are we from the end of the file?
  const std::streamoff diff{end_pos - current_pos};
  // If there is more, something weird happened...
  if (diff != 0) {
    std::ostringstream oss{};
    oss « "Filesize exceeds data specification with ";
    oss « diff;
    oss « " bytes excess. Data corruption suspected.";
    throw io_error(oss.str());
  }
}
```

Here is the caller graph for this function:



10.1.4.40 safe_to_read_data()

```
void sacfmt::safe_to_read_data (
              std::ifstream * sac,
              const size_t n_words,
              const bool data2 )
```

Determines if the SAC-file has enough space remaining to contain a complete data vector.

This must be run after reading the header (and first data vector if applicable) and before the footer (if applicable).

Parameters

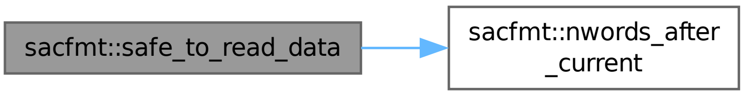
in	*sac*	std::ifstream∗ SAC-file to read.
in	*n_words*	Number of values in data vector.
in	*data2*	bool True if reading data2, false (default) if reading data1.

Exceptions

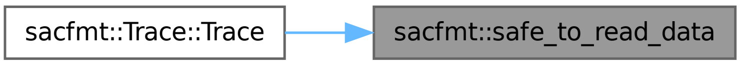
io_error	If unsafe to read.

```
                                                  {
  const std::string data{data2 ? "data2" : "data1"};
  const read_spec spec{n_words, static_cast<size_t>(sac->tellg())};
  if (!nwords_after_current(sac, spec)) {
    throw io_error("Insufficient filesize for " + data + '.');
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.41 safe_to_read_footer()

```
void sacfmt::safe_to_read_footer (
            std::ifstream * sac )
```

Determines if the SAC-file has enough space remaining to contain a complete footer.

This must be run after reading the header and data vector(s), not before.

Parameters

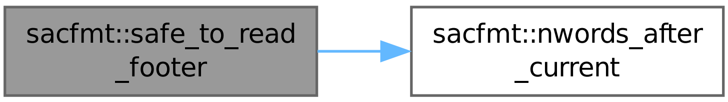
in	*sac*	std::ifstream∗ SAC-file to read.

Exceptions

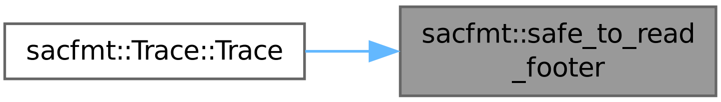
io_error	If unsafe to read.

```
                                                    {
  // doubles are two words long
  const read_spec spec{static_cast<size_t>(num_footer) * 2,
                       static_cast<size_t>(sac->tellg())};
  if (!nwords_after_current(sac, spec)) {
    throw io_error("Insufficient filesize for footer.");
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.42 safe_to_read_header()

```
void sacfmt::safe_to_read_header (
            std::ifstream * sac )
```

Determine if the SAC-file is large enough to contain a complete header.

This must be run prior to reading the data vector(s) and footer (if applicable), not after.

Parameters

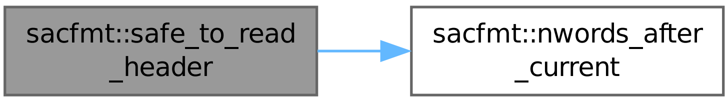
in	*sac*	std::ifstream∗ SAC-file to read.

Exceptions

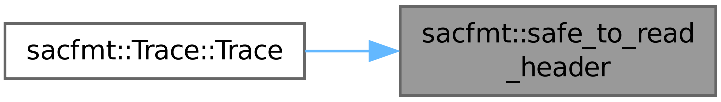
io_error	If unsafe to read.

```
                                          {
  const read_spec spec{data_word, 0};
  if (!nwords_after_current(sac, spec)) {
    throw io_error("Insufficient filesize for header.");
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.43 string_bits()

```
template<typename T >
void sacfmt::string_bits (
            T * bits,
            const std::string & str,
            const size_t str_size )  [noexcept]
```

Template function to convert string into binary bitset.

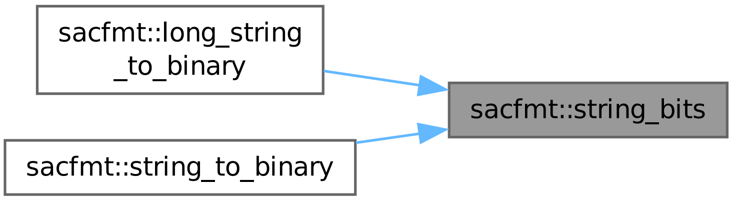
Note that this edits the bitset in place.

Parameters

out	*bits*	Destintation bitset for the string (result).
in	*str*	String to undergo conversion.
in	*str_size*	Desired string size in words (4 chars = 1 word).

```
                                                  {
  constexpr size_t char_size{bits_per_byte};
  char_bit byte{};
  for (size_t i{0}; i < str_size; ++i) {
    size_t character{static_cast<size_t>(str[i])};
    byte = char_bit(character);
    for (size_t j{0}; j < char_size; ++j) {
      (*bits)[(i * char_size) + j] = byte[j];
    }
  }
}
```

Here is the caller graph for this function:



10.1.4.44 string_cleaning()

```
std::string sacfmt::string_cleaning (
            const std::string & str )  [noexcept]
```

Remove leading/trailing spaces and control characters from a string.

Parameters

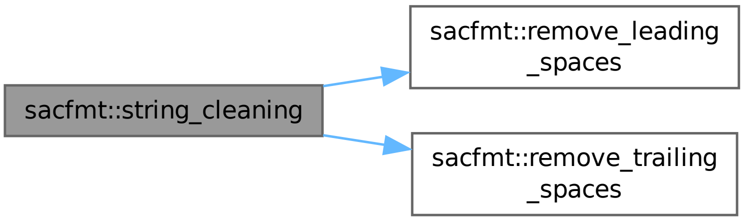
in	*str*	std::string String to be cleaned.

Returns

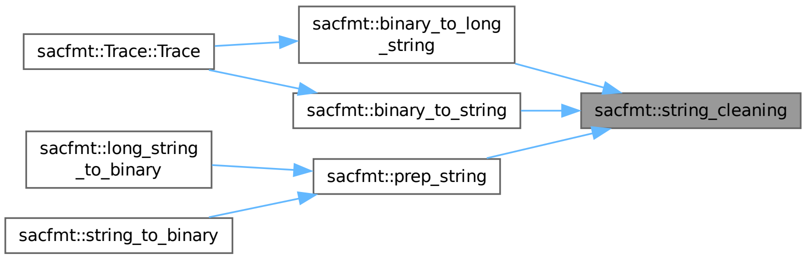
std::string Cleaned string.

```
                                                     {
    std::string result{str};
    size_t null_position{str.find('\0')};
    if (null_position != std::string::npos) {
      result.erase(null_position);
    }
    remove_leading_spaces(&result);
    remove_trailing_spaces(&result);
    return result;
  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.1.4.45 string_to_binary()

```
word_two sacfmt::string_to_binary (
            std::string str ) [noexcept]
```

Convert string to a 64-bit (two word) binary bitset.

If the string is longer than 8 characters, then only the first 8 characters are kept. If the string is less than 8 characters long, it is right-padded with spaces.

Parameters

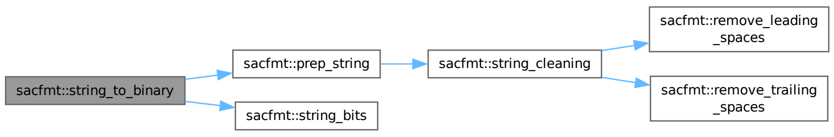
in	*str*	String to be converted to a bitset.

Returns

word_two Converted binary bitset.

```
                                                       {
  constexpr size_t string_size{2 * word_length};
  // 1 byte per character
  prep_string(&str, string_size);
  // Two words (8 characters)
  word_two bits{};
  string_bits(&bits, str, string_size);
  return bits;
}
```

Here is the call graph for this function:



10.1.4.46 uint_to_binary()

```
word_one sacfmt::uint_to_binary (
            uint num )  [noexcept]
```

Convert unsigned integer to 32-bit (one word) binary bitset.

This sets the current bit using bitwise and, updates the bit to manipulate and performs a right-shift (division by 2) until the number is zero.

Parameters

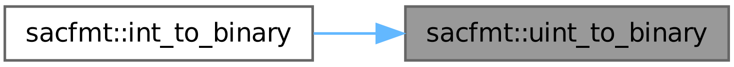
in	*num*	Number to be converted.

Returns

word_one Converted value.

```
                                             {
  word_one bits{};
  for (size_t pos{0}; pos < bits.size(); ++pos) {
    if (num > 0) {
      // Bitwise and to set flag.
      bits.set(pos, static_cast<bool>(num & 1));
      // Right-shift bits by 1, same as division by 2
      num »= 1;
    } else {
      break;
    }
  }
  return bits;
}
```

Here is the caller graph for this function:



10.1.4.47 word_position()

```
std::streamoff sacfmt::word_position (
            const size_t word_number )  [noexcept]
```

Calculates position of word in SAC-file.

Multiplies given word number by the word-length in bytes (defined by the SAC format.)

Parameters

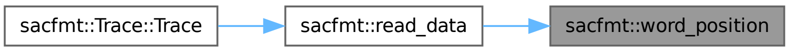
in	*word_number*	Number of desired word in file stream.

Returns

std::streamoff Position in SAC-file of desired word (in bytes).

```
                                                        {
  return static_cast<std::streamoff>(word_number * word_length);
}
```

Here is the caller graph for this function:



10.1.4.48 write_words()

```
void sacfmt::write_words (
            std::ofstream * sac_file,
            const std::vector< char > & input )
```

Write arbitrary number of words (useful for vectors) to a binary SAC-file.

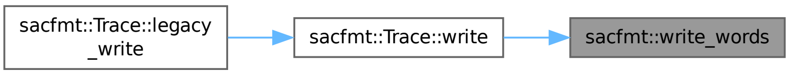
Note that this modifies the position of the writer within the stream (to the end of the written words).

Parameters

in,out	*sac_file*	std::ofstream* Output binary SAC-file.
in	*input*	std::vector<char> Character vector representation of data for writing.

```
                                                                  {
  std::ofstream &sac = *sac_file;
  if (sac.is_open()) {
    for (char character : input) [[likely]] {
      sac.write(&character, sizeof(char));
    }
  }
}
```

Here is the caller graph for this function:



10.1.5 Variable Documentation

10.1.5.1 ascii_space

```
constexpr int sacfmt::ascii_space {32}  [constexpr]
```

ASCII-code of 'space' character.

```
{32};
```

10.1.5.2 binary_word_size

```
constexpr size_t sacfmt::binary_word_size {word_length * bits_per_byte}  [constexpr]
```

Size (bits) of funamental data-chunk.

```
00066 {word_length * bits_per_byte};
```

10.1.5.3 bits_per_byte

```
constexpr size_t sacfmt::bits_per_byte {8}  [constexpr]
```

Size (bits) of binary character.

```
00064 {8};
```

10.1.5.4 circle_deg

```
constexpr double sacfmt::circle_deg {360.0}  [constexpr]
```

Degrees in a circle.

```
00116 {360.0};
```

10.1.5.5 common_skip_num

```
constexpr int sacfmt::common_skip_num {7}  [constexpr]
```

Extremely common number of 'internal use' headers in SAC format.

```
00110 {7};
```

10.1.5.6 data_word

```
constexpr std::streamoff sacfmt::data_word {158}  [constexpr]
```

First word of (first) data-section (stream offset).

```
00068 {158};
```

10.1.5.7 deg_per_rad

```
constexpr double sacfmt::deg_per_rad {1.0 / rad_per_deg}  [constexpr]
```

Degrees per radian.

```
00114 {1.0 / rad_per_deg};
```

10.1.5.8 earth_radius

```
constexpr double sacfmt::earth_radius {6378.14}  [constexpr]
```

Average radius of Earth (kilometers).

```
00118 {6378.14};
```

10.1.5.9 f_eps

```
constexpr float sacfmt::f_eps {2.75e-6F} [constexpr]
```

Accuracy precision expected of SAC floating-point values.

```
00080 {2.75e-6F};
```

10.1.5.10 modern_hdr_version

```
constexpr int sacfmt::modern_hdr_version {7} [constexpr]
```

nVHdr value for newest SAC format (2020+).

```
00106 {7};
```

10.1.5.11 num_bool

```
constexpr int sacfmt::num_bool {4} [constexpr]
```

Number of boolean header values in SAC format.

```
00098 {4};
```

10.1.5.12 num_data

```
constexpr int sacfmt::num_data {2} [constexpr]
```

Number of data arrays in SAC format.

```
00102 {2};
```

10.1.5.13 num_double

```
constexpr int sacfmt::num_double {22} [constexpr]
```

Number of double-precision header values in SAC format.

```
00094 {22};
```

10.1.5.14 num_float

```
constexpr int sacfmt::num_float {39} [constexpr]
```

Number of float-poing header values in SAC format.

```
00092 {39};
```

10.1.5.15 num_footer

```
constexpr int sacfmt::num_footer {22} [constexpr]
```

Number of double-precision footer values in SAC format (version 7).

```
00104 {22};
```

10.1.5.16 num_int

```
constexpr int sacfmt::num_int {26}  [constexpr]
```

Number of integer header values in SAC format.

```
{26};
```

10.1.5.17 num_string

```
constexpr int sacfmt::num_string {23}  [constexpr]
```

Number of string header values in SAC format.

```
{23};
```

10.1.5.18 old_hdr_version

```
constexpr int sacfmt::old_hdr_version {6}  [constexpr]
```

nVHdr value for historic SAC format (pre-2020).

```
{6};
```

10.1.5.19 rad_per_deg

```
constexpr double sacfmt::rad_per_deg {std::numbers::pi_v<double> / 180.0}  [constexpr]
```

Radians per degree.

```
{std::numbers::pi_v<double> / 180.0};
```

10.1.5.20 sac_map

```
const std::unordered_map<name, const size_t> sacfmt::sac_map
```

Lookup table for variable locations.

Maps SAC variables (headers and data) to their internal locations in the Trace class.

```
                                                        {
    // Floats
    {name::depmin, 0},
    {name::depmax, 1},
    {name::odelta, 2},
    {name::resp0, 3},
    {name::resp1, 4},
    {name::resp2, 5},
    {name::resp3, 6},
    {name::resp4, 7},
    {name::resp5, 8},
    {name::resp6, 9},
    {name::resp7, 10},
    {name::resp8, 11},
    {name::resp9, 12},
    {name::stel, 13},
    {name::stdp, 14},
    {name::evel, 15},
    {name::evdp, 16},
    {name::mag, 17},
    {name::user0, 18},
    {name::user1, 19},
    {name::user2, 20},
    {name::user3, 21},
    {name::user4, 22},
    {name::user5, 23},
    {name::user6, 24},
    {name::user7, 25},
    {name::user8, 26},
    {name::user9, 27},
    {name::dist, 28},
    {name::az, 29},
    {name::baz, 30},
    {name::gcarc, 31},
    {name::depmen, 32},
    {name::cmpaz, 33},
    {name::cmpinc, 34},
    {name::xminimum, 35},
    {name::xmaximum, 36},
    {name::yminimum, 37},
    {name::ymaximum, 38},
    // Doubles
    {name::delta, 0},
    {name::b, 1},
    {name::e, 2},
    {name::o, 3},
    {name::a, 4},
    {name::t0, 5},
    {name::t1, 6},
    {name::t2, 7},
    {name::t3, 8},
    {name::t4, 9},
    {name::t5, 10},
    {name::t6, 11},
    {name::t7, 12},
    {name::t8, 13},
    {name::t9, 14},
    {name::f, 15},
    {name::stla, 16},
    {name::stlo, 17},
    {name::evla, 18},
    {name::evlo, 19},
    {name::sb, 20},
    {name::sdelta, 21},
    // Ints
    {name::nzyear, 0},
    {name::nzjday, 1},
    {name::nzhour, 2},
    {name::nzmin, 3},
    {name::nzsec, 4},
    {name::nzmsec, 5},
    {name::nvhdr, 6},
    {name::norid, 7},
    {name::nevid, 8},
    {name::npts, 9},
    {name::nsnpts, 10},
    {name::nwfid, 11},
    {name::nxsize, 12},
    {name::nysize, 13},
    {name::iftype, 14},
    {name::idep, 15},
    {name::iztype, 16},
    {name::iinst, 17},
    {name::istreg, 18},
    {name::ievreg, 19},
    {name::ievtyp, 20},
    {name::iqual, 21},
    {name::isynth, 22},
    {name::imagtyp, 23},
    {name::imagsrc, 24},
    {name::ibody, 25},
    // Bools
    {name::leven, 0},
    {name::lpspol, 1},
    {name::lovrok, 2},
    {name::lcalda, 3},
    // Strings
    {name::kstnm, 0},
    {name::kevnm, 1},
    {name::khole, 2},
    {name::ko, 3},
    {name::ka, 4},
    {name::kt0, 5},
    {name::kt1, 6},
    {name::kt2, 7},
    {name::kt3, 8},
    {name::kt4, 9},
    {name::kt5, 10},
    {name::kt6, 11},
    {name::kt7, 12},
    {name::kt8, 13},
    {name::kt9, 14},
    {name::kf, 15},
    {name::kuser0, 16},
    {name::kuser1, 17},
    {name::kuser2, 18},
    {name::kcmpnm, 19},
    {name::knetwk, 20},
    {name::kdatrd, 21},
    {name::kinst, 22},
    // Data
    {name::data1, 0},
    {name::data2, 1}};
```

10.1.5.21 unset_bool

```
constexpr bool sacfmt::unset_bool {false}  [constexpr]
```

Boolean unset value (SAC Magic).

```
{false};
```

10.1.5.22 unset_double

```
constexpr double sacfmt::unset_double {-12345.0}  [constexpr]
```

Double-precision unset value (SAC Magic).

```
{-12345.0};
```

10.1.5.23 unset_float

```
constexpr float sacfmt::unset_float {-12345.0F}  [constexpr]
```

Float-point unset value (SAC Magic).

```
{-12345.0F};
```

10.1.5.24 unset_int

```
constexpr int sacfmt::unset_int {-12345}  [constexpr]
```

Integer unset value (SAC Magic).

```
{-12345};
```

10.1.5.25 unset_word

```
const std::string sacfmt::unset_word {"-12345"}
```

String unset value (SAC Magic).

```
{"-12345"};
```

10.1.5.26 word_length

```
constexpr size_t sacfmt::word_length {4}  [constexpr]
```

Size (bytes) of fundamental data-chunk.

```
{4};
```

10.2 sacfmt::bitset_type Namespace Reference

bitset type-safety namespace.

Classes

- struct uint

  Ensure type-safety for conversions between floats/doubles and bitsets.
- struct uint< 4 ∗bits_per_byte >

  One-word (floats).
- struct uint< bytes ∗bits_per_byte >

  Two-words (doubles)

Variables

- constexpr int bytes {8}

10.2.1 Detailed Description

bitset type-safety namespace.

10.2.2 Variable Documentation

10.2.2.1 bytes

```
constexpr int sacfmt::bitset_type::bytes {8}  [constexpr]
00138 {8};
```

Chapter 11

Class Documentation

11.1 sacfmt::coord Class Reference

Defines a geographic coordinant (degrees/radians)

```
#include <sac_format.hpp>
```

Public Member Functions

- coord () noexcept

    Default coordinate constructor.
- coord (double value, bool degrees=true) noexcept

    Coordinate constructor.
- double degrees () const noexcept

    Get coordinate value in decimal degrees.
- double radians () const noexcept

    Get coordinate value in radians.
- void degrees (double value) noexcept

    Set coordinate value using decimal degrees.
- void radians (double value) noexcept

    Set coordainate value using radians.

Private Attributes

- double deg {}

    coordinate value in decimal degrees.
- double rad {}

    coordinate value in radians.

11.1.1 Detailed Description

Defines a geographic coordinant (degrees/radians)

11.1.2 Constructor & Destructor Documentation

11.1.2.1 coord() [1/2]

```
sacfmt::coord::coord ( )  [noexcept]
```

Default coordinate constructor.

11.1.2.2 coord() [2/2]

```
sacfmt::coord::coord (
            double value,
            bool degrees = true )  [explicit], [noexcept]
```

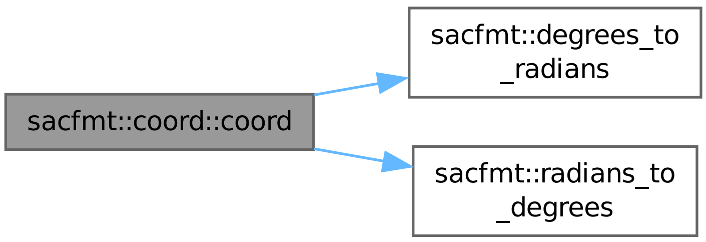
Coordinate constructor.

Parameters

in	*value*	Double value of coordinate
in	*degrees*	Boolean value, true if degrees (false = radians).

```
                                                              {
  if (degrees) {
    deg = value;
    rad = degrees_to_radians(value);
  } else {
    rad = value;
    deg = radians_to_degrees(value);
  }
}
```

Here is the call graph for this function:



11.1.3 Member Function Documentation

11.1.3.1 degrees() [1/2]

```
double sacfmt::coord::degrees ( ) const  [inline], [noexcept]
```

Get coordinate value in decimal degrees.

```
{ return deg; };
```

11.1.3.2 degrees() `[2/2]`

```
void sacfmt::coord::degrees (
             double value )  [noexcept]
```

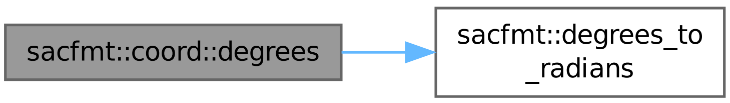
Set coordinate value using decimal degrees.

Parameters

in	*value*	double coordinate in decimal degrees.

```
                                              {
  deg = value;
  rad = degrees_to_radians(value);
}
```

Here is the call graph for this function:



11.1.3.3 radians() `[1/2]`

```
double sacfmt::coord::radians ( ) const  [inline], [noexcept]
```

Get coordinate value in radians.

```
{ return rad; };
```

11.1.3.4 radians() `[2/2]`

```
void sacfmt::coord::radians (
             double value )  [noexcept]
```

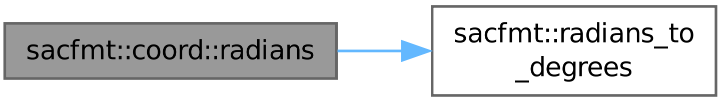
Set coordainate value using radians.

Parameters

in	*value*	double coordinate in radians.

```
                                              {
  rad = value;
  deg = radians_to_degrees(value);
}
```

Here is the call graph for this function:



11.1.4 Member Data Documentation

11.1.4.1 deg

```
double sacfmt::coord::deg {}  [private]
```

coordinate value in decimal degrees.

```
00278 {};
```

11.1.4.2 rad

```
double sacfmt::coord::rad {}  [private]
```

coordinate value in radians.

```
00280 {};
```

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

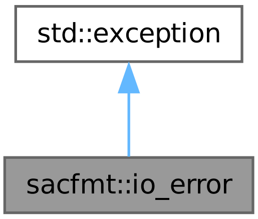
- include/sac-format/sac_format.hpp
- src/sac_format.cpp

11.2 sacfmt::io_error Class Reference

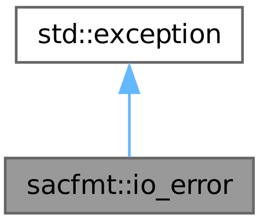
Class for generic I/O exceptions.

```
#include <sac_format.hpp>
```

Inheritance diagram for sacfmt::io_error:



Collaboration diagram for sacfmt::io_error:



Public Member Functions

- io_error (std::string msg)

  io_error Constructor
- const char ∗ what () const noexcept override

  Error message delivery.

Private Attributes

- const std::string message {}

  Error message.

11.2.1 Detailed Description

Class for generic I/O exceptions.

These errors occur due to bad path, bad permissions, or otherwise corrupt SAC-files.

I/O operations may raise other exceptions (disk failure, out of space, etc.), but those are difficult to emulate for testing purposes (therefore I am unable to reliably cover them); they also arise due to conditions that would render how sac-format handles them moot.

11.2.2 Constructor & Destructor Documentation

11.2.2.1 io_error()

```
sacfmt::io_error::io_error (
              std::string msg )  [inline], [explicit]
```

io_error Constructor

Parameters

in	*msg*	std::string Error message.

```
: message(std::move(msg)) {}
```

11.2.3 Member Function Documentation

11.2.3.1 what()

```
const char * sacfmt::io_error::what ( ) const  [inline], [override], [noexcept]
```

Error message delivery.

Returns

what char∗ Error message.

```
                                                          {
    return message.c_str();
  }
```

11.2.4 Member Data Documentation

11.2.4.1 message

```
const std::string sacfmt::io_error::message {}  [private]
```

Error message.

```
{};
```

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

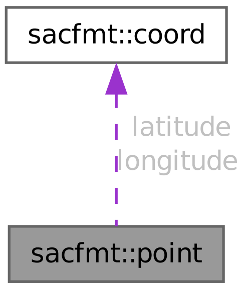
- include/sac-format/sac_format.hpp

11.3 sacfmt::point Struct Reference

Defines a geographic point (latitude, longitude)

`#include <sac_format.hpp>`

Collaboration diagram for sacfmt::point:



Public Member Functions

- point (coord lat, coord lon) noexcept

  Construct point from latitude and longitude.

Public Attributes

- coord latitude {}

  Latitude of point.
- coord longitude {}

  Longitude of point.

11.3.1 Detailed Description

Defines a geographic point (latitude, longitude)

11.3.2 Constructor & Destructor Documentation

11.3.2.1 point()

```
sacfmt::point::point (
            coord lat,
            coord lon )  [inline], [noexcept]
```

Construct point from latitude and longitude.

Parameters

in	*lat*	coord latitude of point.
in	*lon*	coord longitude of point.

```
00295 : latitude(lat), longitude(lon) {}
```

11.3.3 Member Data Documentation

11.3.3.1 latitude

```
coord sacfmt::point::latitude {}
```

Latitude of point.

```
00286 {};
```

11.3.3.2 longitude

```
coord sacfmt::point::longitude {}
```

Longitude of point.

```
00287 {};
```

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

- include/sac-format/sac_format.hpp

11.4 sacfmt::read_spec Struct Reference

Struct that specifies parameters for reading.

```
#include <sac_format.hpp>
```

Public Attributes

- size_t num_words {}

  Number of words to read.
- size_t start_word {}

  Word to start reading from.

11.4.1 Detailed Description

Struct that specifies parameters for reading.

Prevents bug-prone number-swapping in functions that use a reading specification.

11.4.2 Member Data Documentation

11.4.2.1 num_words

```
size_t sacfmt::read_spec::num_words {}
```

Number of words to read.

```
00211 {};
```

11.4.2.2 start_word

```
size_t sacfmt::read_spec::start_word {}
```

Word to start reading from.

```
00213 {};
```

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

- include/sac-format/sac_format.hpp

11.5 sacfmt::Trace Class Reference

The Trace class.

```
#include <sac_format.hpp>
```

Public Member Functions

- Trace () noexcept

  Trace default constructor.
- Trace (const std::filesystem::path &path)

  Binary SAC-file reader.
- void write (const std::filesystem::path &path, bool legacy=false) const

  Binary SAC-file writer.
- void legacy_write (const std::filesystem::path &path) const

  Binary SAC-file legacy-write convenience function.
- bool operator== (const Trace &other) const noexcept

  Trace equality operator.
- void calc_geometry () noexcept

  Calculates gcarc, dist, az, and baz from stla, stlo, evla, and evlo.
- double frequency () const noexcept

  Calculate frequency from delta.
- std::string date () const noexcept

  Get date string.
- std::string time () const noexcept

  Get time string.
- float depmin () const noexcept
- float depmax () const noexcept
- float odelta () const noexcept
- float resp0 () const noexcept
- float resp1 () const noexcept
- float resp2 () const noexcept
- float resp3 () const noexcept
- float resp4 () const noexcept
- float resp5 () const noexcept
- float resp6 () const noexcept
- float resp7 () const noexcept
- float resp8 () const noexcept
- float resp9 () const noexcept
- float stel () const noexcept
- float stdp () const noexcept
- float evel () const noexcept
- float evdp () const noexcept
- float mag () const noexcept
- float user0 () const noexcept
- float user1 () const noexcept
- float user2 () const noexcept
- float user3 () const noexcept
- float user4 () const noexcept
- float user5 () const noexcept
- float user6 () const noexcept
- float user7 () const noexcept
- float user8 () const noexcept
- float user9 () const noexcept
- float dist () const noexcept
- float az () const noexcept
- float baz () const noexcept
- float gcarc () const noexcept
- float depmen () const noexcept

- float cmpaz () const noexcept
- float cmpinc () const noexcept
- float xminimum () const noexcept
- float xmaximum () const noexcept
- float yminimum () const noexcept
- float ymaximum () const noexcept
- double delta () const noexcept
- double b () const noexcept
- double e () const noexcept
- double o () const noexcept
- double a () const noexcept
- double t0 () const noexcept
- double t1 () const noexcept
- double t2 () const noexcept
- double t3 () const noexcept
- double t4 () const noexcept
- double t5 () const noexcept
- double t6 () const noexcept
- double t7 () const noexcept
- double t8 () const noexcept
- double t9 () const noexcept
- double f () const noexcept
- double stla () const noexcept
- double stlo () const noexcept
- double evla () const noexcept
- double evlo () const noexcept
- double sb () const noexcept
- double sdelta () const noexcept
- int nzyear () const noexcept
- int nzjday () const noexcept
- int nzhour () const noexcept
- int nzmin () const noexcept
- int nzsec () const noexcept
- int nzmsec () const noexcept
- int nvhdr () const noexcept
- int norid () const noexcept
- int nevid () const noexcept
- int npts () const noexcept
- int nsnpts () const noexcept
- int nwfid () const noexcept
- int nxsize () const noexcept
- int nysize () const noexcept
- int iftype () const noexcept
- int idep () const noexcept
- int iztype () const noexcept
- int iinst () const noexcept
- int istreg () const noexcept
- int ievreg () const noexcept
- int ievtyp () const noexcept
- int iqual () const noexcept
- int isynth () const noexcept
- int imagtyp () const noexcept
- int imagsrc () const noexcept
- int ibody () const noexcept
- bool leven () const noexcept

- bool lpspol () const noexcept
- bool lovrok () const noexcept
- bool lcalda () const noexcept
- std::string kstnm () const noexcept
- std::string kevnm () const noexcept
- std::string khole () const noexcept
- std::string ko () const noexcept
- std::string ka () const noexcept
- std::string kt0 () const noexcept
- std::string kt1 () const noexcept
- std::string kt2 () const noexcept
- std::string kt3 () const noexcept
- std::string kt4 () const noexcept
- std::string kt5 () const noexcept
- std::string kt6 () const noexcept
- std::string kt7 () const noexcept
- std::string kt8 () const noexcept
- std::string kt9 () const noexcept
- std::string kf () const noexcept
- std::string kuser0 () const noexcept
- std::string kuser1 () const noexcept
- std::string kuser2 () const noexcept
- std::string kcmpnm () const noexcept
- std::string knetwk () const noexcept
- std::string kdatrd () const noexcept
- std::string kinst () const noexcept
- std::vector< double > data1 () const noexcept
- std::vector< double > data2 () const noexcept
- void depmin (float input) noexcept
- void depmax (float input) noexcept
- void odelta (float input) noexcept
- void resp0 (float input) noexcept
- void resp1 (float input) noexcept
- void resp2 (float input) noexcept
- void resp3 (float input) noexcept
- void resp4 (float input) noexcept
- void resp5 (float input) noexcept
- void resp6 (float input) noexcept
- void resp7 (float input) noexcept
- void resp8 (float input) noexcept
- void resp9 (float input) noexcept
- void stel (float input) noexcept
- void stdp (float input) noexcept
- void evel (float input) noexcept
- void evdp (float input) noexcept
- void mag (float input) noexcept
- void user0 (float input) noexcept
- void user1 (float input) noexcept
- void user2 (float input) noexcept
- void user3 (float input) noexcept
- void user4 (float input) noexcept
- void user5 (float input) noexcept
- void user6 (float input) noexcept
- void user7 (float input) noexcept
- void user8 (float input) noexcept

- void user9 (float input) noexcept
- void dist (float input) noexcept
- void az (float input) noexcept
- void baz (float input) noexcept
- void gcarc (float input) noexcept
- void depmen (float input) noexcept
- void cmpaz (float input) noexcept
- void cmpinc (float input) noexcept
- void xminimum (float input) noexcept
- void xmaximum (float input) noexcept
- void yminimum (float input) noexcept
- void ymaximum (float input) noexcept
- void delta (double input) noexcept
- void b (double input) noexcept
- void e (double input) noexcept
- void o (double input) noexcept
- void a (double input) noexcept
- void t0 (double input) noexcept
- void t1 (double input) noexcept
- void t2 (double input) noexcept
- void t3 (double input) noexcept
- void t4 (double input) noexcept
- void t5 (double input) noexcept
- void t6 (double input) noexcept
- void t7 (double input) noexcept
- void t8 (double input) noexcept
- void t9 (double input) noexcept
- void f (double input) noexcept
- void stla (double input) noexcept
- void stlo (double input) noexcept
- void evla (double input) noexcept
- void evlo (double input) noexcept
- void sb (double input) noexcept
- void sdelta (double input) noexcept
- void nzyear (int input) noexcept
- void nzjday (int input) noexcept
- void nzhour (int input) noexcept
- void nzmin (int input) noexcept
- void nzsec (int input) noexcept
- void nzmsec (int input) noexcept
- void nvhdr (int input) noexcept
- void norid (int input) noexcept
- void nevid (int input) noexcept
- void npts (int input) noexcept
- void nsnpts (int input) noexcept
- void nwfid (int input) noexcept
- void nxsize (int input) noexcept
- void nysize (int input) noexcept
- void iftype (int input) noexcept
- void idep (int input) noexcept
- void iztype (int input) noexcept
- void iinst (int input) noexcept
- void istreg (int input) noexcept
- void ievreg (int input) noexcept
- void ievtyp (int input) noexcept

- void iqual (int input) noexcept
- void isynth (int input) noexcept
- void imagtyp (int input) noexcept
- void imagsrc (int input) noexcept
- void ibody (int input) noexcept
- void leven (bool input) noexcept
- void lpspol (bool input) noexcept
- void lovrok (bool input) noexcept
- void lcalda (bool input) noexcept
- void kstnm (const std::string &input) noexcept
- void kevnm (const std::string &input) noexcept
- void khole (const std::string &input) noexcept
- void ko (const std::string &input) noexcept
- void ka (const std::string &input) noexcept
- void kt0 (const std::string &input) noexcept
- void kt1 (const std::string &input) noexcept
- void kt2 (const std::string &input) noexcept
- void kt3 (const std::string &input) noexcept
- void kt4 (const std::string &input) noexcept
- void kt5 (const std::string &input) noexcept
- void kt6 (const std::string &input) noexcept
- void kt7 (const std::string &input) noexcept
- void kt8 (const std::string &input) noexcept
- void kt9 (const std::string &input) noexcept
- void kf (const std::string &input) noexcept
- void kuser0 (const std::string &input) noexcept
- void kuser1 (const std::string &input) noexcept
- void kuser2 (const std::string &input) noexcept
- void kcmpnm (const std::string &input) noexcept
- void knetwk (const std::string &input) noexcept
- void kdatrd (const std::string &input) noexcept
- void kinst (const std::string &input) noexcept
- void data1 (const std::vector$<$ double $>$ &input) noexcept
- void data2 (const std::vector$<$ double $>$ &input) noexcept

Private Member Functions

- void calc_gcarc () noexcept

  Calculate great-circle arc-distance (gcarc).
- void calc_dist () noexcept

  Calculate distance (using gcarc).
- void calc_az () noexcept

  Calculate azimuth.
- void calc_baz () noexcept

  Calculate back-azimuth.
- bool geometry_set () const noexcept

  Determine if locations are set for geometry calculation.
- point station_location () const noexcept

  Return station location as a point.
- point event_location () const noexcept

  Return even location as a point.
- void resize_data1 (size_t size) noexcept
- void resize_data2 (size_t size) noexcept
- void resize_data (size_t size) noexcept

  Resize data vectors (only if eligible).

Private Attributes

- std::array< float, num_float > floats {}

  Float storage array.
- std::array< double, num_double > doubles {}

  Double storage array.
- std::array< int, num_int > ints {}

  Integer storage array.
- std::array< bool, num_bool > bools {}

  Boolean storage array.
- std::array< std::string, num_string > strings {}

  String storage array.
- std::array< std::vector< double >, num_data > data {}

  std::vector<double> storage array.

11.5.1 Detailed Description

The Trace class.

This class is the recommended way for reading/writing SAC-files.

It safely reads all data, provides automatic write support based upon the nVHdr header value (determine if a footer should be included or not).

It provides getters and setters for all SAC headers and the data.

11.5.2 Constructor & Destructor Documentation

11.5.2.1 Trace() [1/2]

```
sacfmt::Trace::Trace ( )  [noexcept]
```

Trace default constructor.

Fills all values with their default (unset) values. Data vectors are of size zero.

Returns

Default created Trace object.

```
                     {
  std::ranges::fill(floats.begin(), floats.end(), unset_float);
  std::ranges::fill(doubles.begin(), doubles.end(), unset_double);
  std::ranges::fill(ints.begin(), ints.end(), unset_int);
  std::ranges::fill(bools.begin(), bools.end(), unset_bool);
  std::ranges::fill(strings.begin(), strings.end(), unset_word);
}
```

11.5.2.2 Trace() [2/2]

```
sacfmt::Trace::Trace (
            const std::filesystem::path & path )  [explicit]
```

Binary SAC-file reader.

Parameters

in	*path*	std::filesystem::path SAC-file to be read.

Returns

Trace read in-file.

Exceptions

io_error	If the file is not safe to read for whatever reason.
std::exception	(disk failure).

```
                                        {
std::ifstream file(path, std::ifstream::binary);
if (!file) {
  throw io_error(path.string() + " cannot be opened to read.");
}
safe_to_read_header(&file);  // throws io_error if not safe
//--------------------------------------------------------------------------
// Header
delta(binary_to_float(read_word(&file)));
depmin(binary_to_float(read_word(&file)));
depmax(binary_to_float(read_word(&file)));
// Skip 'unused'
read_word(&file);
odelta(binary_to_float(read_word(&file)));
b(binary_to_float(read_word(&file)));
e(binary_to_float(read_word(&file)));
o(binary_to_float(read_word(&file)));
a(binary_to_float(read_word(&file)));
// Skip 'internal'
read_word(&file);
// T# pick headers
t0(binary_to_float(read_word(&file)));
t1(binary_to_float(read_word(&file)));
t2(binary_to_float(read_word(&file)));
t3(binary_to_float(read_word(&file)));
t4(binary_to_float(read_word(&file)));
t5(binary_to_float(read_word(&file)));
t6(binary_to_float(read_word(&file)));
t7(binary_to_float(read_word(&file)));
t8(binary_to_float(read_word(&file)));
t9(binary_to_float(read_word(&file)));
f(binary_to_float(read_word(&file)));
// Response headers
resp0(binary_to_float(read_word(&file)));
resp1(binary_to_float(read_word(&file)));
resp2(binary_to_float(read_word(&file)));
resp3(binary_to_float(read_word(&file)));
resp4(binary_to_float(read_word(&file)));
resp5(binary_to_float(read_word(&file)));
resp6(binary_to_float(read_word(&file)));
resp7(binary_to_float(read_word(&file)));
resp8(binary_to_float(read_word(&file)));
resp9(binary_to_float(read_word(&file)));
// Station headers
stla(binary_to_float(read_word(&file)));
stlo(binary_to_float(read_word(&file)));
stel(binary_to_float(read_word(&file)));
stdp(binary_to_float(read_word(&file)));
// Event headers
evla(binary_to_float(read_word(&file)));
evlo(binary_to_float(read_word(&file)));
evel(binary_to_float(read_word(&file)));
evdp(binary_to_float(read_word(&file)));
mag(binary_to_float(read_word(&file)));
// User misc headers
user0(binary_to_float(read_word(&file)));
user1(binary_to_float(read_word(&file)));
user2(binary_to_float(read_word(&file)));
user3(binary_to_float(read_word(&file)));
user4(binary_to_float(read_word(&file)));
user5(binary_to_float(read_word(&file)));
user6(binary_to_float(read_word(&file)));
user7(binary_to_float(read_word(&file)));
user8(binary_to_float(read_word(&file)));
user9(binary_to_float(read_word(&file)));
```

```
// Geometry headers
dist(binary_to_float(read_word(&file)));
az(binary_to_float(read_word(&file)));
baz(binary_to_float(read_word(&file)));
gcarc(binary_to_float(read_word(&file)));
// Metadata headers
sb(binary_to_float(read_word(&file)));
sdelta(binary_to_float(read_word(&file)));
depmen(binary_to_float(read_word(&file)));
cmpaz(binary_to_float(read_word(&file)));
cmpinc(binary_to_float(read_word(&file)));
xminimum(binary_to_float(read_word(&file)));
xmaximum(binary_to_float(read_word(&file)));
yminimum(binary_to_float(read_word(&file)));
ymaximum(binary_to_float(read_word(&file)));
// Skip 'unused' (xcommon_skip_num)
for (int i{0}; i < common_skip_num; ++i) {
  read_word(&file);
}
// Date/time headers
nzyear(binary_to_int(read_word(&file)));
nzjday(binary_to_int(read_word(&file)));
nzhour(binary_to_int(read_word(&file)));
nzmin(binary_to_int(read_word(&file)));
nzsec(binary_to_int(read_word(&file)));
nzmsec(binary_to_int(read_word(&file)));
// More metadata headers
nvhdr(binary_to_int(read_word(&file)));
norid(binary_to_int(read_word(&file)));
nevid(binary_to_int(read_word(&file)));
npts(binary_to_int(read_word(&file)));
nsnpts(binary_to_int(read_word(&file)));
nwfid(binary_to_int(read_word(&file)));
nxsize(binary_to_int(read_word(&file)));
nysize(binary_to_int(read_word(&file)));
// Skip 'unused'
read_word(&file);
iftype(binary_to_int(read_word(&file)));
idep(binary_to_int(read_word(&file)));
iztype(binary_to_int(read_word(&file)));
// Skip 'unused'
read_word(&file);
iinst(binary_to_int(read_word(&file)));
istreg(binary_to_int(read_word(&file)));
ievreg(binary_to_int(read_word(&file)));
ievtyp(binary_to_int(read_word(&file)));
iqual(binary_to_int(read_word(&file)));
isynth(binary_to_int(read_word(&file)));
imagtyp(binary_to_int(read_word(&file)));
imagsrc(binary_to_int(read_word(&file)));
ibody(binary_to_int(read_word(&file)));
// Skip 'unused' (xcommon_skip_num)
for (int i{0}; i < common_skip_num; ++i) {
  read_word(&file);
}
// Logical headers
leven(binary_to_bool(read_word(&file)));
lpspol(binary_to_bool(read_word(&file)));
lovrok(binary_to_bool(read_word(&file)));
lcalda(binary_to_bool(read_word(&file)));
// Skip 'unused'
read_word(&file);
// KSTNM is 2 words (normal)
kstnm(binary_to_string(read_two_words(&file)));
// KEVNM is 4 words long (unique!)
kevnm(binary_to_long_string(read_four_words(&file)));
// All other 'K' headers are 2 words
khole(binary_to_string(read_two_words(&file)));
ko(binary_to_string(read_two_words(&file)));
ka(binary_to_string(read_two_words(&file)));
kt0(binary_to_string(read_two_words(&file)));
kt1(binary_to_string(read_two_words(&file)));
kt2(binary_to_string(read_two_words(&file)));
kt3(binary_to_string(read_two_words(&file)));
kt4(binary_to_string(read_two_words(&file)));
kt5(binary_to_string(read_two_words(&file)));
kt6(binary_to_string(read_two_words(&file)));
kt7(binary_to_string(read_two_words(&file)));
kt8(binary_to_string(read_two_words(&file)));
kt9(binary_to_string(read_two_words(&file)));
kf(binary_to_string(read_two_words(&file)));
kuser0(binary_to_string(read_two_words(&file)));
kuser1(binary_to_string(read_two_words(&file)));
kuser2(binary_to_string(read_two_words(&file)));
kcmpnm(binary_to_string(read_two_words(&file)));
knetwk(binary_to_string(read_two_words(&file)));
kdatrd(binary_to_string(read_two_words(&file)));
```

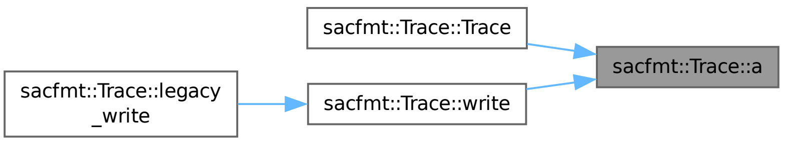
```
  kinst(binary_to_string(read_two_words(&file)));
  //--------------------------------------------------------------------------
  // DATA
  const bool is_data{npts() != unset_int};
  // data1
  const size_t n_words{static_cast<size_t>(npts())};
  if (is_data) {
    // false flags for data1
    safe_to_read_data(&file, n_words, false);  // throws io_error if unsafe
    const read_spec spec{n_words, data_word};
    // Originally floats, read as doubles
    data1(read_data(&file, spec));
  }
  // data2 (uneven or spectral data)
  if (is_data && (!leven() || (iftype() > 1))) {
    // true flags for data2
    safe_to_read_data(&file, n_words, true);  // throws io_error if unsafe
    const read_spec spec{n_words, data_word + static_cast<size_t>(npts())};
    data2(read_data(&file, spec));
  }
  //--------------------------------------------------------------------------
  // Footer
  if (nvhdr() == modern_hdr_version) {
    safe_to_read_footer(&file);  // throws io_error if not safe
    delta(binary_to_double(read_two_words(&file)));
    b(binary_to_double(read_two_words(&file)));
    e(binary_to_double(read_two_words(&file)));
    o(binary_to_double(read_two_words(&file)));
    a(binary_to_double(read_two_words(&file)));
    t0(binary_to_double(read_two_words(&file)));
    t1(binary_to_double(read_two_words(&file)));
    t2(binary_to_double(read_two_words(&file)));
    t3(binary_to_double(read_two_words(&file)));
    t4(binary_to_double(read_two_words(&file)));
    t5(binary_to_double(read_two_words(&file)));
    t6(binary_to_double(read_two_words(&file)));
    t7(binary_to_double(read_two_words(&file)));
    t8(binary_to_double(read_two_words(&file)));
    t9(binary_to_double(read_two_words(&file)));
    f(binary_to_double(read_two_words(&file)));
    evlo(binary_to_double(read_two_words(&file)));
    evla(binary_to_double(read_two_words(&file)));
    stlo(binary_to_double(read_two_words(&file)));
    stla(binary_to_double(read_two_words(&file)));
    sb(binary_to_double(read_two_words(&file)));
    sdelta(binary_to_double(read_two_words(&file)));
  }
  safe_to_finish_reading(&file);  // throws io_error if the file isn't finished
  file.close();
}
```

11.5.3 Member Function Documentation

11.5.3.1 a() [1/2]

```
double sacfmt::Trace::a ( ) const  [noexcept]
{ return doubles[sac_map.at(name::a)]; }
```

Here is the caller graph for this function:



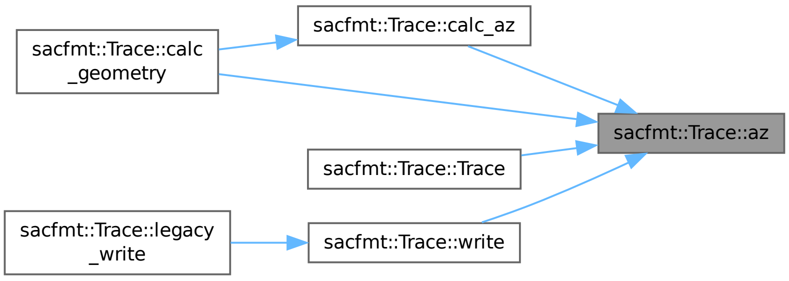
11.5.3.2 a() [2/2]

```
void sacfmt::Trace::a (
            double input ) [noexcept]
{
  doubles[sac_map.at(name::a)] = input;
}
```

11.5.3.3 az() [1/2]

```
float sacfmt::Trace::az ( ) const [noexcept]
{ return floats[sac_map.at(name::az)]; }
```

Here is the caller graph for this function:



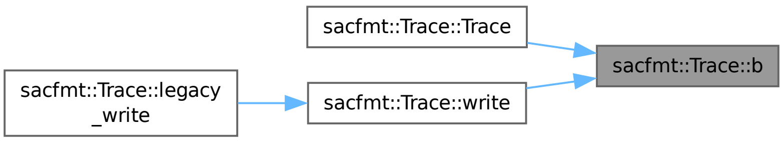
11.5.3.4 az() [2/2]

```
void sacfmt::Trace::az (
            float input ) [noexcept]
{
  floats[sac_map.at(name::az)] = input;
}
```

11.5.3.5 b() [1/2]

```
double sacfmt::Trace::b ( ) const [noexcept]
{ return doubles[sac_map.at(name::b)]; }
```

Here is the caller graph for this function:



11.5.3.6 b() [2/2]

```
void sacfmt::Trace::b (
            double input )  [noexcept]
```

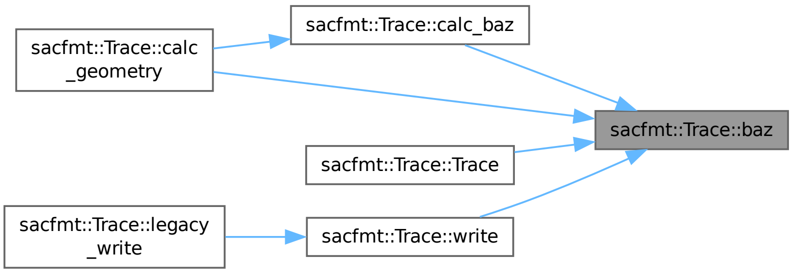
```
{
  doubles[sac_map.at(name::b)] = input;
}
```

11.5.3.7 baz() [1/2]

```
float sacfmt::Trace::baz ( ) const  [noexcept]
```

```
{ return floats[sac_map.at(name::baz)]; }
```

Here is the caller graph for this function:



11.5.3.8 baz() [2/2]

```
void sacfmt::Trace::baz (
            float input )  [noexcept]
```

```
{
  floats[sac_map.at(name::baz)] = input;
}
```

11.5.3.9 calc_az()

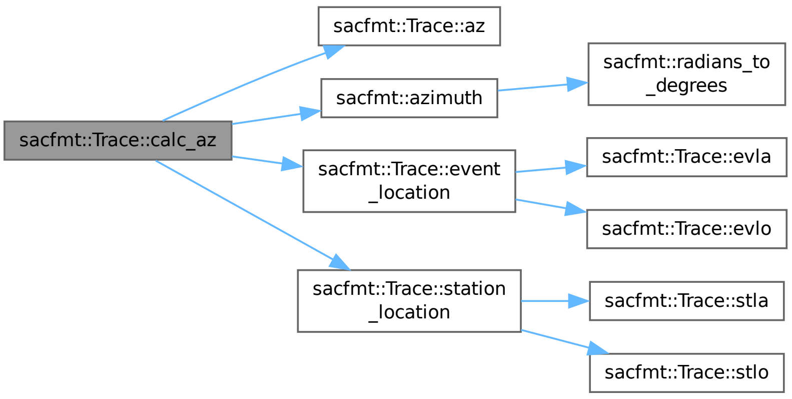
```
void sacfmt::Trace::calc_az ( )  [private], [noexcept]
```

Calculate azimuth.

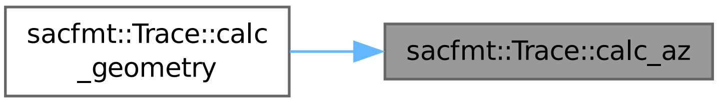
$$Station \rightarrow Event$$

```
{
  az(static_cast<float>(azimuth(event_location(), station_location())));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.5.3.10 calc_baz()

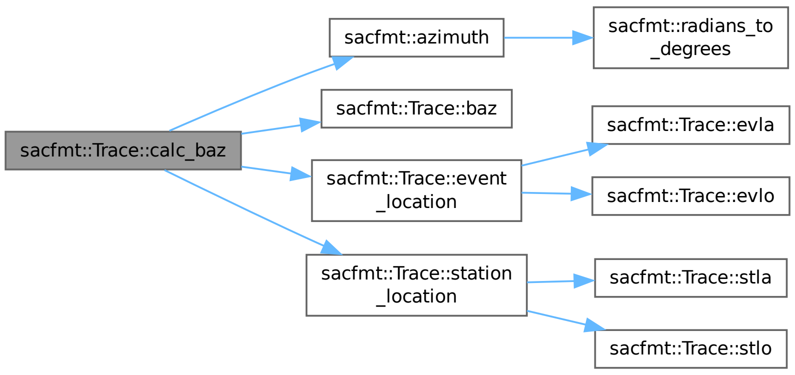
```
void sacfmt::Trace::calc_baz ( ) [private], [noexcept]
```

Calculate back-azimuth.

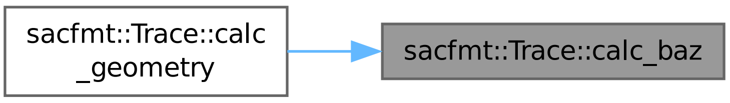
$$Event \rightarrow Station$$

```
{
  baz(static_cast<float>(azimuth(station_location(), event_location())));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.5.3.11 calc_dist()

```
void sacfmt::Trace::calc_dist ( )  [private], [noexcept]
```

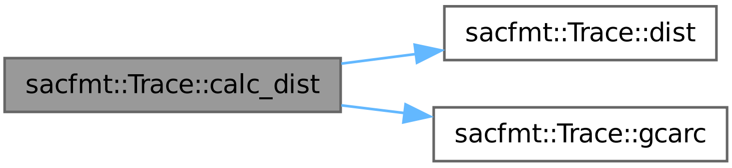
Calculate distance (using gcarc).

Assumes spherical Earth (in future may update to include flattening and different planteray bodies).

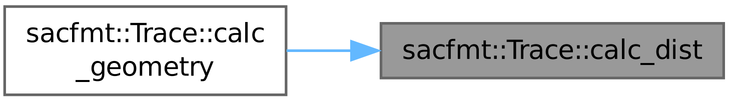
$$d = r_E \cdot \Delta$$

```
                               {
  dist(static_cast<float>(earth_radius * rad_per_deg * gcarc()));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



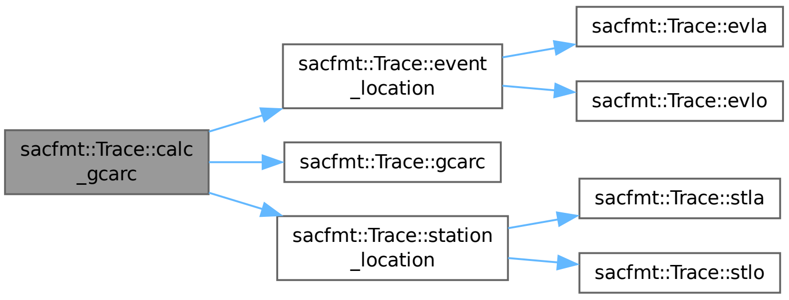
11.5.3.12 calc_gcarc()

```
void sacfmt::Trace::calc_gcarc ( )  [private], [noexcept]
```

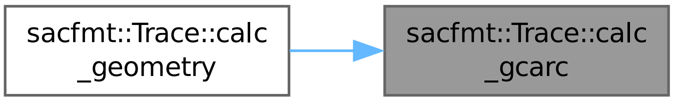
Calculate great-circle arc-distance (gcarc).

```
                               {
  Trace::gcarc(
      static_cast<float>(sacfmt::gcarc(station_location(), event_location())));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



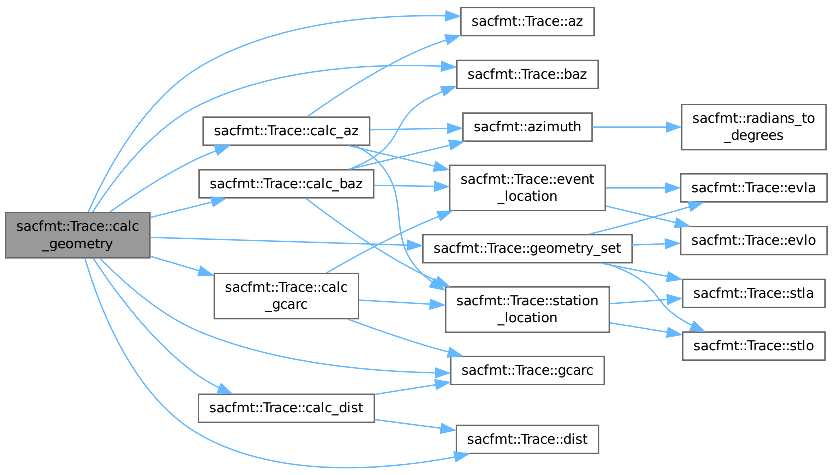
11.5.3.13 calc_geometry()

```
void sacfmt::Trace::calc_geometry ( )  [noexcept]
```

Calculates gcarc, dist, az, and baz from stla, stlo, evla, and evlo.

```
                                          {
  if (geometry_set()) {
    calc_gcarc();
    calc_dist();
    calc_az();
    calc_baz();
  } else {
    gcarc(unset_double);
    dist(unset_double);
    az(unset_double);
    baz(unset_double);
  }
}
```

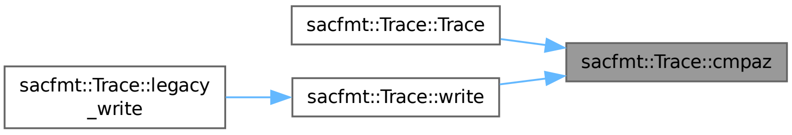
Here is the call graph for this function:



11.5.3.14 cmpaz() [1/2]

```
float sacfmt::Trace::cmpaz ( ) const  [noexcept]
{ return floats[sac_map.at(name::cmpaz)]; }
```

Here is the caller graph for this function:



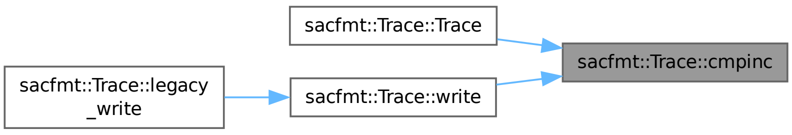
11.5.3.15 cmpaz() [2/2]

```
void sacfmt::Trace::cmpaz (
            float input )  [noexcept]
                                                {
  floats[sac_map.at(name::cmpaz)] = input;
}
```

11.5.3.16 cmpinc() [1/2]

```
float sacfmt::Trace::cmpinc ( ) const  [noexcept]
                                         {
  return floats[sac_map.at(name::cmpinc)];
}
```

Here is the caller graph for this function:



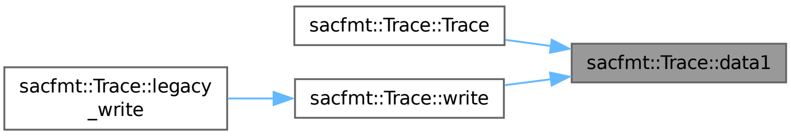
11.5.3.17 cmpinc() [2/2]

```
void sacfmt::Trace::cmpinc (
            float input )  [noexcept]
                                                 {
  floats[sac_map.at(name::cmpinc)] = input;
}
```

11.5.3.18 data1() [1/2]

```
std::vector< double > sacfmt::Trace::data1 ( ) const  [noexcept]
                                                  {
  return data[sac_map.at(name::data1)];
}
```

Here is the caller graph for this function:



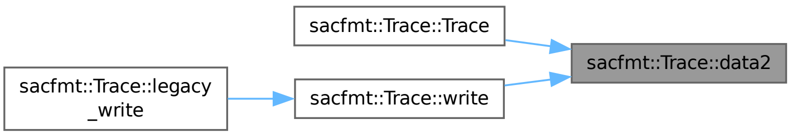
11.5.3.19 data1() [2/2]

```
void sacfmt::Trace::data1 (
            const std::vector< double > & input )  [noexcept]
                                                          {
  data[sac_map.at(name::data1)] = input;
  // Propagate change as needed
  int size{static_cast<int>(data1().size())};
  size = (((size == 0) && (npts() == unset_int)) ? unset_int : size);
  if (size != npts()) {
    npts(size);
  }
}
```

11.5.3.20 data2() [1/2]

```
std::vector< double > sacfmt::Trace::data2 ( ) const  [noexcept]
                                                  {
  return data[sac_map.at(name::data2)];
}
```

Here is the caller graph for this function:



11.5.3.21 data2() [2/2]

```
void sacfmt::Trace::data2 (
            const std::vector< double > & input ) [noexcept]
```

```
                                                      {
  data[sac_map.at(name::data2)] = input;
  // Proagate change as needed
  int size{static_cast<int>(data2().size())};
  size = (((size == 0) && (npts() == unset_int)) ? unset_int : size);
  // Need to make sure this is legal
  // If positive size and not-legal, make spectral
  if (size > 0) {
    // If not legal, make spectral
    if (leven() && (iftype() <= 1)) {
      iftype(2);
    }
    // If legal and different from npts, update npts
    if ((!leven() || (iftype() > 1)) && (size != npts())) {
      npts(size);
    }
  }
}
```

11.5.3.22 date()

```
std::string sacfmt::Trace::date ( ) const [noexcept]
```

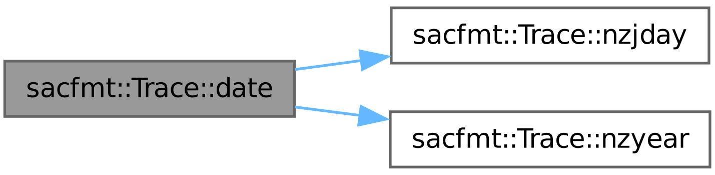
Get date string.

Returns

std::string Date (YYYY-JJJ).

```
                                    {
  // Require all to be set
  if ((nzyear() == unset_int) || (nzjday() == unset_int)) {
    return unset_word;
  }
  std::ostringstream oss{};
  oss « nzyear();
  oss « '-';
  oss « nzjday();
  return oss.str();
}
```

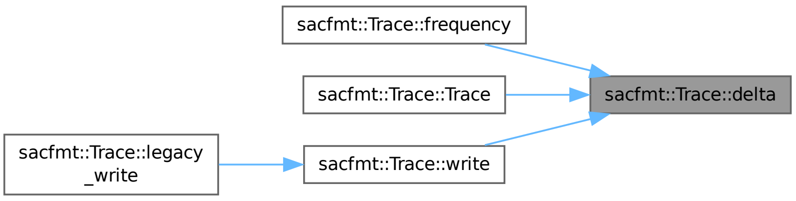
Here is the call graph for this function:



11.5.3.23 delta() [1/2]

```
double sacfmt::Trace::delta ( ) const  [noexcept]
                                      {
  return doubles[sac_map.at(name::delta)];
}
```

Here is the caller graph for this function:



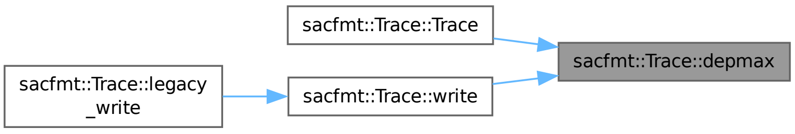
11.5.3.24 delta() [2/2]

```
void sacfmt::Trace::delta (
              double input )  [noexcept]
                                                {
  doubles[sac_map.at(name::delta)] = input;
}
```

11.5.3.25 depmax() [1/2]

```
float sacfmt::Trace::depmax ( ) const  [noexcept]
                                      {
  return floats[sac_map.at(name::depmax)];
}
```

Here is the caller graph for this function:



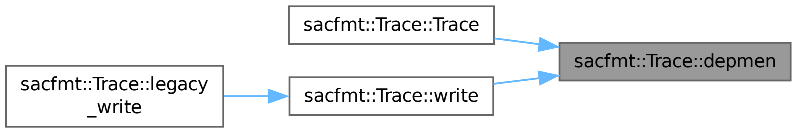
11.5.3.26 depmax() [2/2]

```
void sacfmt::Trace::depmax (
              float input )  [noexcept]
                                                {
  floats[sac_map.at(name::depmax)] = input;
}
```

11.5.3.27 depmen() [1/2]

```
float sacfmt::Trace::depmen ( ) const  [noexcept]
                                 {
  return floats[sac_map.at(name::depmen)];
}
```

Here is the caller graph for this function:



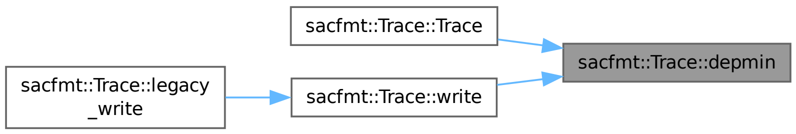
11.5.3.28 depmen() [2/2]

```
void sacfmt::Trace::depmen (
             float input )  [noexcept]
                                          {
  floats[sac_map.at(name::depmen)] = input;
}
```

11.5.3.29 depmin() [1/2]

```
float sacfmt::Trace::depmin ( ) const  [noexcept]
                                 {
  return floats[sac_map.at(name::depmin)];
}
```

Here is the caller graph for this function:



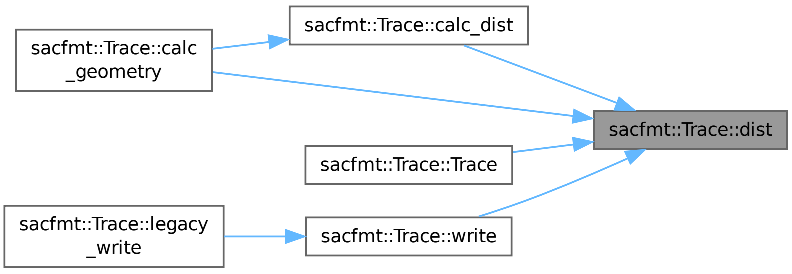
11.5.3.30 depmin() [2/2]

```
void sacfmt::Trace::depmin (
             float input )  [noexcept]
                                          {
  floats[sac_map.at(name::depmin)] = input;
}
```

11.5.3.31 dist() [1/2]

```
float sacfmt::Trace::dist ( ) const  [noexcept]
{ return floats[sac_map.at(name::dist)]; }
```

Here is the caller graph for this function:



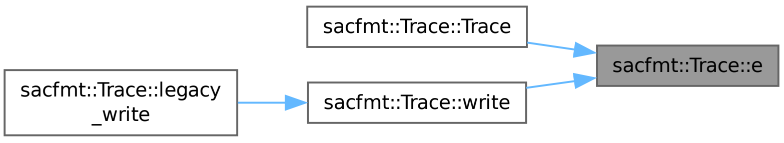
11.5.3.32 dist() [2/2]

```
void sacfmt::Trace::dist (
            float input )  [noexcept]
                                              {
  floats[sac_map.at(name::dist)] = input;
}
```

11.5.3.33 e() [1/2]

```
double sacfmt::Trace::e ( ) const  [noexcept]
{ return doubles[sac_map.at(name::e)]; }
```

Here is the caller graph for this function:



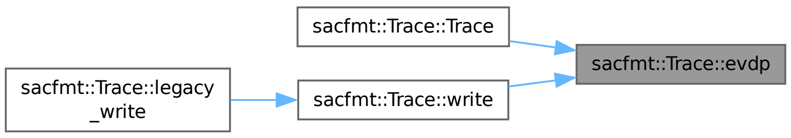
11.5.3.34 e() [2/2]

```
void sacfmt::Trace::e (
            double input )  [noexcept]
                                            {
  doubles[sac_map.at(name::e)] = input;
}
```

11.5.3.35 evdp() [1/2]

```
float sacfmt::Trace::evdp ( ) const  [noexcept]
{ return floats[sac_map.at(name::evdp)]; }
```

Here is the caller graph for this function:



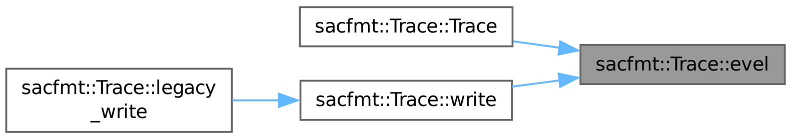
11.5.3.36 evdp() [2/2]

```
void sacfmt::Trace::evdp (
            float input )  [noexcept]
{
  floats[sac_map.at(name::evdp)] = input;
}
```

11.5.3.37 evel() [1/2]

```
float sacfmt::Trace::evel ( ) const  [noexcept]
{ return floats[sac_map.at(name::evel)]; }
```

Here is the caller graph for this function:



11.5.3.38 evel() [2/2]

```
void sacfmt::Trace::evel (
            float input )  [noexcept]
{
  floats[sac_map.at(name::evel)] = input;
}
```

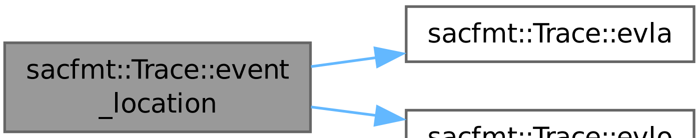
11.5.3.39 event_location()

```
point sacfmt::Trace::event_location ( ) const  [inline], [private], [noexcept]
```

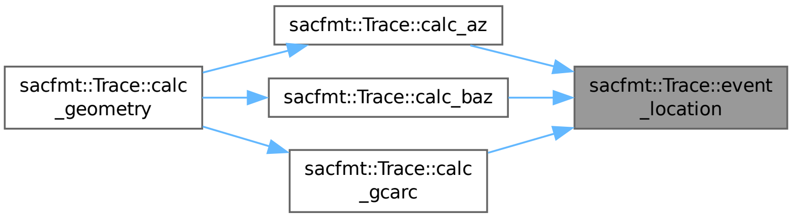
Return even location as a point.

```
                                                          {
      return point{coord{evla(), true}, coord{evlo(), true}};
  }
```

Here is the call graph for this function:



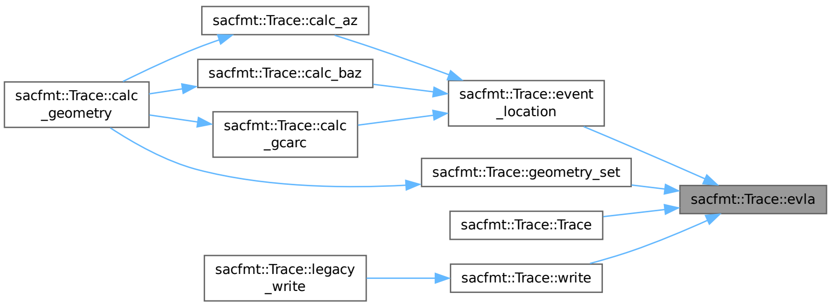
Here is the caller graph for this function:


sacfmt::Trace::calc _geometry
sacfmt::Trace::calc_az
sacfmt::Trace::calc_baz
sacfmt::Trace::calc _gcarc
sacfmt::Trace::event _location

11.5.3.40 evla() [1/2]

```
double sacfmt::Trace::evla ( ) const  [noexcept]
{ return doubles[sac_map.at(name::evla)]; }
```

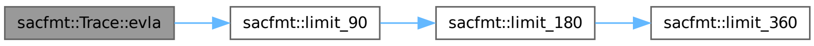
Here is the caller graph for this function:


sacfmt::Trace::calc_az
sacfmt::Trace::calc_baz
sacfmt::Trace::calc _geometry
sacfmt::Trace::calc _gcarc
sacfmt::Trace::event _location
sacfmt::Trace::geometry_set
sacfmt::Trace::Trace
sacfmt::Trace::evla
sacfmt::Trace::legacy _write
sacfmt::Trace::write

11.5.3.41 evla() [2/2]

```
void sacfmt::Trace::evla (
            double input )  [noexcept]
                                                     {
  double clean_input{input};
  if (clean_input != unset_double) {
    clean_input = limit_90(clean_input);
  }
  doubles[sac_map.at(name::evla)] = clean_input;
}
```

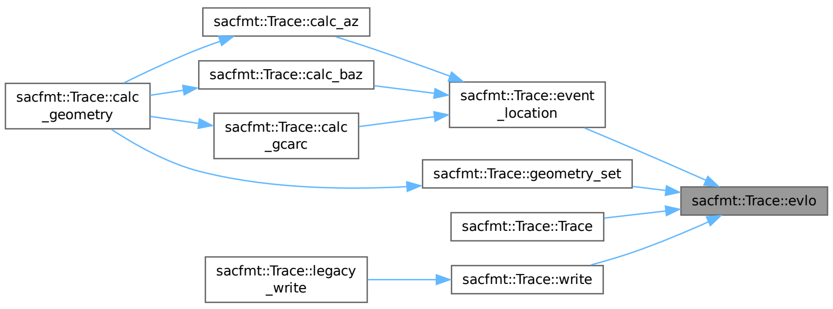
Here is the call graph for this function:



11.5.3.42 evlo() [1/2]

```
double sacfmt::Trace::evlo ( ) const  [noexcept]
{ return doubles[sac_map.at(name::evlo)]; }
```

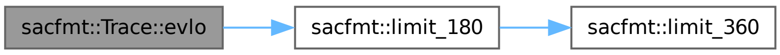
Here is the caller graph for this function:


sacfmt::Trace::calc_az
sacfmt::Trace::calc_baz
sacfmt::Trace::calc_geometry
sacfmt::Trace::calc_gcarc
sacfmt::Trace::event_location
sacfmt::Trace::geometry_set
sacfmt::Trace::evlo
sacfmt::Trace::Trace
sacfmt::Trace::legacy_write
sacfmt::Trace::write

11.5.3.43 evlo() [2/2]

```
void sacfmt::Trace::evlo (
            double input )  [noexcept]
                                                     {
  double clean_input{input};
  if (clean_input != unset_double) {
    clean_input = limit_180(clean_input);
  }
  doubles[sac_map.at(name::evlo)] = clean_input;
}
```

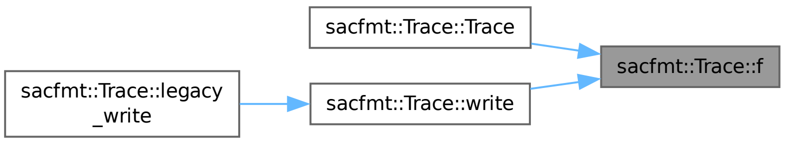
Here is the call graph for this function:



11.5.3.44 f() [1/2]

```
double sacfmt::Trace::f ( ) const  [noexcept]
{ return doubles[sac_map.at(name::f)]; }
```

Here is the caller graph for this function:



11.5.3.45 f() [2/2]

```
void sacfmt::Trace::f (
            double input )  [noexcept]
                                              {
    doubles[sac_map.at(name::f)] = input;
}
```

11.5.3.46 frequency()

```
double sacfmt::Trace::frequency ( ) const  [noexcept]
```

Calculate frequency from delta.

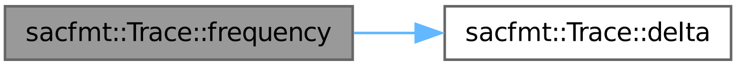
$$f = \frac{1}{\delta}$$

Returns

double Frequency.

```
                                            {
  const double delta_val{delta()};
  if ((delta_val == unset_double) || (delta_val <= 0)) {
    return unset_double;
  }
  return 1.0 / delta_val;
}
```

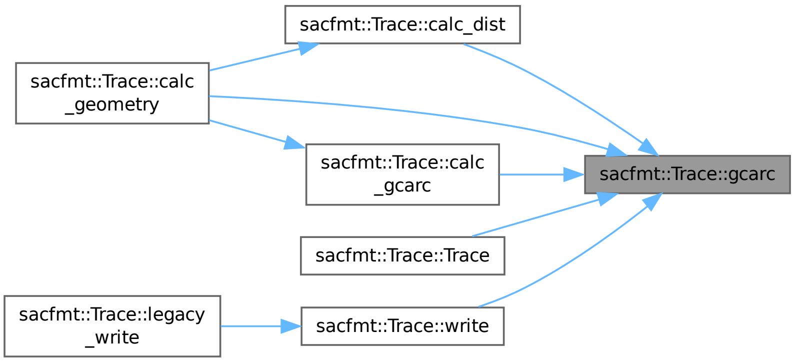
Here is the call graph for this function:



11.5.3.47 gcarc() [1/2]

```
float sacfmt::Trace::gcarc ( ) const  [noexcept]
{ return floats[sac_map.at(name::gcarc)]; }
```

Here is the caller graph for this function:



11.5.3.48 gcarc() [2/2]

```
void sacfmt::Trace::gcarc (
            float input )  [noexcept]
                                            {
  floats[sac_map.at(name::gcarc)] = input;
}
```

11.5.3.49 geometry_set()

```
bool sacfmt::Trace::geometry_set ( ) const  [private], [noexcept]
```

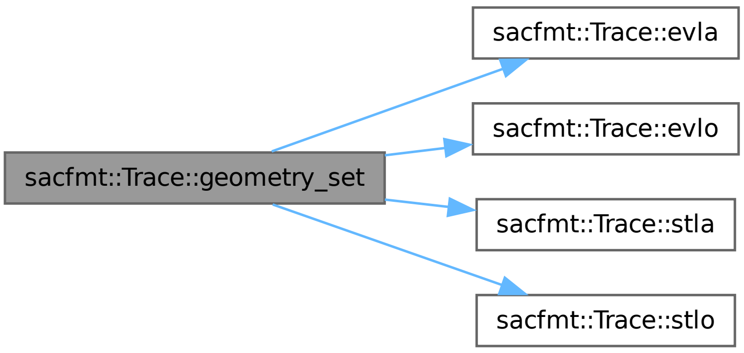
Determine if locations are set for geometry calculation.

Returns

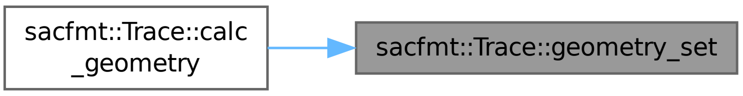
bool True if able to calculate geometry.

```
                                                   {
  return (stla() != unset_double) && (stlo() != unset_double) &&
         (evla() != unset_double) && (evlo() != unset_double);
}
```

Here is the call graph for this function:



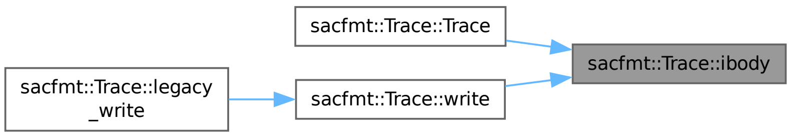
Here is the caller graph for this function:



11.5.3.50 ibody() [1/2]

```
int sacfmt::Trace::ibody ( ) const  [noexcept]
{ return ints[sac_map.at(name::ibody)]; }
```

Here is the caller graph for this function:



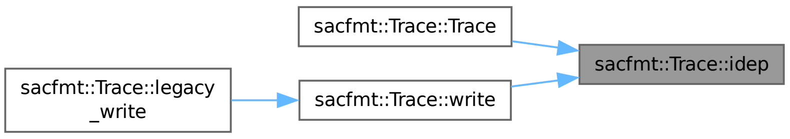
11.5.3.51 ibody() [2/2]

```
void sacfmt::Trace::ibody (
            int input )  [noexcept]
{
  ints[sac_map.at(name::ibody)] = input;
}
```

11.5.3.52 idep() [1/2]

```
int sacfmt::Trace::idep ( ) const  [noexcept]
{ return ints[sac_map.at(name::idep)]; }
```

Here is the caller graph for this function:



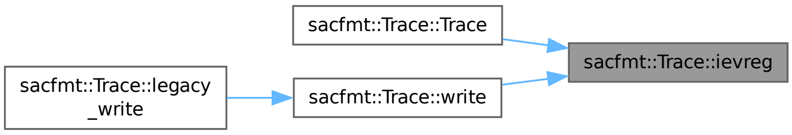
11.5.3.53 idep() [2/2]

```
void sacfmt::Trace::idep (
            int input )  [noexcept]
{
  ints[sac_map.at(name::idep)] = input;
}
```

11.5.3.54 ievreg() [1/2]

```
int sacfmt::Trace::ievreg ( ) const  [noexcept]
{ return ints[sac_map.at(name::ievreg)]; }
```

Here is the caller graph for this function:



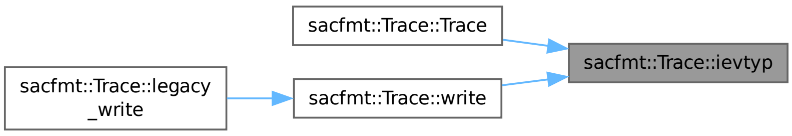
11.5.3.55 ievreg() [2/2]

```
void sacfmt::Trace::ievreg (
            int input )  [noexcept]
                                                 {
  ints[sac_map.at(name::ievreg)] = input;
}
```

11.5.3.56 ievtyp() [1/2]

```
int sacfmt::Trace::ievtyp ( ) const  [noexcept]
{ return ints[sac_map.at(name::ievtyp)]; }
```

Here is the caller graph for this function:



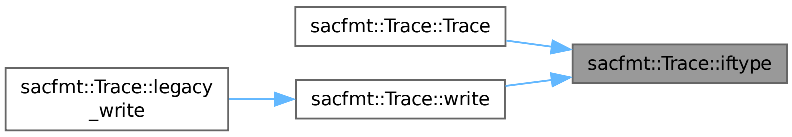
11.5.3.57 ievtyp() [2/2]

```
void sacfmt::Trace::ievtyp (
            int input )  [noexcept]
                                                 {
  ints[sac_map.at(name::ievtyp)] = input;
}
```

11.5.3.58 iftype() [1/2]

```
int sacfmt::Trace::iftype ( ) const  [noexcept]
{ return ints[sac_map.at(name::iftype)]; }
```

Here is the caller graph for this function:



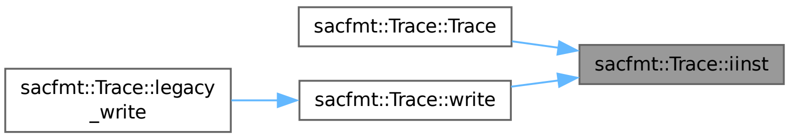
11.5.3.59 iftype() [2/2]

```
void sacfmt::Trace::iftype (
            int input )  [noexcept]
                                                     {
  ints[sac_map.at(name::iftype)] = input;
  const size_t size{npts() >= 0 ? static_cast<size_t>(npts()) : 0};
  // Uneven 2D data not supported as not in specification
  if ((input > 1) && !leven()) {
    leven(true);
  }
  resize_data2(size);
}
```

11.5.3.60 iinst() [1/2]

```
int sacfmt::Trace::iinst ( ) const  [noexcept]
{ return ints[sac_map.at(name::iinst)]; }
```

Here is the caller graph for this function:



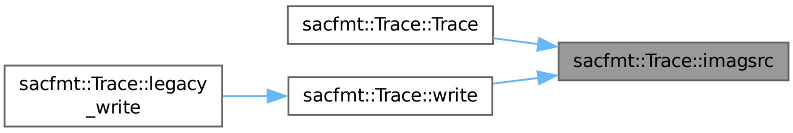
11.5.3.61 iinst() [2/2]

```
void sacfmt::Trace::iinst (
            int input )  [noexcept]
                                                    {
  ints[sac_map.at(name::iinst)] = input;
}
```

11.5.3.62 imagsrc() [1/2]

```
int sacfmt::Trace::imagsrc ( ) const  [noexcept]
{ return ints[sac_map.at(name::imagsrc)]; }
```

Here is the caller graph for this function:



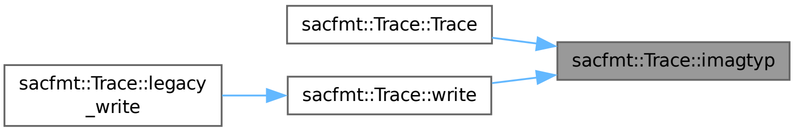
11.5.3.63 imagsrc() [2/2]

```
void sacfmt::Trace::imagsrc (
             int input )  [noexcept]
                                              {
  ints[sac_map.at(name::imagsrc)] = input;
}
```

11.5.3.64 imagtyp() [1/2]

```
int sacfmt::Trace::imagtyp ( ) const  [noexcept]
{ return ints[sac_map.at(name::imagtyp)]; }
```

Here is the caller graph for this function:



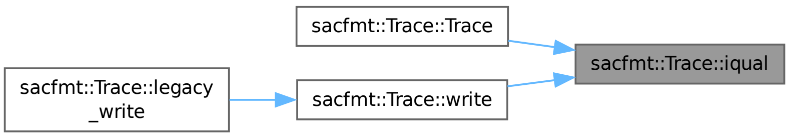
11.5.3.65 imagtyp() [2/2]

```
void sacfmt::Trace::imagtyp (
             int input )  [noexcept]
                                              {
  ints[sac_map.at(name::imagtyp)] = input;
}
```

11.5.3.66 iqual() [1/2]

```
int sacfmt::Trace::iqual ( ) const  [noexcept]
{ return ints[sac_map.at(name::iqual)]; }
```

Here is the caller graph for this function:



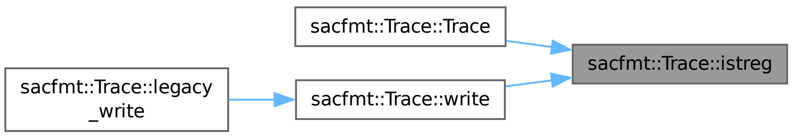
11.5.3.67 iqual() [2/2]

```
void sacfmt::Trace::iqual (
            int input )  [noexcept]
                                            {
  ints[sac_map.at(name::iqual)] = input;
}
```

11.5.3.68 istreg() [1/2]

```
int sacfmt::Trace::istreg ( ) const  [noexcept]
{ return ints[sac_map.at(name::istreg)]; }
```

Here is the caller graph for this function:



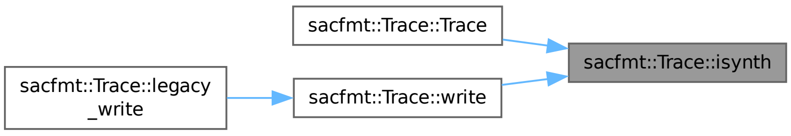
11.5.3.69 istreg() [2/2]

```
void sacfmt::Trace::istreg (
            int input )  [noexcept]
                                             {
  ints[sac_map.at(name::istreg)] = input;
}
```

11.5.3.70 isynth() [1/2]

```
int sacfmt::Trace::isynth ( ) const  [noexcept]
{ return ints[sac_map.at(name::isynth)]; }
```

Here is the caller graph for this function:



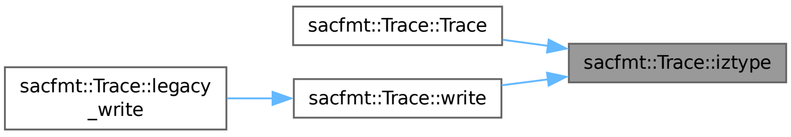
11.5.3.71 isynth() [2/2]

```
void sacfmt::Trace::isynth (
            int input )  [noexcept]
                                                    {
  ints[sac_map.at(name::isynth)] = input;
}
```

11.5.3.72 iztype() [1/2]

```
int sacfmt::Trace::iztype ( ) const  [noexcept]
{ return ints[sac_map.at(name::iztype)]; }
```

Here is the caller graph for this function:



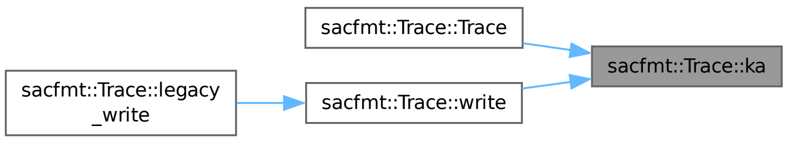
11.5.3.73 iztype() [2/2]

```
void sacfmt::Trace::iztype (
            int input )  [noexcept]
                                                    {
  ints[sac_map.at(name::iztype)] = input;
}
```

11.5.3.74 ka() [1/2]

```
std::string sacfmt::Trace::ka ( ) const  [noexcept]
{ return strings[sac_map.at(name::ka)]; }
```

Here is the caller graph for this function:



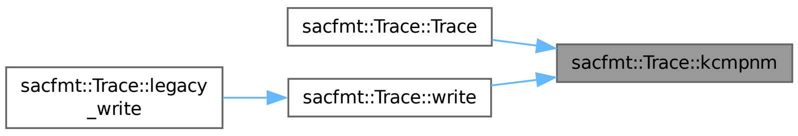
11.5.3.75 ka() [2/2]

```
void sacfmt::Trace::ka (
            const std::string & input )  [noexcept]
                                              {
  strings[sac_map.at(name::ka)] = input;
}
```

11.5.3.76 kcmpnm() [1/2]

```
std::string sacfmt::Trace::kcmpnm ( ) const  [noexcept]
                                        {
  return strings[sac_map.at(name::kcmpnm)];
}
```

Here is the caller graph for this function:



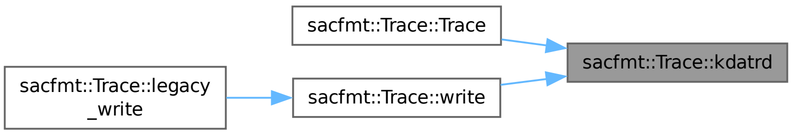
11.5.3.77 kcmpnm() [2/2]

```
void sacfmt::Trace::kcmpnm (
            const std::string & input )  [noexcept]
                                                 {
  strings[sac_map.at(name::kcmpnm)] = input;
}
```

11.5.3.78 kdatrd() [1/2]

```
std::string sacfmt::Trace::kdatrd ( ) const  [noexcept]
                                        {
  return strings[sac_map.at(name::kdatrd)];
}
```

Here is the caller graph for this function:



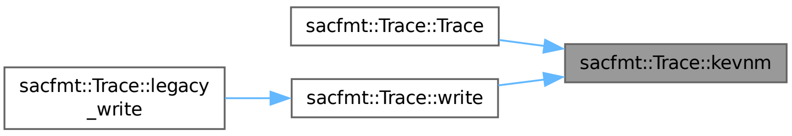
11.5.3.79 kdatrd() [2/2]

```
void sacfmt::Trace::kdatrd (
            const std::string & input )  [noexcept]
                                                     {
  strings[sac_map.at(name::kdatrd)] = input;
}
```

11.5.3.80 kevnm() [1/2]

```
std::string sacfmt::Trace::kevnm ( ) const  [noexcept]
                                       {
  return strings[sac_map.at(name::kevnm)];
}
```

Here is the caller graph for this function:



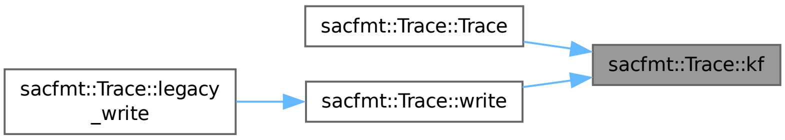
11.5.3.81 kevnm() [2/2]

```
void sacfmt::Trace::kevnm (
            const std::string & input )  [noexcept]
                                                    {
  strings[sac_map.at(name::kevnm)] = input;
}
```

11.5.3.82 kf() [1/2]

```
std::string sacfmt::Trace::kf ( ) const  [noexcept]
{ return strings[sac_map.at(name::kf)]; }
```

Here is the caller graph for this function:



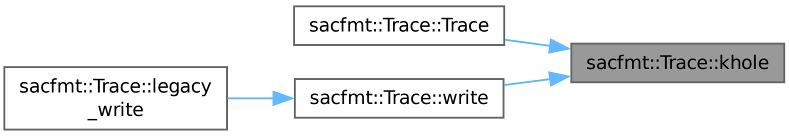
11.5.3.83 kf() [2/2]

```
void sacfmt::Trace::kf (
            const std::string & input )  [noexcept]
                                                    {
  strings[sac_map.at(name::kf)] = input;
}
```

11.5.3.84 khole() [1/2]

```
std::string sacfmt::Trace::khole ( ) const  [noexcept]
                                             {
  return strings[sac_map.at(name::khole)];
}
```

Here is the caller graph for this function:



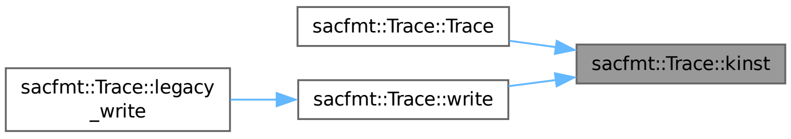
11.5.3.85 khole() [2/2]

```
void sacfmt::Trace::khole (
            const std::string & input )  [noexcept]
                                                       {
  strings[sac_map.at(name::khole)] = input;
}
```

11.5.3.86 kinst() [1/2]

```
std::string sacfmt::Trace::kinst ( ) const [noexcept]
{
  return strings[sac_map.at(name::kinst)];
}
```

Here is the caller graph for this function:



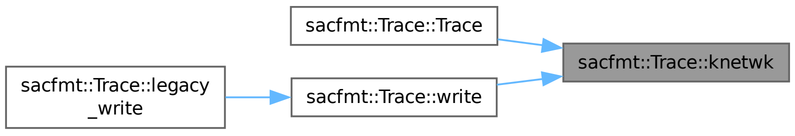
11.5.3.87 kinst() [2/2]

```
void sacfmt::Trace::kinst (
            const std::string & input ) [noexcept]
{
  strings[sac_map.at(name::kinst)] = input;
}
```

11.5.3.88 knetwk() [1/2]

```
std::string sacfmt::Trace::knetwk ( ) const [noexcept]
{
  return strings[sac_map.at(name::knetwk)];
}
```

Here is the caller graph for this function:



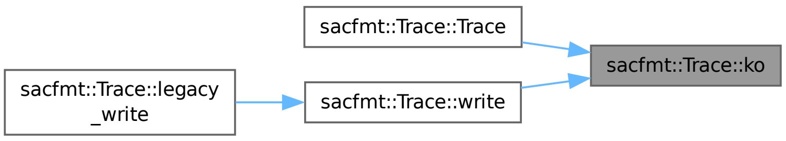
11.5.3.89 knetwk() [2/2]

```
void sacfmt::Trace::knetwk (
            const std::string & input ) [noexcept]
{
  strings[sac_map.at(name::knetwk)] = input;
}
```

11.5.3.90 ko() [1/2]

```
std::string sacfmt::Trace::ko ( ) const  [noexcept]
{ return strings[sac_map.at(name::ko)]; }
```

Here is the caller graph for this function:



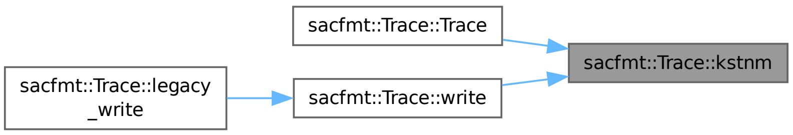
11.5.3.91 ko() [2/2]

```
void sacfmt::Trace::ko (
            const std::string & input )  [noexcept]
                                             {
  strings[sac_map.at(name::ko)] = input;
}
```

11.5.3.92 kstnm() [1/2]

```
std::string sacfmt::Trace::kstnm ( ) const  [noexcept]
                                      {
  return strings[sac_map.at(name::kstnm)];
}
```

Here is the caller graph for this function:



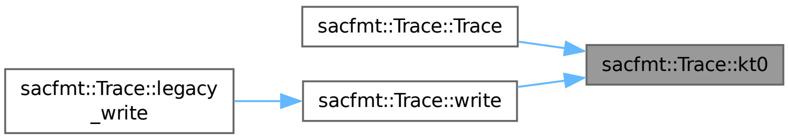
11.5.3.93 kstnm() [2/2]

```
void sacfmt::Trace::kstnm (
            const std::string & input )  [noexcept]
                                                {
  strings[sac_map.at(name::kstnm)] = input;
}
```

11.5.3.94 kt0() [1/2]

```
std::string sacfmt::Trace::kt0 ( ) const  [noexcept]
                                    {
  return strings[sac_map.at(name::kt0)];
}
```

Here is the caller graph for this function:



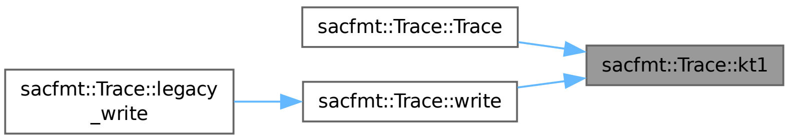
11.5.3.95 kt0() [2/2]

```
void sacfmt::Trace::kt0 (
             const std::string & input )  [noexcept]
                                              {
  strings[sac_map.at(name::kt0)] = input;
}
```

11.5.3.96 kt1() [1/2]

```
std::string sacfmt::Trace::kt1 ( ) const  [noexcept]
                                    {
  return strings[sac_map.at(name::kt1)];
}
```

Here is the caller graph for this function:



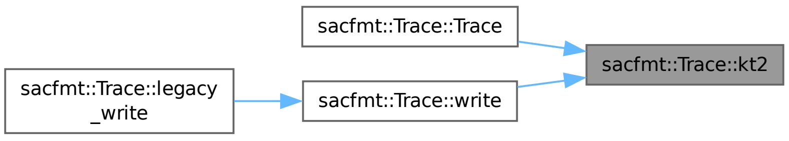
11.5.3.97 kt1() [2/2]

```
void sacfmt::Trace::kt1 (
             const std::string & input )  [noexcept]
                                              {
  strings[sac_map.at(name::kt1)] = input;
}
```

11.5.3.98 kt2() [1/2]

```
std::string sacfmt::Trace::kt2 ( ) const  [noexcept]
                                             {
  return strings[sac_map.at(name::kt2)];
}
```

Here is the caller graph for this function:



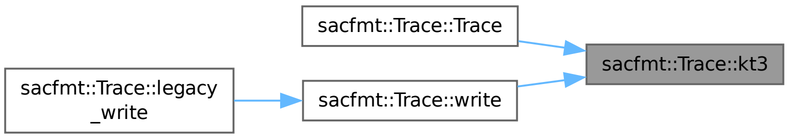
11.5.3.99 kt2() [2/2]

```
void sacfmt::Trace::kt2 (
            const std::string & input )  [noexcept]
                                                          {
  strings[sac_map.at(name::kt2)] = input;
}
```

11.5.3.100 kt3() [1/2]

```
std::string sacfmt::Trace::kt3 ( ) const  [noexcept]
                                             {
  return strings[sac_map.at(name::kt3)];
}
```

Here is the caller graph for this function:



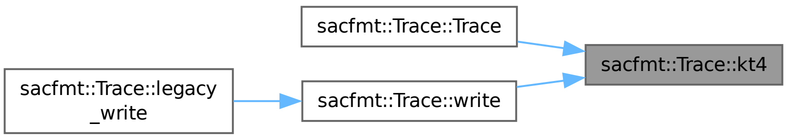
11.5.3.101 kt3() [2/2]

```
void sacfmt::Trace::kt3 (
            const std::string & input )  [noexcept]
                                                          {
  strings[sac_map.at(name::kt3)] = input;
}
```

11.5.3.102 kt4() [1/2]

```
std::string sacfmt::Trace::kt4 ( ) const  [noexcept]
                                     {
  return strings[sac_map.at(name::kt4)];
}
```

Here is the caller graph for this function:



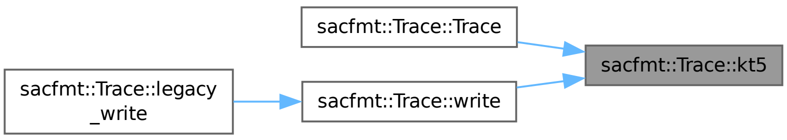
11.5.3.103 kt4() [2/2]

```
void sacfmt::Trace::kt4 (
             const std::string & input )  [noexcept]
                                              {
  strings[sac_map.at(name::kt4)] = input;
}
```

11.5.3.104 kt5() [1/2]

```
std::string sacfmt::Trace::kt5 ( ) const  [noexcept]
                                     {
  return strings[sac_map.at(name::kt5)];
}
```

Here is the caller graph for this function:



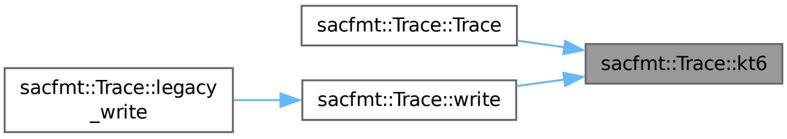
11.5.3.105 kt5() [2/2]

```
void sacfmt::Trace::kt5 (
             const std::string & input )  [noexcept]
                                              {
  strings[sac_map.at(name::kt5)] = input;
}
```

11.5.3.106 kt6() [1/2]

```
std::string sacfmt::Trace::kt6 ( ) const  [noexcept]
                                        {
  return strings[sac_map.at(name::kt6)];
}
```

Here is the caller graph for this function:



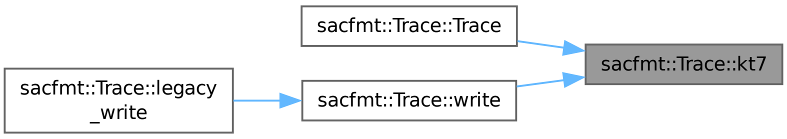
11.5.3.107 kt6() [2/2]

```
void sacfmt::Trace::kt6 (
             const std::string & input )  [noexcept]
                                                   {
  strings[sac_map.at(name::kt6)] = input;
}
```

11.5.3.108 kt7() [1/2]

```
std::string sacfmt::Trace::kt7 ( ) const  [noexcept]
                                        {
  return strings[sac_map.at(name::kt7)];
}
```

Here is the caller graph for this function:



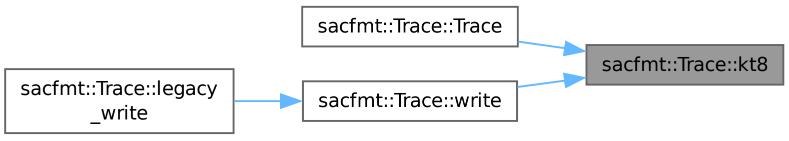
11.5.3.109 kt7() [2/2]

```
void sacfmt::Trace::kt7 (
             const std::string & input )  [noexcept]
                                                   {
  strings[sac_map.at(name::kt7)] = input;
}
```

11.5.3.110 kt8() [1/2]

```
std::string sacfmt::Trace::kt8 ( ) const  [noexcept]
                                    {
  return strings[sac_map.at(name::kt8)];
}
```

Here is the caller graph for this function:



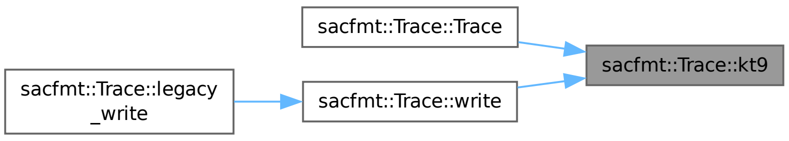
11.5.3.111 kt8() [2/2]

```
void sacfmt::Trace::kt8 (
            const std::string & input )  [noexcept]
                                             {
  strings[sac_map.at(name::kt8)] = input;
}
```

11.5.3.112 kt9() [1/2]

```
std::string sacfmt::Trace::kt9 ( ) const  [noexcept]
                                    {
  return strings[sac_map.at(name::kt9)];
}
```

Here is the caller graph for this function:



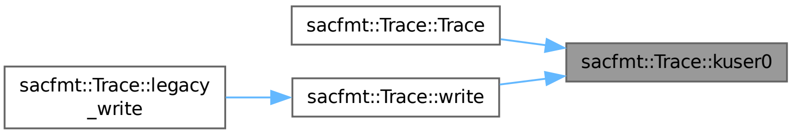
11.5.3.113 kt9() [2/2]

```
void sacfmt::Trace::kt9 (
            const std::string & input )  [noexcept]
                                             {
  strings[sac_map.at(name::kt9)] = input;
}
```

11.5.3.114 kuser0() [1/2]

```
std::string sacfmt::Trace::kuser0 ( ) const [noexcept]
{
  return strings[sac_map.at(name::kuser0)];
}
```

Here is the caller graph for this function:



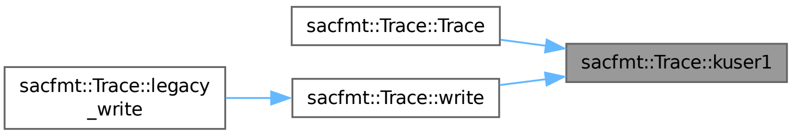
11.5.3.115 kuser0() [2/2]

```
void sacfmt::Trace::kuser0 (
            const std::string & input ) [noexcept]
{
  strings[sac_map.at(name::kuser0)] = input;
}
```

11.5.3.116 kuser1() [1/2]

```
std::string sacfmt::Trace::kuser1 ( ) const [noexcept]
{
  return strings[sac_map.at(name::kuser1)];
}
```

Here is the caller graph for this function:



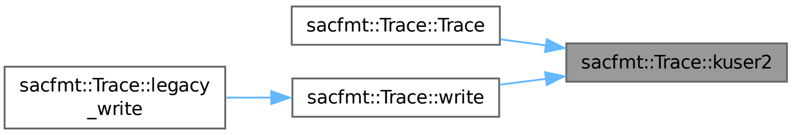
11.5.3.117 kuser1() [2/2]

```
void sacfmt::Trace::kuser1 (
            const std::string & input ) [noexcept]
{
  strings[sac_map.at(name::kuser1)] = input;
}
```

11.5.3.118 kuser2() [1/2]

```
std::string sacfmt::Trace::kuser2 ( ) const  [noexcept]
                                             {
  return strings[sac_map.at(name::kuser2)];
}
```

Here is the caller graph for this function:



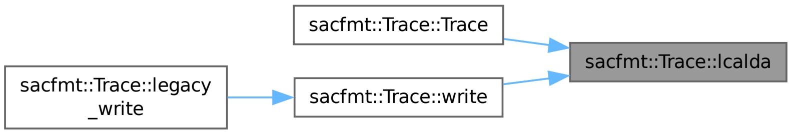
11.5.3.119 kuser2() [2/2]

```
void sacfmt::Trace::kuser2 (
            const std::string & input )  [noexcept]
                                                   {
  strings[sac_map.at(name::kuser2)] = input;
}
```

11.5.3.120 lcalda() [1/2]

```
bool sacfmt::Trace::lcalda ( ) const  [noexcept]
{ return bools[sac_map.at(name::lcalda)]; }
```

Here is the caller graph for this function:



11.5.3.121 lcalda() [2/2]

```
void sacfmt::Trace::lcalda (
            bool input )  [noexcept]
                                             {
  bools[sac_map.at(name::lcalda)] = input;
}
```

11.5.3.122 legacy_write()

```
void sacfmt::Trace::legacy_write (
            const std::filesystem::path & path ) const
```

Binary SAC-file legacy-write convenience function.

Parameters

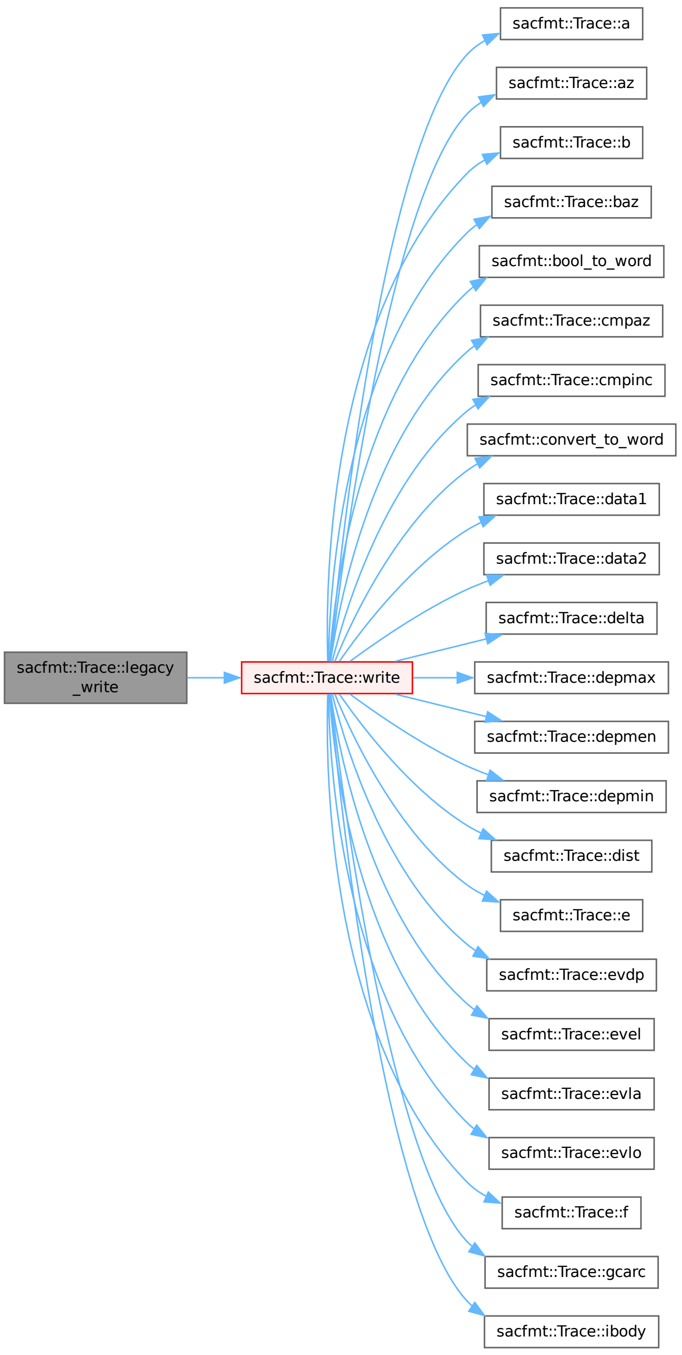
in	*path*	std::filesystem::path SAC-file to be written.

Exceptions

io_error	If the file cannot be written (bad path or bad permissions).
std::execption	Other unwritable issues (not enough space, disk failure, etc.).

```
                                                                          {
  write(path, true);
}
```

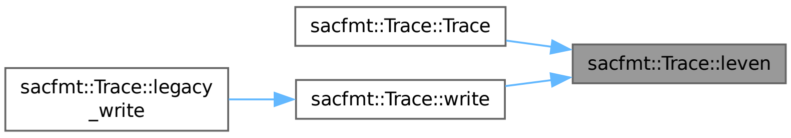
Here is the call graph for this function:



11.5.3.123 leven() [1/2]

```
bool sacfmt::Trace::leven ( ) const  [noexcept]
01141 { return bools[sac_map.at(name::leven)]; }
```

Here is the caller graph for this function:



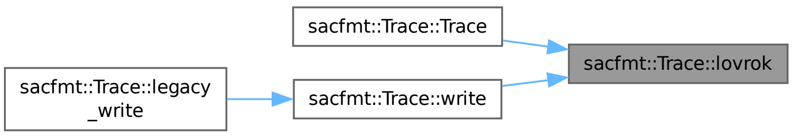
11.5.3.124 leven() [2/2]

```
void sacfmt::Trace::leven (
              bool input )  [noexcept]
                                              {
  bools[sac_map.at(name::leven)] = input;
  const size_t size{npts() >= 0 ? static_cast<size_t>(npts()) : 0};
  // Uneven 2D data not supported since not in specification
  if (!input && (iftype() > 1)) {
    iftype(unset_int);
  }
  resize_data2(size);
}
```

11.5.3.125 lovrok() [1/2]

```
bool sacfmt::Trace::lovrok ( ) const  [noexcept]
{ return bools[sac_map.at(name::lovrok)]; }
```

Here is the caller graph for this function:



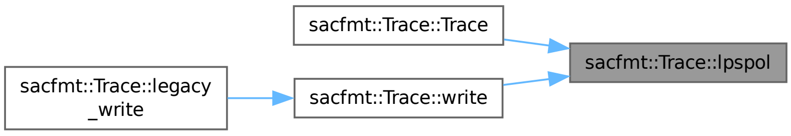
11.5.3.126 lovrok() [2/2]

```
void sacfmt::Trace::lovrok (
              bool input )  [noexcept]
                                              {
  bools[sac_map.at(name::lovrok)] = input;
}
```

11.5.3.127 lpspol() [1/2]

```
bool sacfmt::Trace::lpspol ( ) const [noexcept]
{ return bools[sac_map.at(name::lpspol)]; }
```

Here is the caller graph for this function:



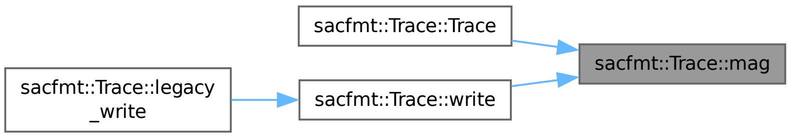
11.5.3.128 lpspol() [2/2]

```
void sacfmt::Trace::lpspol (
            bool input ) [noexcept]
                                         {
  bools[sac_map.at(name::lpspol)] = input;
}
```

11.5.3.129 mag() [1/2]

```
float sacfmt::Trace::mag ( ) const [noexcept]
{ return floats[sac_map.at(name::mag)]; }
```

Here is the caller graph for this function:



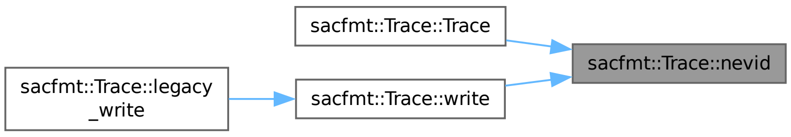
11.5.3.130 mag() [2/2]

```
void sacfmt::Trace::mag (
            float input ) [noexcept]
                                       {
  floats[sac_map.at(name::mag)] = input;
}
```

11.5.3.131 nevid() [1/2]

```
int sacfmt::Trace::nevid ( ) const  [noexcept]
{ return ints[sac_map.at(name::nevid)]; }
```

Here is the caller graph for this function:



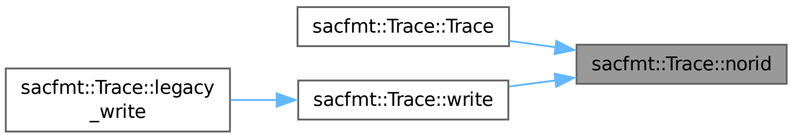
11.5.3.132 nevid() [2/2]

```
void sacfmt::Trace::nevid (
             int input )  [noexcept]
                                             {
  ints[sac_map.at(name::nevid)] = input;
}
```

11.5.3.133 norid() [1/2]

```
int sacfmt::Trace::norid ( ) const  [noexcept]
{ return ints[sac_map.at(name::norid)]; }
```

Here is the caller graph for this function:



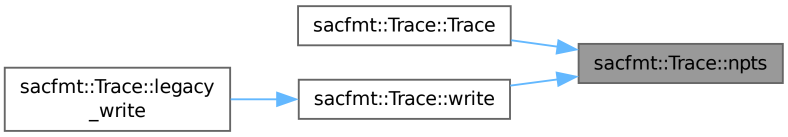
11.5.3.134 norid() [2/2]

```
void sacfmt::Trace::norid (
             int input )  [noexcept]
                                             {
  ints[sac_map.at(name::norid)] = input;
}
```

11.5.3.135 npts() [1/2]

```
int sacfmt::Trace::npts ( ) const  [noexcept]
{ return ints[sac_map.at(name::npts)]; }
```

Here is the caller graph for this function:



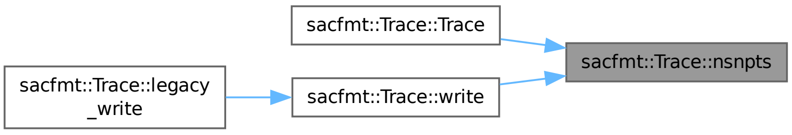
11.5.3.136 npts() [2/2]

```
void sacfmt::Trace::npts (
            int input )  [noexcept]
                                         {
  if ((input >= 0) || (input == unset_int)) {
    ints[sac_map.at(name::npts)] = input;
    const size_t size{static_cast<size_t>(input >= 0 ? input : 0)};
    resize_data(size);
  }
}
```

11.5.3.137 nsnpts() [1/2]

```
int sacfmt::Trace::nsnpts ( ) const  [noexcept]
{ return ints[sac_map.at(name::nsnpts)]; }
```

Here is the caller graph for this function:



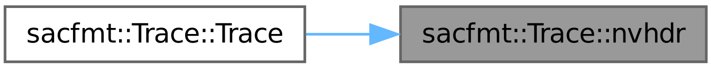
11.5.3.138 nsnpts() [2/2]

```
void sacfmt::Trace::nsnpts (
            int input )  [noexcept]
                                         {
  ints[sac_map.at(name::nsnpts)] = input;
}
```

11.5.3.139 nvhdr() [1/2]

```
int sacfmt::Trace::nvhdr ( ) const  [noexcept]
01120 { return ints[sac_map.at(name::nvhdr)]; }
```

Here is the caller graph for this function:



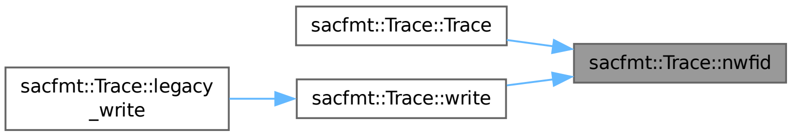
11.5.3.140 nvhdr() [2/2]

```
void sacfmt::Trace::nvhdr (
             int input )  [noexcept]
01437                                              {
01438   ints[sac_map.at(name::nvhdr)] = input;
01439 }
```

11.5.3.141 nwfid() [1/2]

```
int sacfmt::Trace::nwfid ( ) const  [noexcept]
01125 { return ints[sac_map.at(name::nwfid)]; }
```

Here is the caller graph for this function:



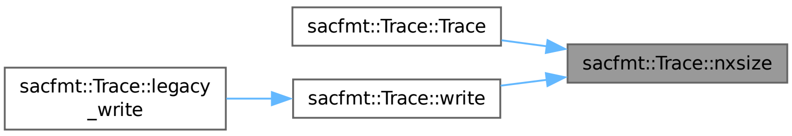
11.5.3.142 nwfid() [2/2]

```
void sacfmt::Trace::nwfid (
             int input )  [noexcept]
01456                                              {
01457   ints[sac_map.at(name::nwfid)] = input;
01458 }
```

11.5.3.143 nxsize() [1/2]

```
int sacfmt::Trace::nxsize ( ) const  [noexcept]
{ return ints[sac_map.at(name::nxsize)]; }
```

Here is the caller graph for this function:



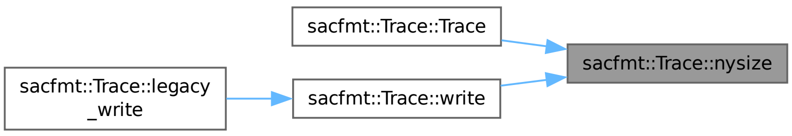
11.5.3.144 nxsize() [2/2]

```
void sacfmt::Trace::nxsize (
            int input )  [noexcept]
                                       {
  ints[sac_map.at(name::nxsize)] = input;
}
```

11.5.3.145 nysize() [1/2]

```
int sacfmt::Trace::nysize ( ) const  [noexcept]
{ return ints[sac_map.at(name::nysize)]; }
```

Here is the caller graph for this function:



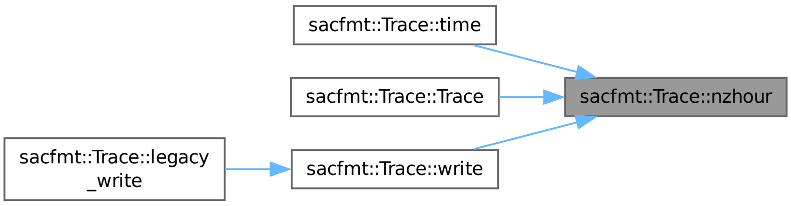
11.5.3.146 nysize() [2/2]

```
void sacfmt::Trace::nysize (
            int input )  [noexcept]
                                       {
  ints[sac_map.at(name::nysize)] = input;
}
```

11.5.3.147 nzhour() [1/2]

```
int sacfmt::Trace::nzhour ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzhour)]; }
```

Here is the caller graph for this function:



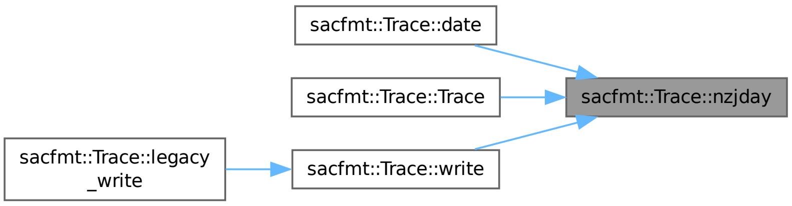
11.5.3.148 nzhour() [2/2]

```
void sacfmt::Trace::nzhour (
            int input )  [noexcept]
                                                 {
  ints[sac_map.at(name::nzhour)] = input;
}
```

11.5.3.149 nzjday() [1/2]

```
int sacfmt::Trace::nzjday ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzjday)]; }
```

Here is the caller graph for this function:



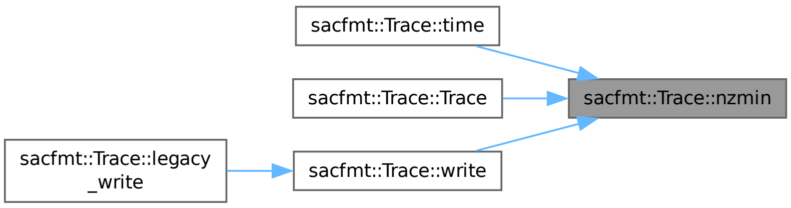
11.5.3.150 nzjday() [2/2]

```
void sacfmt::Trace::nzjday (
            int input )  [noexcept]
                                                 {
  ints[sac_map.at(name::nzjday)] = input;
}
```

11.5.3.151 nzmin() [1/2]

```
int sacfmt::Trace::nzmin ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzmin)]; }
```

Here is the caller graph for this function:



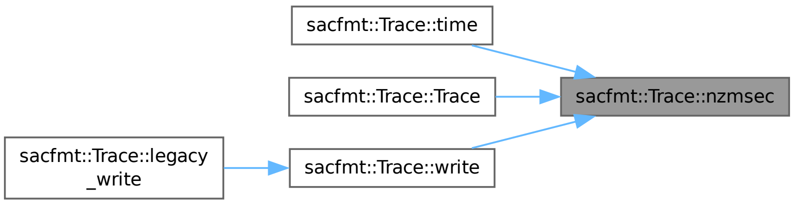
11.5.3.152 nzmin() [2/2]

```
void sacfmt::Trace::nzmin (
            int input )  [noexcept]
                                             {
  ints[sac_map.at(name::nzmin)] = input;
}
```

11.5.3.153 nzmsec() [1/2]

```
int sacfmt::Trace::nzmsec ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzmsec)]; }
```

Here is the caller graph for this function:



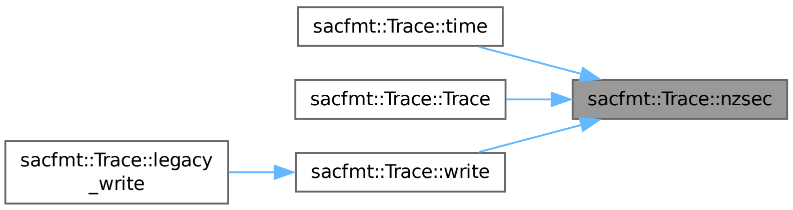
11.5.3.154 nzmsec() [2/2]

```
void sacfmt::Trace::nzmsec (
            int input )  [noexcept]
                                              {
  ints[sac_map.at(name::nzmsec)] = input;
}
```

11.5.3.155 nzsec() [1/2]

```
int sacfmt::Trace::nzsec ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzsec)]; }
```

Here is the caller graph for this function:



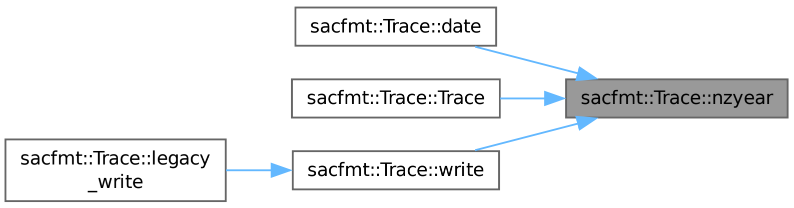
11.5.3.156 nzsec() [2/2]

```
void sacfmt::Trace::nzsec (
            int input )  [noexcept]
                                             {
  ints[sac_map.at(name::nzsec)] = input;
}
```

11.5.3.157 nzyear() [1/2]

```
int sacfmt::Trace::nzyear ( ) const  [noexcept]
{ return ints[sac_map.at(name::nzyear)]; }
```

Here is the caller graph for this function:



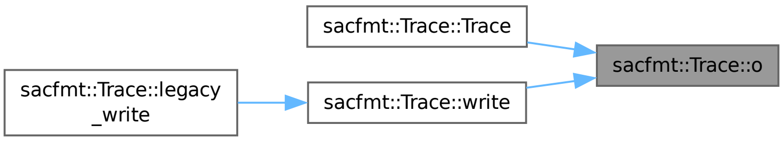
11.5.3.158 nzyear() [2/2]

```
void sacfmt::Trace::nzyear (
            int input )  [noexcept]
                                              {
  ints[sac_map.at(name::nzyear)] = input;
}
```

11.5.3.159 o() [1/2]

```
double sacfmt::Trace::o ( ) const  [noexcept]
{ return doubles[sac_map.at(name::o)]; }
```

Here is the caller graph for this function:



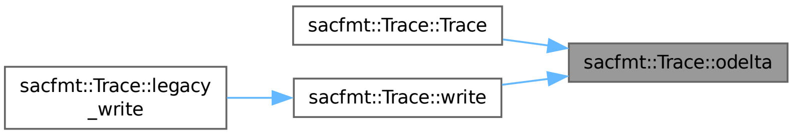
11.5.3.160 o() [2/2]

```
void sacfmt::Trace::o (
            double input )  [noexcept]
                                          {
  doubles[sac_map.at(name::o)] = input;
}
```

11.5.3.161 odelta() [1/2]

```
float sacfmt::Trace::odelta ( ) const  [noexcept]
                                     {
  return floats[sac_map.at(name::odelta)];
}
```

Here is the caller graph for this function:



11.5.3.162 odelta() [2/2]

```
void sacfmt::Trace::odelta (
            float input )  [noexcept]
                                             {
  floats[sac_map.at(name::odelta)] = input;
}
```

11.5.3.163 operator==()

```
bool sacfmt::Trace::operator== (
            const Trace & other ) const  [noexcept]
```

Trace equality operator.

Parameters

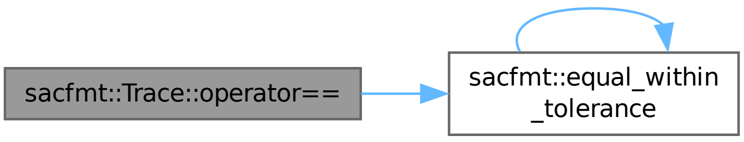
in	*this*	First Trace in comparison (LHS).
in	*other*	Second Trace in comparison (RHS).

Returns

bool Truth value of equality.

```
                                                    {
  if (floats != other.floats) {
    return false;
  }
  if (doubles != other.doubles) {
    return false;
  }
  if (ints != other.ints) {
    return false;
  }
  if (strings != other.strings) {
    return false;
  }
  if (!equal_within_tolerance(data[0], other.data[0])) {
    return false;
  }
  if (!equal_within_tolerance(data[1], other.data[1])) {
    return false;
  }
  return true;
}
```

Here is the call graph for this function:



11.5.3.164 resize_data()

```
void sacfmt::Trace::resize_data (
             size_t size )   [private], [noexcept]
```

Resize data vectors (only if eligible).

Will always resize data1, data2 only resizes if it can have non-zero size.

```
                                                  {
  resize_data1(size);
  resize_data2(size);
}
```

11.5.3.165 resize_data1()

```
void sacfmt::Trace::resize_data1 (
            size_t size )  [private], [noexcept]
```

```
{
  if (size != data1().size()) {
    std::vector<double> new_data1{data1()};
    new_data1.resize(size, 0.0);
    data1(new_data1);
  }
}
```

11.5.3.166 resize_data2()

```
void sacfmt::Trace::resize_data2 (
            size_t size )  [private], [noexcept]
```

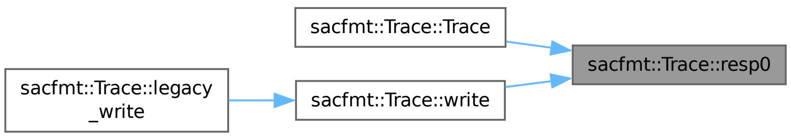
```
{
  // Data2 is legal
  if (!leven() || (iftype() > 1)) {
    if (size != data2().size()) {
      std::vector<double> new_data2{data2()};
      new_data2.resize(size, 0.0);
      data2(new_data2);
    }
  } else {
    if (!data2().empty()) {
      std::vector<double> new_data2{};
      data2(new_data2);
    }
  }
}
```

11.5.3.167 resp0() [1/2]

```
float sacfmt::Trace::resp0 ( ) const  [noexcept]
```

```
{ return floats[sac_map.at(name::resp0)]; }
```

Here is the caller graph for this function:



11.5.3.168 resp0() [2/2]

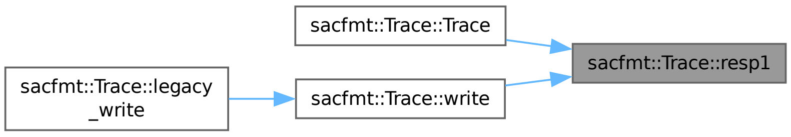
```
void sacfmt::Trace::resp0 (
            float input )  [noexcept]
```

```
{
  floats[sac_map.at(name::resp0)] = input;
}
```

11.5.3.169 resp1() [1/2]

```
float sacfmt::Trace::resp1 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp1)]; }
```

Here is the caller graph for this function:



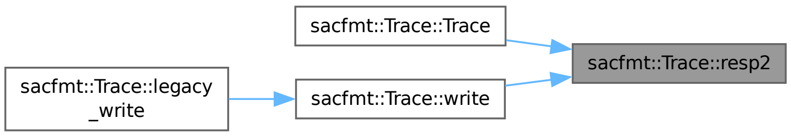
11.5.3.170 resp1() [2/2]

```
void sacfmt::Trace::resp1 (
            float input )  [noexcept]
                                                   {
  floats[sac_map.at(name::resp1)] = input;
}
```

11.5.3.171 resp2() [1/2]

```
float sacfmt::Trace::resp2 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp2)]; }
```

Here is the caller graph for this function:



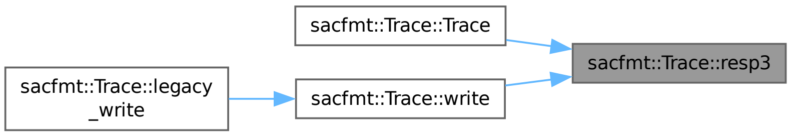
11.5.3.172 resp2() [2/2]

```
void sacfmt::Trace::resp2 (
            float input )  [noexcept]
                                                   {
  floats[sac_map.at(name::resp2)] = input;
}
```

11.5.3.173 resp3() [1/2]

```
float sacfmt::Trace::resp3 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp3)]; }
```

Here is the caller graph for this function:



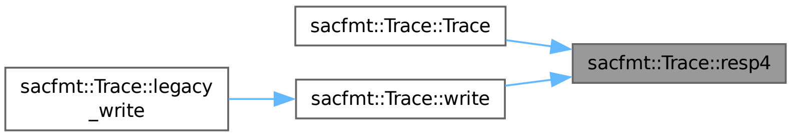
11.5.3.174 resp3() [2/2]

```
void sacfmt::Trace::resp3 (
            float input )  [noexcept]
                                                {
  floats[sac_map.at(name::resp3)] = input;
}
```

11.5.3.175 resp4() [1/2]

```
float sacfmt::Trace::resp4 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp4)]; }
```

Here is the caller graph for this function:



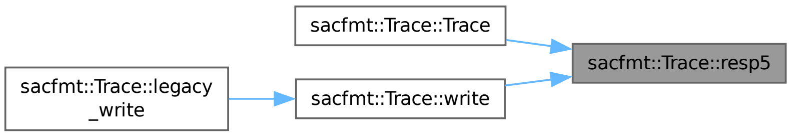
11.5.3.176 resp4() [2/2]

```
void sacfmt::Trace::resp4 (
            float input )  [noexcept]
                                                {
  floats[sac_map.at(name::resp4)] = input;
}
```

11.5.3.177 resp5() [1/2]

```
float sacfmt::Trace::resp5 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp5)]; }
```

Here is the caller graph for this function:



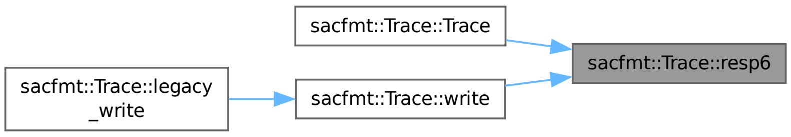
11.5.3.178 resp5() [2/2]

```
void sacfmt::Trace::resp5 (
            float input )  [noexcept]
                                         {
  floats[sac_map.at(name::resp5)] = input;
}
```

11.5.3.179 resp6() [1/2]

```
float sacfmt::Trace::resp6 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp6)]; }
```

Here is the caller graph for this function:



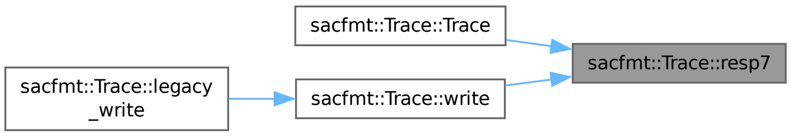
11.5.3.180 resp6() [2/2]

```
void sacfmt::Trace::resp6 (
            float input )  [noexcept]
                                         {
  floats[sac_map.at(name::resp6)] = input;
}
```

11.5.3.181 resp7() [1/2]

```
float sacfmt::Trace::resp7 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp7)]; }
```

Here is the caller graph for this function:



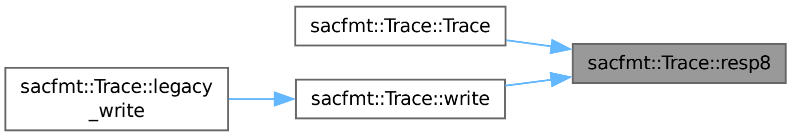
11.5.3.182 resp7() [2/2]

```
void sacfmt::Trace::resp7 (
            float input )  [noexcept]
                                                    {
  floats[sac_map.at(name::resp7)] = input;
}
```

11.5.3.183 resp8() [1/2]

```
float sacfmt::Trace::resp8 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp8)]; }
```

Here is the caller graph for this function:



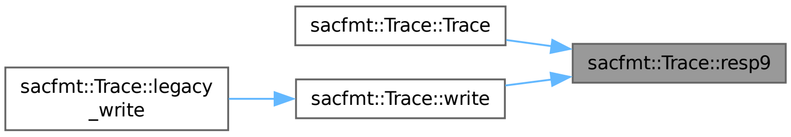
11.5.3.184 resp8() [2/2]

```
void sacfmt::Trace::resp8 (
            float input )  [noexcept]
                                                    {
  floats[sac_map.at(name::resp8)] = input;
}
```

11.5.3.185 resp9() [1/2]

```
float sacfmt::Trace::resp9 ( ) const  [noexcept]
{ return floats[sac_map.at(name::resp9)]; }
```

Here is the caller graph for this function:



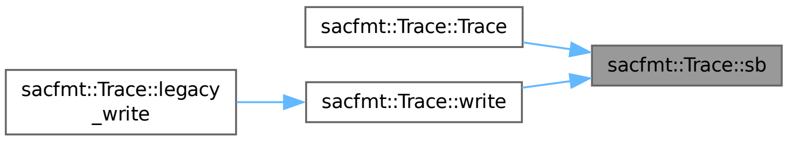
11.5.3.186 resp9() [2/2]

```
void sacfmt::Trace::resp9 (
            float input )  [noexcept]
                                         {
  floats[sac_map.at(name::resp9)] = input;
}
```

11.5.3.187 sb() [1/2]

```
double sacfmt::Trace::sb ( ) const  [noexcept]
{ return doubles[sac_map.at(name::sb)]; }
```

Here is the caller graph for this function:



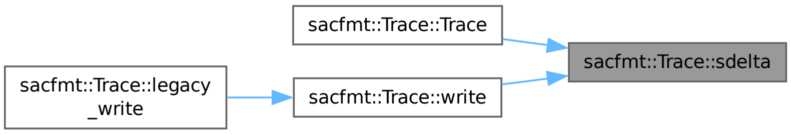
11.5.3.188 sb() [2/2]

```
void sacfmt::Trace::sb (
            double input )  [noexcept]
                                         {
  doubles[sac_map.at(name::sb)] = input;
}
```

11.5.3.189 sdelta() [1/2]

```
double sacfmt::Trace::sdelta ( ) const  [noexcept]
                                   {
  return doubles[sac_map.at(name::sdelta)];
}
```

Here is the caller graph for this function:



11.5.3.190 sdelta() [2/2]

```
void sacfmt::Trace::sdelta (
            double input )  [noexcept]
                                                {
  doubles[sac_map.at(name::sdelta)] = input;
}
```

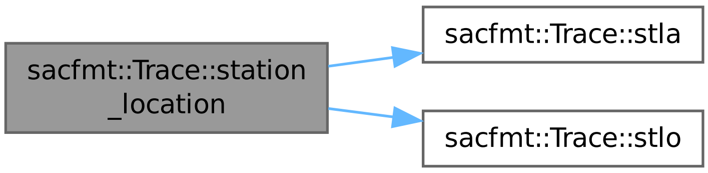
11.5.3.191 station_location()

```
point sacfmt::Trace::station_location ( ) const  [inline], [private], [noexcept]
```

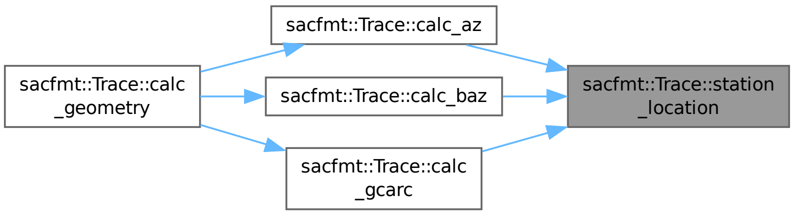
Return station location as a point.

```
                                                {
    return point{coord{stla(), true}, coord{stlo(), true}};
  }
```

Here is the call graph for this function:



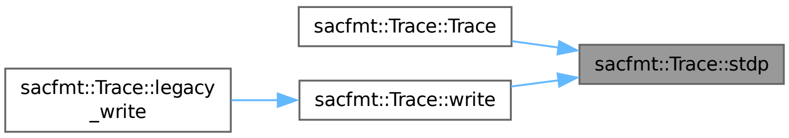
Here is the caller graph for this function:


sacfmt::Trace::calc_geometry
sacfmt::Trace::calc_az
sacfmt::Trace::calc_baz
sacfmt::Trace::calc_gcarc
sacfmt::Trace::station_location

11.5.3.192 stdp() [1/2]

```
float sacfmt::Trace::stdp ( ) const  [noexcept]
{ return floats[sac_map.at(name::stdp)]; }
```

Here is the caller graph for this function:



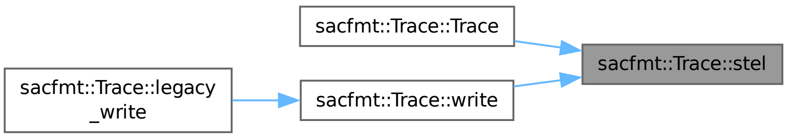
11.5.3.193 stdp() [2/2]

```
void sacfmt::Trace::stdp (
             float input )  [noexcept]
{
  floats[sac_map.at(name::stdp)] = input;
}
```

11.5.3.194 stel() [1/2]

```
float sacfmt::Trace::stel ( ) const  [noexcept]
{ return floats[sac_map.at(name::stel)]; }
```

Here is the caller graph for this function:



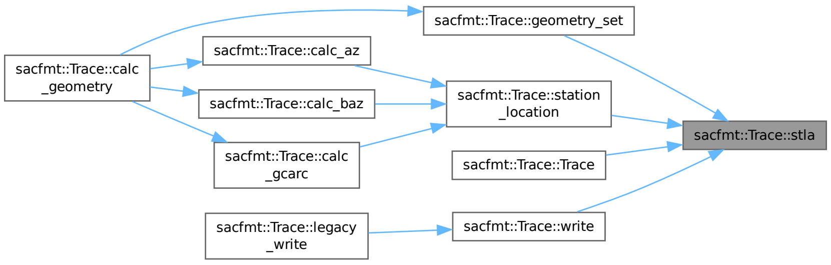
11.5.3.195 stel() [2/2]

```
void sacfmt::Trace::stel (
              float input )   [noexcept]
{
  floats[sac_map.at(name::stel)] = input;
}
```

11.5.3.196 stla() [1/2]

```
double sacfmt::Trace::stla ( ) const  [noexcept]
{ return doubles[sac_map.at(name::stla)]; }
```

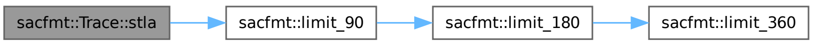
Here is the caller graph for this function:


sacfmt::Trace::geometry_set
sacfmt::Trace::calc_az
sacfmt::Trace::calc_geometry
sacfmt::Trace::calc_baz
sacfmt::Trace::station_location
sacfmt::Trace::stla
sacfmt::Trace::calc_gcarc
sacfmt::Trace::Trace
sacfmt::Trace::legacy_write
sacfmt::Trace::write

11.5.3.197 stla() [2/2]

```
void sacfmt::Trace::stla (
              double input )   [noexcept]
{
  double clean_input{input};
  if (clean_input != unset_double) {
    clean_input = limit_90(clean_input);
  }
  doubles[sac_map.at(name::stla)] = clean_input;
}
```

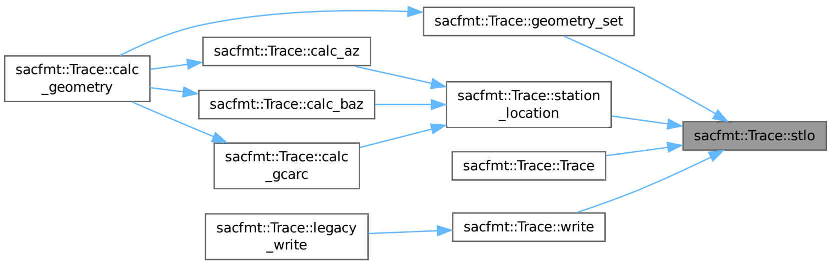
Here is the call graph for this function:



11.5.3.198 stlo() [1/2]

```
double sacfmt::Trace::stlo ( ) const  [noexcept]
{ return doubles[sac_map.at(name::stlo)]; }
```

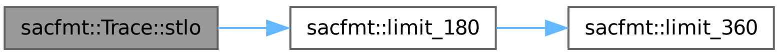
Here is the caller graph for this function:


sacfmt::Trace::geometry_set
sacfmt::Trace::calc_az
sacfmt::Trace::calc_geometry
sacfmt::Trace::calc_baz
sacfmt::Trace::station_location
sacfmt::Trace::stlo
sacfmt::Trace::calc_gcarc
sacfmt::Trace::Trace
sacfmt::Trace::legacy_write
sacfmt::Trace::write

11.5.3.199 stlo() [2/2]

```
void sacfmt::Trace::stlo (
            double input )  [noexcept]
                                         {
  double clean_input{input};
  if (clean_input != unset_double) {
    clean_input = limit_180(clean_input);
  }
  doubles[sac_map.at(name::stlo)] = clean_input;
}
```

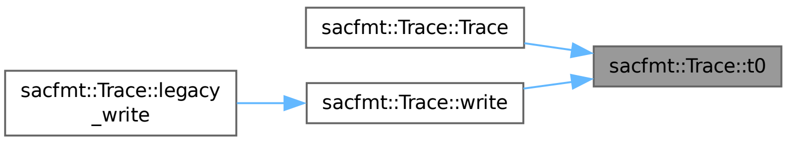
Here is the call graph for this function:



11.5.3.200 t0() [1/2]

```
double sacfmt::Trace::t0 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t0)]; }
```

Here is the caller graph for this function:



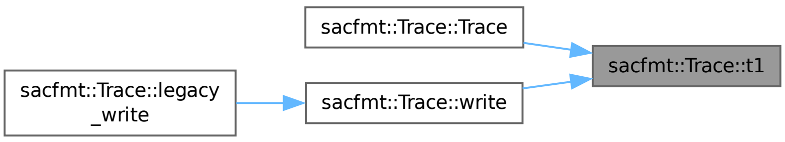
11.5.3.201 t0() [2/2]

```
void sacfmt::Trace::t0 (
            double input )  [noexcept]
                                      {
  doubles[sac_map.at(name::t0)] = input;
}
```

11.5.3.202 t1() [1/2]

```
double sacfmt::Trace::t1 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t1)]; }
```

Here is the caller graph for this function:



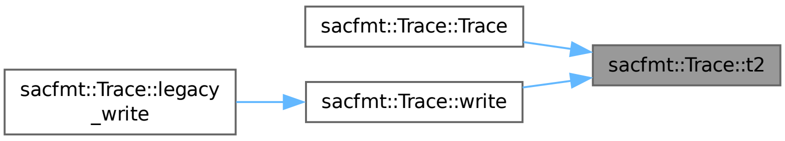
11.5.3.203 t1() [2/2]

```
void sacfmt::Trace::t1 (
            double input )  [noexcept]
                                      {
  doubles[sac_map.at(name::t1)] = input;
}
```

11.5.3.204 t2() [1/2]

```
double sacfmt::Trace::t2 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t2)]; }
```

Here is the caller graph for this function:



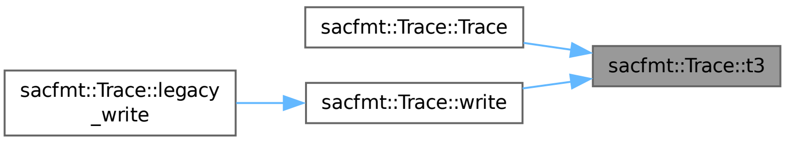
11.5.3.205 t2() [2/2]

```
void sacfmt::Trace::t2 (
            double input ) [noexcept]
{
  doubles[sac_map.at(name::t2)] = input;
}
```

11.5.3.206 t3() [1/2]

```
double sacfmt::Trace::t3 ( ) const [noexcept]
{ return doubles[sac_map.at(name::t3)]; }
```

Here is the caller graph for this function:



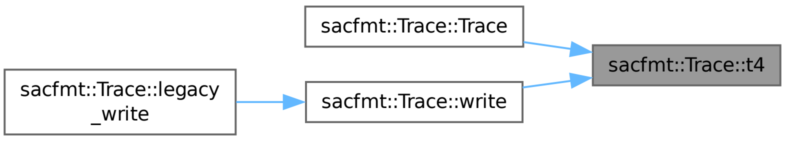
11.5.3.207 t3() [2/2]

```
void sacfmt::Trace::t3 (
            double input ) [noexcept]
{
  doubles[sac_map.at(name::t3)] = input;
}
```

11.5.3.208 t4() [1/2]

```
double sacfmt::Trace::t4 ( ) const [noexcept]
{ return doubles[sac_map.at(name::t4)]; }
```

Here is the caller graph for this function:



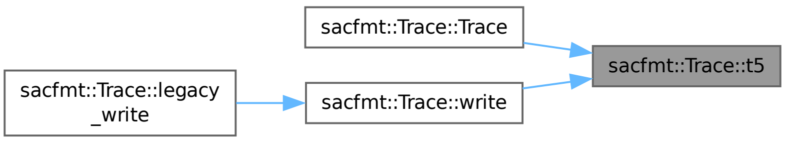
11.5.3.209 t4() [2/2]

```
void sacfmt::Trace::t4 (
            double input )  [noexcept]
                                        {
  doubles[sac_map.at(name::t4)] = input;
}
```

11.5.3.210 t5() [1/2]

```
double sacfmt::Trace::t5 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t5)]; }
```

Here is the caller graph for this function:



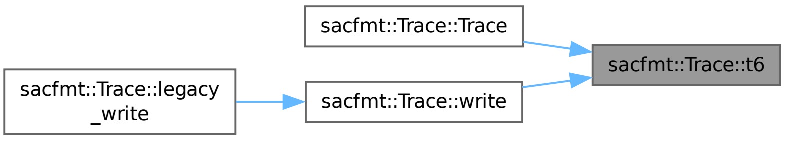
11.5.3.211 t5() [2/2]

```
void sacfmt::Trace::t5 (
            double input )  [noexcept]
                                        {
  doubles[sac_map.at(name::t5)] = input;
}
```

11.5.3.212 t6() [1/2]

```
double sacfmt::Trace::t6 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t6)]; }
```

Here is the caller graph for this function:



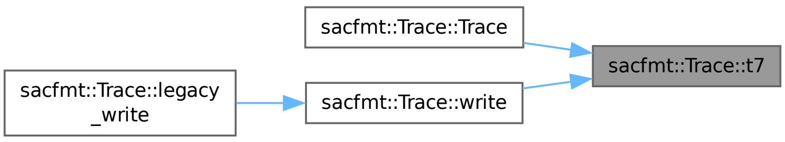
11.5.3.213 t6() [2/2]

```
void sacfmt::Trace::t6 (
            double input )   [noexcept]
                                        {
  doubles[sac_map.at(name::t6)] = input;
}
```

11.5.3.214 t7() [1/2]

```
double sacfmt::Trace::t7 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t7)]; }
```

Here is the caller graph for this function:



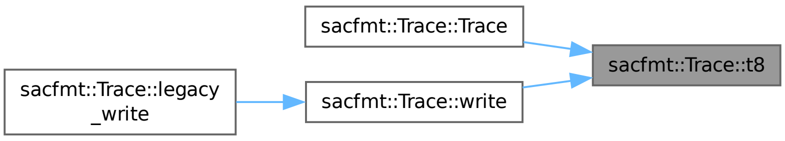
11.5.3.215 t7() [2/2]

```
void sacfmt::Trace::t7 (
            double input )   [noexcept]
                                        {
  doubles[sac_map.at(name::t7)] = input;
}
```

11.5.3.216 t8() [1/2]

```
double sacfmt::Trace::t8 ( ) const  [noexcept]
{ return doubles[sac_map.at(name::t8)]; }
```

Here is the caller graph for this function:



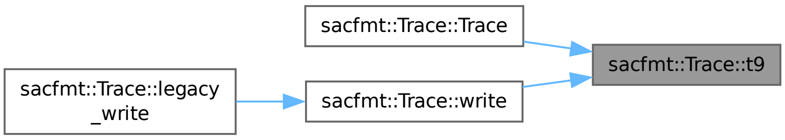
11.5.3.217 t8() [2/2]

```
void sacfmt::Trace::t8 (
            double input ) [noexcept]
                                        {
  doubles[sac_map.at(name::t8)] = input;
}
```

11.5.3.218 t9() [1/2]

```
double sacfmt::Trace::t9 ( ) const [noexcept]
{ return doubles[sac_map.at(name::t9)]; }
```

Here is the caller graph for this function:



11.5.3.219 t9() [2/2]

```
void sacfmt::Trace::t9 (
            double input ) [noexcept]
                                        {
  doubles[sac_map.at(name::t9)] = input;
}
```

11.5.3.220 time()

```
std::string sacfmt::Trace::time ( ) const [noexcept]
```

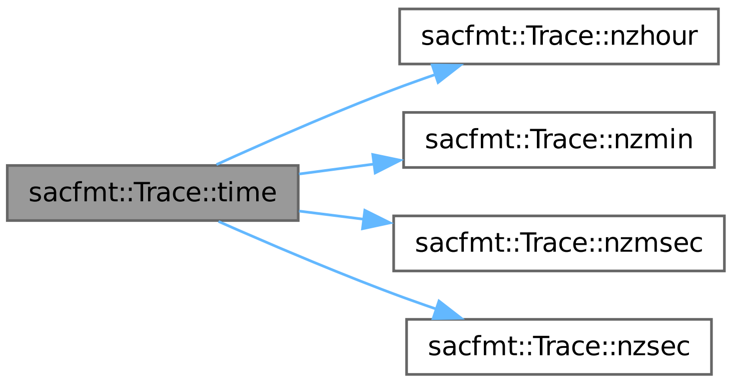
Get time string.

Returns

sstd::string Time (HH::MM:SS.sss).

```
                                    {
  // Require all to be set
  if ((nzhour() == unset_int) || (nzmin() == unset_int) ||
      (nzsec() == unset_int) || (nzmsec() == unset_int)) {
    return unset_word;
  }
  std::ostringstream oss{};
  oss « nzhour();
  oss « ':';
  oss « nzmin();
  oss « ':';
  oss « nzsec();
  oss « '.';
  oss « nzmsec();
  return oss.str();
}
```

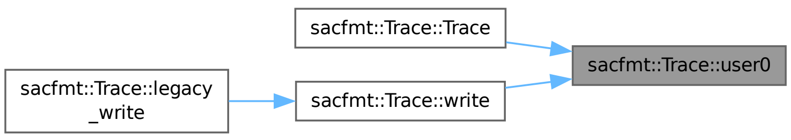
Here is the call graph for this function:



11.5.3.221 user0() [1/2]

```
float sacfmt::Trace::user0 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user0)]; }
```

Here is the caller graph for this function:



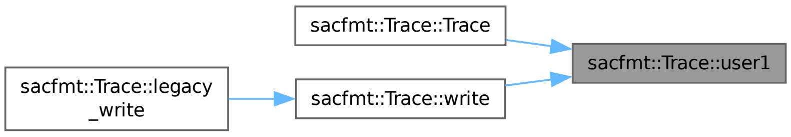
11.5.3.222 user0() [2/2]

```
void sacfmt::Trace::user0 (
             float input )  [noexcept]
                                              {
  floats[sac_map.at(name::user0)] = input;
}
```

11.5.3.223 user1() [1/2]

```
float sacfmt::Trace::user1 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user1)]; }
```

Here is the caller graph for this function:



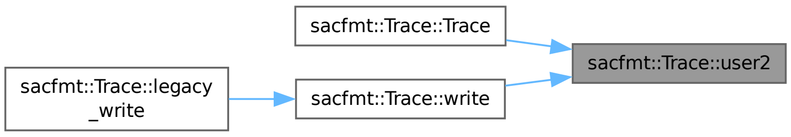
11.5.3.224 user1() [2/2]

```
void sacfmt::Trace::user1 (
            float input )  [noexcept]
                                              {
  floats[sac_map.at(name::user1)] = input;
}
```

11.5.3.225 user2() [1/2]

```
float sacfmt::Trace::user2 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user2)]; }
```

Here is the caller graph for this function:



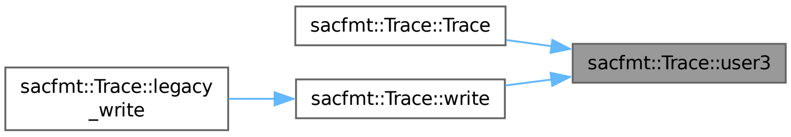
11.5.3.226 user2() [2/2]

```
void sacfmt::Trace::user2 (
            float input )  [noexcept]
                                              {
  floats[sac_map.at(name::user2)] = input;
}
```

11.5.3.227 user3() [1/2]

```
float sacfmt::Trace::user3 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user3)]; }
```

Here is the caller graph for this function:



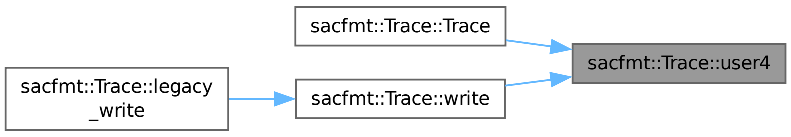
11.5.3.228 user3() [2/2]

```
void sacfmt::Trace::user3 (
            float input )  [noexcept]
{
  floats[sac_map.at(name::user3)] = input;
}
```

11.5.3.229 user4() [1/2]

```
float sacfmt::Trace::user4 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user4)]; }
```

Here is the caller graph for this function:



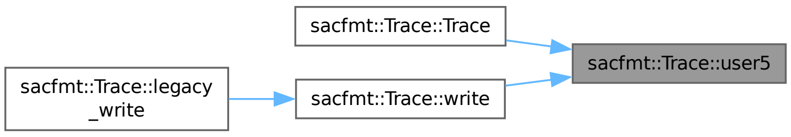
11.5.3.230 user4() [2/2]

```
void sacfmt::Trace::user4 (
            float input )  [noexcept]
{
  floats[sac_map.at(name::user4)] = input;
}
```

11.5.3.231 user5() [1/2]

```
float sacfmt::Trace::user5 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user5)]; }
```

Here is the caller graph for this function:



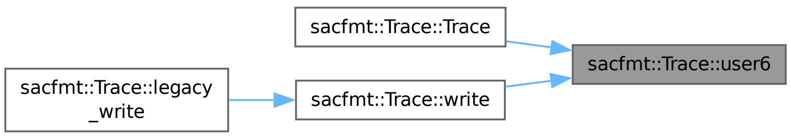
11.5.3.232 user5() [2/2]

```
void sacfmt::Trace::user5 (
            float input )  [noexcept]
{
  floats[sac_map.at(name::user5)] = input;
}
```

11.5.3.233 user6() [1/2]

```
float sacfmt::Trace::user6 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user6)]; }
```

Here is the caller graph for this function:



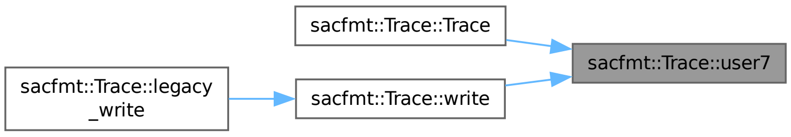
11.5.3.234 user6() [2/2]

```
void sacfmt::Trace::user6 (
            float input )  [noexcept]
{
  floats[sac_map.at(name::user6)] = input;
}
```

11.5.3.235 user7() [1/2]

```
float sacfmt::Trace::user7 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user7)]; }
```

Here is the caller graph for this function:



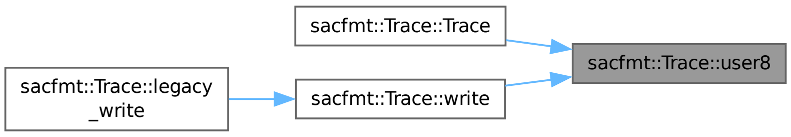
11.5.3.236 user7() [2/2]

```
void sacfmt::Trace::user7 (
            float input )  [noexcept]
                                                    {
  floats[sac_map.at(name::user7)] = input;
}
```

11.5.3.237 user8() [1/2]

```
float sacfmt::Trace::user8 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user8)]; }
```

Here is the caller graph for this function:



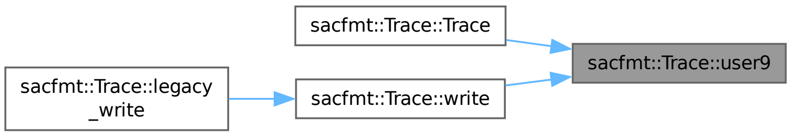
11.5.3.238 user8() [2/2]

```
void sacfmt::Trace::user8 (
            float input )  [noexcept]
                                                    {
  floats[sac_map.at(name::user8)] = input;
}
```

11.5.3.239 user9() [1/2]

```
float sacfmt::Trace::user9 ( ) const  [noexcept]
{ return floats[sac_map.at(name::user9)]; }
```

Here is the caller graph for this function:



11.5.3.240 user9() [2/2]

```
void sacfmt::Trace::user9 (
            float input )  [noexcept]
                                                {
  floats[sac_map.at(name::user9)] = input;
}
```

11.5.3.241 write()

```
void sacfmt::Trace::write (
            const std::filesystem::path & path,
            bool legacy = false ) const
```

Binary SAC-file writer.

Parameters

in	*path*	std::filesystem::path SAC-file to write.
in	*legacy*	bool Legacy-write flag (default false = v7, true = v6).

Exceptions

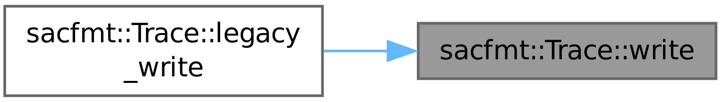
io_error	If the file cannot be written (bad path or bad permissions).
std::exception	Other unwritable issues (not enough space, disk failure, etc.).

```
                                                                          {
  std::ofstream file(path, std::ios::binary | std::ios::out | std::ios::trunc);
  if (!file) {
    throw io_error(path.string() + " cannot be opened to write.");
  }
  const int header_version{legacy ? old_hdr_version : modern_hdr_version};
  write_words(&file, convert_to_word(static_cast<float>(delta())));
  write_words(&file, convert_to_word(depmin()));
  write_words(&file, convert_to_word(depmax()));
  // Fill 'unused'
  write_words(&file, convert_to_word(depmax()));
  write_words(&file, convert_to_word(odelta()));
  write_words(&file, convert_to_word(static_cast<float>(b())));
```

```
write_words(&file, convert_to_word(static_cast<float>(e())));
write_words(&file, convert_to_word(static_cast<float>(o())));
write_words(&file, convert_to_word(static_cast<float>(a())));
// Fill 'internal'
write_words(&file, convert_to_word(depmin()));
write_words(&file, convert_to_word(static_cast<float>(t0())));
write_words(&file, convert_to_word(static_cast<float>(t1())));
write_words(&file, convert_to_word(static_cast<float>(t2())));
write_words(&file, convert_to_word(static_cast<float>(t3())));
write_words(&file, convert_to_word(static_cast<float>(t4())));
write_words(&file, convert_to_word(static_cast<float>(t5())));
write_words(&file, convert_to_word(static_cast<float>(t6())));
write_words(&file, convert_to_word(static_cast<float>(t7())));
write_words(&file, convert_to_word(static_cast<float>(t8())));
write_words(&file, convert_to_word(static_cast<float>(t9())));
write_words(&file, convert_to_word(static_cast<float>(f())));
write_words(&file, convert_to_word(resp0()));
write_words(&file, convert_to_word(resp1()));
write_words(&file, convert_to_word(resp2()));
write_words(&file, convert_to_word(resp3()));
write_words(&file, convert_to_word(resp4()));
write_words(&file, convert_to_word(resp5()));
write_words(&file, convert_to_word(resp6()));
write_words(&file, convert_to_word(resp7()));
write_words(&file, convert_to_word(resp8()));
write_words(&file, convert_to_word(resp9()));
write_words(&file, convert_to_word(static_cast<float>(stla())));
write_words(&file, convert_to_word(static_cast<float>(stlo())));
write_words(&file, convert_to_word(stel()));
write_words(&file, convert_to_word(stdp()));
write_words(&file, convert_to_word(static_cast<float>(evla())));
write_words(&file, convert_to_word(static_cast<float>(evlo())));
write_words(&file, convert_to_word(evel()));
write_words(&file, convert_to_word(evdp()));
write_words(&file, convert_to_word(mag()));
write_words(&file, convert_to_word(user0()));
write_words(&file, convert_to_word(user1()));
write_words(&file, convert_to_word(user2()));
write_words(&file, convert_to_word(user3()));
write_words(&file, convert_to_word(user4()));
write_words(&file, convert_to_word(user5()));
write_words(&file, convert_to_word(user6()));
write_words(&file, convert_to_word(user7()));
write_words(&file, convert_to_word(user8()));
write_words(&file, convert_to_word(user9()));
write_words(&file, convert_to_word(dist()));
write_words(&file, convert_to_word(az()));
write_words(&file, convert_to_word(baz()));
write_words(&file, convert_to_word(gcarc()));
write_words(&file, convert_to_word(static_cast<float>(sb())));
write_words(&file, convert_to_word(static_cast<float>(sdelta())));
write_words(&file, convert_to_word(depmen()));
write_words(&file, convert_to_word(cmpaz()));
write_words(&file, convert_to_word(cmpinc()));
write_words(&file, convert_to_word(xminimum()));
write_words(&file, convert_to_word(xmaximum()));
write_words(&file, convert_to_word(yminimum()));
write_words(&file, convert_to_word(ymaximum()));
// Fill 'unused' (xcommon_skip_num)
for (int i{0}; i < common_skip_num; ++i) {
  write_words(&file, convert_to_word(az()));
}
write_words(&file, convert_to_word(nzyear()));
write_words(&file, convert_to_word(nzjday()));
write_words(&file, convert_to_word(nzhour()));
write_words(&file, convert_to_word(nzmin()));
write_words(&file, convert_to_word(nzsec()));
write_words(&file, convert_to_word(nzmsec()));
write_words(&file, convert_to_word(header_version));
write_words(&file, convert_to_word(norid()));
write_words(&file, convert_to_word(nevid()));
write_words(&file, convert_to_word(npts()));
write_words(&file, convert_to_word(nsnpts()));
write_words(&file, convert_to_word(nwfid()));
write_words(&file, convert_to_word(nxsize()));
write_words(&file, convert_to_word(nysize()));
// Fill 'unused'
write_words(&file, convert_to_word(nysize()));
write_words(&file, convert_to_word(iftype()));
write_words(&file, convert_to_word(idep()));
write_words(&file, convert_to_word(iztype()));
// Fill 'unused'
write_words(&file, convert_to_word(iztype()));
write_words(&file, convert_to_word(iinst()));
write_words(&file, convert_to_word(istreg()));
write_words(&file, convert_to_word(ievreg()));
write_words(&file, convert_to_word(ievtyp()));
write_words(&file, convert_to_word(iqual()));
write_words(&file, convert_to_word(isynth()));
write_words(&file, convert_to_word(imagtyp()));
write_words(&file, convert_to_word(imagsrc()));
write_words(&file, convert_to_word(ibody()));
// Fill 'unused' (xcommon_skip_num)
for (int i{0}; i < common_skip_num; ++i) {
  write_words(&file, convert_to_word(ibody()));
}
write_words(&file, bool_to_word(leven()));
write_words(&file, bool_to_word(lpspol()));
write_words(&file, bool_to_word(lovrok()));
write_words(&file, bool_to_word(lcalda()));
// Fill 'unused'
write_words(&file, bool_to_word(lcalda()));
// Strings are special
std::array<char, static_cast<size_t>(2) * word_length> two_words{
    convert_to_words<sizeof(two_words)>(kstnm(), 2)};
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

std::array<char, static_cast<size_t>(4) * word_length> four_words{
    convert_to_words<sizeof(four_words)>(kevnm(), 4)};
write_words(&file, std::vector<char>(four_words.begin(), four_words.end()));

two_words = convert_to_words<sizeof(two_words)>(khole(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(ko(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(ka(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt0(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt1(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt2(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt3(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt4(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt5(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt6(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt7(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt8(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kt9(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kf(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kuser0(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kuser1(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kuser2(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kcmpnm(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(knetwk(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kdatrd(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));

two_words = convert_to_words<sizeof(two_words)>(kinst(), 2);
write_words(&file, std::vector<char>(two_words.begin(), two_words.end()));
// Data
```

```
  for (double dub : data1()) [[likely]] {
    write_words(&file, convert_to_word(static_cast<float>(dub)));
  }
  if (!leven() || (iftype() > 1)) {
    for (double dub : data2()) {
      write_words(&file, convert_to_word(static_cast<float>(dub)));
    }
  }
  if (header_version == modern_hdr_version) {
    // Write footer
    write_words(&file, convert_to_word(delta()));
    write_words(&file, convert_to_word(b()));
    write_words(&file, convert_to_word(e()));
    write_words(&file, convert_to_word(o()));
    write_words(&file, convert_to_word(a()));
    write_words(&file, convert_to_word(t0()));
    write_words(&file, convert_to_word(t1()));
    write_words(&file, convert_to_word(t2()));
    write_words(&file, convert_to_word(t3()));
    write_words(&file, convert_to_word(t4()));
    write_words(&file, convert_to_word(t5()));
    write_words(&file, convert_to_word(t6()));
    write_words(&file, convert_to_word(t7()));
    write_words(&file, convert_to_word(t8()));
    write_words(&file, convert_to_word(t9()));
    write_words(&file, convert_to_word(f()));
    write_words(&file, convert_to_word(evlo()));
    write_words(&file, convert_to_word(evla()));
    write_words(&file, convert_to_word(stlo()));
    write_words(&file, convert_to_word(stla()));
    write_words(&file, convert_to_word(sb()));
    write_words(&file, convert_to_word(sdelta()));
  }
  file.close();
}
```

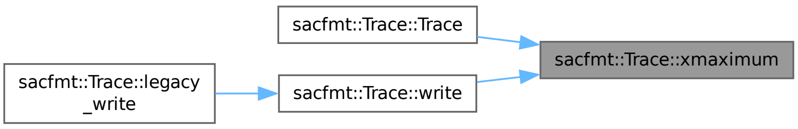
Here is the caller graph for this function:



11.5.3.242 xmaximum() [1/2]

```
float sacfmt::Trace::xmaximum ( ) const  [noexcept]
                                        {
  return floats[sac_map.at(name::xmaximum)];
}
```

Here is the caller graph for this function:



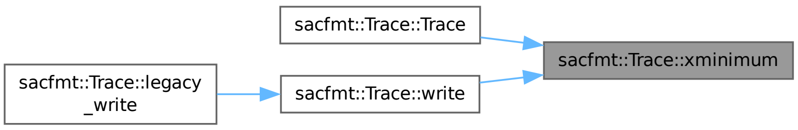
11.5.3.243 xmaximum() [2/2]

```
void sacfmt::Trace::xmaximum (
              float input )  [noexcept]
{
  floats[sac_map.at(name::xmaximum)] = input;
}
```

11.5.3.244 xminimum() [1/2]

```
float sacfmt::Trace::xminimum ( ) const  [noexcept]
{
  return floats[sac_map.at(name::xminimum)];
}
```

Here is the caller graph for this function:



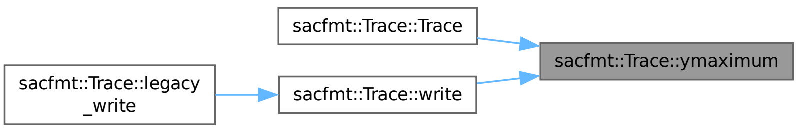
11.5.3.245 xminimum() [2/2]

```
void sacfmt::Trace::xminimum (
              float input )  [noexcept]
{
  floats[sac_map.at(name::xminimum)] = input;
}
```

11.5.3.246 ymaximum() [1/2]

```
float sacfmt::Trace::ymaximum ( ) const  [noexcept]
{
  return floats[sac_map.at(name::ymaximum)];
}
```

Here is the caller graph for this function:



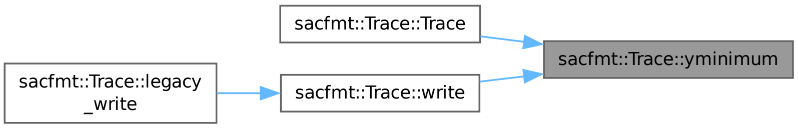
11.5.3.247 ymaximum() [2/2]

```
void sacfmt::Trace::ymaximum (
                float input )   [noexcept]
{
  floats[sac_map.at(name::ymaximum)] = input;
}
```

11.5.3.248 yminimum() [1/2]

```
float sacfmt::Trace::yminimum ( ) const  [noexcept]
{
  return floats[sac_map.at(name::yminimum)];
}
```

Here is the caller graph for this function:



11.5.3.249 yminimum() [2/2]

```
void sacfmt::Trace::yminimum (
                float input )   [noexcept]
{
  floats[sac_map.at(name::yminimum)] = input;
}
```

11.5.4 Member Data Documentation

11.5.4.1 bools

```
std::array<bool, num_bool> sacfmt::Trace::bools {}  [private]
```

Boolean storage array.

```
{};
```

11.5.4.2 data

```
std::array<std::vector<double>, num_data> sacfmt::Trace::data {}  [private]
```

std::vector<double> storage array.

```
{};
```

11.5.4.3 doubles

```
std::array<double, num_double> sacfmt::Trace::doubles {}  [private]
```

Double storage array.

```
01363 {};
```

11.5.4.4 floats

```
std::array<float, num_float> sacfmt::Trace::floats {}  [private]
```

Float storage array.

```
01361 {};
```

11.5.4.5 ints

```
std::array<int, num_int> sacfmt::Trace::ints {}  [private]
```

Integer storage array.

```
01365 {};
```

11.5.4.6 strings

```
std::array<std::string, num_string> sacfmt::Trace::strings {}  [private]
```

String storage array.

```
01369 {};
```

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

- include/sac-format/sac_format.hpp
- src/sac_format.cpp

11.6 sacfmt::bitset_type::uint< nbits > Struct Template Reference

Ensure type-safety for conversions between floats/doubles and bitsets.

```
#include <sac_format.hpp>
```

11.6.1 Detailed Description

template<unsigned nbits>
struct sacfmt::bitset_type::uint< nbits >

Ensure type-safety for conversions between floats/doubles and bitsets.

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

- include/sac-format/sac_format.hpp

11.7 sacfmt::bitset_type::uint< 4 *bits_per_byte > Struct Reference

One-word (floats).

```
#include <sac_format.hpp>
```

Public Types

- using type = uint32_t

11.7.1 Detailed Description

One-word (floats).

11.7.2 Member Typedef Documentation

11.7.2.1 type

```
using sacfmt::bitset_type::uint< 4 *bits_per_byte >::type = uint32_t
```

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

- include/sac-format/sac_format.hpp

11.8 sacfmt::bitset_type::uint< bytes *bits_per_byte > Struct Reference

Two-words (doubles)

```
#include <sac_format.hpp>
```

Public Types

- using type = uint64_t

11.8.1 Detailed Description

Two-words (doubles)

11.8.2 Member Typedef Documentation

11.8.2.1 type

```
using sacfmt::bitset_type::uint< bytes *bits_per_byte >::type = uint64_t
```

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

- include/sac-format/sac_format.hpp

11.9 sacfmt::word_pair< T > Struct Template Reference

Struct containing a pair of words.

```
#include <sac_format.hpp>
```

Public Attributes

- T first {}

  First 'word' in the pair.
- T second {}

  Second 'word' in the pair.

11.9.1 Detailed Description

template<typename T>
struct sacfmt::word_pair< T >

Struct containing a pair of words.

Prevents bug-prone word-swapping in functions that use a pair of words.

These are not necessarily single words, it could be a pair of word_one or a pair of word_two.

11.9.2 Member Data Documentation

11.9.2.1 first

```
template<typename T >
T sacfmt::word_pair< T >::first {}
```

First 'word' in the pair.

```
00192 {};
```

11.9.2.2 second

```
template<typename T >
T sacfmt::word_pair< T >::second {}
```

Second 'word' in the pair.

```
00193 {};
```

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

- include/sac-format/sac_format.hpp

Index