Codata

0.4.0

Generated by Doxygen 1.9.6

Chapter 1

Introduction

codata is a Fortran library providing the lastest codata constants (2018). It also provides a API for the C language. The raw codata from http://physics.nist.gov/constants are parsed line by line where the columns name, value, uncertainty and unit are formatted to be conform to Fortran double precision. The formatted (as strings) names, values, uncertainties and units are then inserted in a derived type in the generated Fortran module. The latter are then inserted into an array.

The generated Fortran module is then compiled (f2008+) into a shared and a static library libcodata with the Fortran and C headers.

The compilation was tested on Linux (Debian), MacOS and Windows.

Links:

- Sources: https://github.com/MilanSkocic/codata.
- Online documentation: https://milanskocic.github.io/codata/index.html.
- PDF documentation: https://milanskocic.github.io/codata/refman.pdf.
- Python wrapper: https://milanskocic.github.io/codata/pycodata/index.html.

1.1 Installation

See the file INSTALL.

1.2 Dependencies

See the file REQUIREMENTS.

1.3 License information

See the file LICENSE.

Chapter 2

Codata 0.1.0 Release Note

2.1 Changes

Implementation of:

- the parser of the codata raw data
- the generator of the Fortran modules
- the C API and C header
- the python wrapper (will be moved to its repository next release).

2.2 Download

Codata Releases

2.3 Contributors

Milan Skocic

2.4 Commits

Full Changelog: https://github.com/MilanSkocic/codata/compare/....0.1.0

Chapter 3

Codata 0.2.0 Release Note

3.1 Changes

- Bug fixes for the codata 2010.
- Bug fixes in the tests linked to the codata 2010.
- Add python wrapper for the number of constants method.

3.2 Download

Codata Releases

3.3 Contributors

Milan Skocic

3.4 Commits

Full Changelog: https://github.com/MilanSkocic/codata/compare/0.1.0...0.2.0

Chapter 4

Codata 0.2.1 Release Note

4.1 Changes

- Integration of Intel Fortran compiler and MSVC in cmake scripts.
- Add specifications and instructions for compiling on Windows

4.2 Download

Codata Releases

4.3 Contributors

Milan Skocic

4.4 Commits

Full Changelog: https://github.com/MilanSkocic/codata/compare/0.2.0...0.2.1

Chapter 5

Codata 0.3.0 Release Note

5.1 Changes

- Only last codata constants.

5.2 Download

Codata Releases

5.3 Contributors

Milan Skocic

5.4 Commits

Full Changelog: https://github.com/MilanSkocic/codata/compare/0.2.1...0.3.0

Chapter 6

install

Cmake is necessary for compiling and instaling the library.

6.1 Create build directory

- mkdir build
- cd build

6.2 Generate a makefile

- On Unix-like OS: `cmake -G "Unix Makefiles" -S .. -DCMAKE_BUILD_TYPE=release -DCMAKE_INSTALL_PREFIX=/path/to/folder`
- On windows with MSYS2: `cmake -G "Unix Makefiles" -S .. -DCMAKE_BUILD_TYPE=release -DCMAKE_INSTALL_PREFIX=/path/to/folder`
- On windows with ifort and msvc: `cmake -G "NMake Makefiles" -S .. -DCMAKE_BUILD_TYPE=release -DCMAKE_INSTALL_PREFIX=/path/to/folder`

6.3 Build either with cmake

```
cmake --build .
```

6.4 Run tests

```
ctest
```

6.5 Install

```
cmake --install .
```

6.6 Dependencies

On windows when compiled with Intel Fortran compiler, the `Intel Fortran redistributable` must be installed.

Chapter 7

license

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to

avoid the special danger that patents applied to a free program could
make it effectively proprietary. To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License. Each licensee is addressed as "you". "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy. The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy. Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies. Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License. If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it. "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form. A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities. However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work. For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding

Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source. This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge. You need not require recipients to copy the
    Corresponding Source along with the object code. If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source. Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling. In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage. For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product. A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source. The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the

Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly

provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a

patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you

to choose that version for the Program.

  Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see

Chapter 8

Codata Raw Data

The raw data that are processed for generating the C and Fortran source codes are the followings:

```
            Fundamental Physical Constants --- Complete Listing
            2018 CODATA adjustment


  From:  http://physics.nist.gov/constants
```

Quantity	Value	Uncertainty	Unit
alpha particle-electron mass ratio	7294.299 541 42	0.000 000 24	
alpha particle mass	6.644 657 3357 e-27	0.000 000 0020 e-27	kg
alpha particle mass energy equivalent	5.971 920 1914 e-10	0.000 000 0018 e-10	J
alpha particle mass energy equivalent in MeV	3727.379 4066	0.000 0011	MeV
alpha particle mass in u	4.001 506 179 127	0.000 000 000 063	u
alpha particle molar mass	4.001 506 1777 e-3	0.000 000 0012 e-3	kg mol^-1
alpha particle-proton mass ratio	3.972 599 690 09	0.000 000 000 22	
alpha particle relative atomic mass	4.001 506 179 127	0.000 000 000 063	
Angstrom star	1.000 014 95 e-10	0.000 000 90 e-10	m
atomic mass constant	1.660 539 066 60 e-27	0.000 000 000 50 e-27	kg
atomic mass constant energy equivalent	1.492 418 085 60 e-10	0.000 000 000 45 e-10	J
atomic mass constant energy equivalent in MeV	931.494 102 42	0.000 000 28	MeV
atomic mass unit-electron volt relationship	9.314 941 0242 e8	0.000 000 0028 e8	eV
atomic mass unit-hartree relationship	3.423 177 6874 e7	0.000 000 0010 e7	E_h
atomic mass unit-hertz relationship	2.252 342 718 71 e23	0.000 000 000 68 e23	Hz
atomic mass unit-inverse meter relationship	7.513 006 6104 e14	0.000 000 0023 e14	m^-1
atomic mass unit-joule relationship	1.492 418 085 60 e-10	0.000 000 000 45 e-10	J
atomic mass unit-kelvin relationship	1.080 954 019 16 e13	0.000 000 000 33 e13	K
atomic mass unit-kilogram relationship	1.660 539 066 60 e-27	0.000 000 000 50 e-27	kg
atomic unit of 1st hyperpolarizability	3.206 361 3061 e-53	0.000 000 0015 e-53	C^3 m^3 J^-2
atomic unit of 2nd hyperpolarizability	6.235 379 9905 e-65	0.000 000 0038 e-65	C^4 m^4 J^-3
atomic unit of action	1.054 571 817... e-34	(exact)	J s
atomic unit of charge	1.602 176 634 e-19	(exact)	C
atomic unit of charge density	1.081 202 384 57 e12	0.000 000 000 49 e12	C m^-3

atomic unit of current	6.623 618 237 510 e-3	0.000 000 000 013 e-3
A		
atomic unit of electric dipole mom.	8.478 353 6255 e-30	0.000 000 0013 e-30
C m		
atomic unit of electric field	5.142 206 747 63 e11	0.000 000 000 78 e11
V m^-1		
atomic unit of electric field gradient	9.717 362 4292 e21	0.000 000 0029 e21
V m^-2		
atomic unit of electric polarizability	1.648 777 274 36 e-41	0.000 000 000 50 e-41
C^2 m^2 J^-1		
atomic unit of electric potential	27.211 386 245 988	0.000 000 000 053
V		
atomic unit of electric quadrupole mom.	4.486 551 5246 e-40	0.000 000 0014 e-40
C m^2		
atomic unit of energy	4.359 744 722 2071 e-18	0.000 000 000 0085 e-18
J		
atomic unit of force	8.238 723 4983 e-8	0.000 000 0012 e-8
N		
atomic unit of length	5.291 772 109 03 e-11	0.000 000 000 80 e-11
m		
atomic unit of mag. dipole mom.	1.854 802 015 66 e-23	0.000 000 000 56 e-23
J T^-1		
atomic unit of mag. flux density	2.350 517 567 58 e5	0.000 000 000 71 e5
T		
atomic unit of magnetizability	7.891 036 6008 e-29	0.000 000 0048 e-29
J T^-2		
atomic unit of mass	9.109 383 7015 e-31	0.000 000 0028 e-31
kg		
atomic unit of momentum	1.992 851 914 10 e-24	0.000 000 000 30 e-24
kg m s^-1		
atomic unit of permittivity	1.112 650 055 45 e-10	0.000 000 000 17 e-10
F m^-1		
atomic unit of time	2.418 884 326 5857 e-17	0.000 000 000 0047 e-17
s		
atomic unit of velocity	2.187 691 263 64 e6	0.000 000 000 33 e6
m s^-1		
Avogadro constant	6.022 140 76 e23	(exact)
mol^-1		
Bohr magneton	9.274 010 0783 e-24	0.000 000 0028 e-24
J T^-1		
Bohr magneton in eV/T	5.788 381 8060 e-5	0.000 000 0017 e-5
eV T^-1		
Bohr magneton in Hz/T	1.399 624 493 61 e10	0.000 000 000 42 e10
Hz T^-1		
Bohr magneton in inverse meter per tesla	46.686 447 783	0.000 000 014
m^-1 T^-1		
Bohr magneton in K/T	0.671 713 815 63	0.000 000 000 20
K T^-1		
Bohr radius	5.291 772 109 03 e-11	0.000 000 000 80 e-11
m		
Boltzmann constant	1.380 649 e-23	(exact)
J K^-1		
Boltzmann constant in eV/K	8.617 333 262... e-5	(exact)
eV K^-1		
Boltzmann constant in Hz/K	2.083 661 912... e10	(exact)
Hz K^-1		
Boltzmann constant in inverse meter per kelvin	69.503 480 04...	(exact)
m^-1 K^-1		
characteristic impedance of vacuum	376.730 313 668	0.000 000 057
ohm		
classical electron radius	2.817 940 3262 e-15	0.000 000 0013 e-15
m		
Compton wavelength	2.426 310 238 67 e-12	0.000 000 000 73 e-12
m		
conductance quantum	7.748 091 729... e-5	(exact)
S		
conventional value of ampere-90	1.000 000 088 87...	(exact)
A		
conventional value of coulomb-90	1.000 000 088 87...	(exact)
C		
conventional value of farad-90	0.999 999 982 20...	(exact)
F		
conventional value of henry-90	1.000 000 017 79...	(exact)
H		
conventional value of Josephson constant	483 597.9 e9	(exact)
Hz V^-1		
conventional value of ohm-90	1.000 000 017 79...	(exact)
ohm		
conventional value of volt-90	1.000 000 106 66...	(exact)
V		
conventional value of von Klitzing constant	25 812.807	(exact)
ohm		
conventional value of watt-90	1.000 000 195 53...	(exact)
W		
Copper x unit	1.002 076 97 e-13	0.000 000 28 e-13
m		
deuteron-electron mag. mom. ratio	-4.664 345 551 e-4	0.000 000 012 e-4

deuteron-electron mass ratio	3670.482 967 88	0.000 000 13
deuteron g factor	0.857 438 2338	0.000 000 0022
deuteron mag. mom. J T^-1	4.330 735 094 e-27	0.000 000 011 e-27
deuteron mag. mom. to Bohr magneton ratio	4.669 754 570 e-4	0.000 000 012 e-4
deuteron mag. mom. to nuclear magneton ratio	0.857 438 2338	0.000 000 0022
deuteron mass kg	3.343 583 7724 e-27	0.000 000 0010 e-27
deuteron mass energy equivalent J	3.005 063 231 02 e-10	0.000 000 000 91 e-10
deuteron mass energy equivalent in MeV MeV	1875.612 942 57	0.000 000 57
deuteron mass in u u	2.013 553 212 745	0.000 000 000 040
deuteron molar mass kg mol^-1	2.013 553 212 05 e-3	0.000 000 000 61 e-3
deuteron-neutron mag. mom. ratio	-0.448 206 53	0.000 000 11
deuteron-proton mag. mom. ratio	0.307 012 209 39	0.000 000 000 79
deuteron-proton mass ratio	1.999 007 501 39	0.000 000 000 11
deuteron relative atomic mass	2.013 553 212 745	0.000 000 000 040
deuteron rms charge radius m	2.127 99 e-15	0.000 74 e-15
electron charge to mass quotient C kg^-1	-1.758 820 010 76 e11	0.000 000 000 53 e11
electron-deuteron mag. mom. ratio	-2143.923 4915	0.000 0056
electron-deuteron mass ratio	2.724 437 107 462 e-4	0.000 000 000 096 e-4
electron g factor	-2.002 319 304 362 56	0.000 000 000 000 35
electron gyromag. ratio s^-1 T^-1	1.760 859 630 23 e11	0.000 000 000 53 e11
electron gyromag. ratio in MHz/T MHz T^-1	28 024.951 4242	0.000 0085
electron-helion mass ratio	1.819 543 074 573 e-4	0.000 000 000 079 e-4
electron mag. mom. J T^-1	-9.284 764 7043 e-24	0.000 000 0028 e-24
electron mag. mom. anomaly	1.159 652 181 28 e-3	0.000 000 000 18 e-3
electron mag. mom. to Bohr magneton ratio	-1.001 159 652 181 28	0.000 000 000 000 18
electron mag. mom. to nuclear magneton ratio	-1838.281 971 88	0.000 000 11
electron mass kg	9.109 383 7015 e-31	0.000 000 0028 e-31
electron mass energy equivalent J	8.187 105 7769 e-14	0.000 000 0025 e-14
electron mass energy equivalent in MeV MeV	0.510 998 950 00	0.000 000 000 15
electron mass in u u	5.485 799 090 65 e-4	0.000 000 000 16 e-4
electron molar mass kg mol^-1	5.485 799 0888 e-7	0.000 000 0017 e-7
electron-muon mag. mom. ratio	206.766 9883	0.000 0046
electron-muon mass ratio	4.836 331 69 e-3	0.000 000 11 e-3
electron-neutron mag. mom. ratio	960.920 50	0.000 23
electron-neutron mass ratio	5.438 673 4424 e-4	0.000 000 0026 e-4
electron-proton mag. mom. ratio	-658.210 687 89	0.000 000 20
electron-proton mass ratio	5.446 170 214 87 e-4	0.000 000 000 33 e-4
electron relative atomic mass	5.485 799 090 65 e-4	0.000 000 000 16 e-4
electron-tau mass ratio	2.875 85 e-4	0.000 19 e-4
electron to alpha particle mass ratio	1.370 933 554 787 e-4	0.000 000 000 045 e-4
electron to shielded helion mag. mom. ratio	864.058 257	0.000 010
electron to shielded proton mag. mom. ratio	-658.227 5971	0.000 0072
electron-triton mass ratio	1.819 200 062 251 e-4	0.000 000 000 090 e-4

electron volt	1.602 176 634 e-19	(exact)
J		
electron volt-atomic mass unit relationship	1.073 544 102 33 e-9	0.000 000 000 32 e-9
u		
electron volt-hartree relationship	3.674 932 217 5655 e-2	0.000 000 000 0071 e-2
E_h		
electron volt-hertz relationship	2.417 989 242... e14	(exact)
Hz		
electron volt-inverse meter relationship	8.065 543 937... e5	(exact)
m^-1		
electron volt-joule relationship	1.602 176 634 e-19	(exact)
J		
electron volt-kelvin relationship	1.160 451 812... e4	(exact)
K		
electron volt-kilogram relationship	1.782 661 921... e-36	(exact)
kg		
elementary charge	1.602 176 634 e-19	(exact)
C		
elementary charge over h-bar	1.519 267 447... e15	(exact)
A J^-1		
Faraday constant	96 485.332 12...	(exact)
C mol^-1		
Fermi coupling constant	1.166 3787 e-5	0.000 0006 e-5
GeV^-2		
fine-structure constant	7.297 352 5693 e-3	0.000 000 0011 e-3
first radiation constant	3.741 771 852... e-16	(exact)
W m^2		
first radiation constant for spectral radiance	1.191 042 972... e-16	(exact)
W m^2 sr^-1		
hartree-atomic mass unit relationship	2.921 262 322 05 e-8	0.000 000 000 88 e-8
u		
hartree-electron volt relationship	27.211 386 245 988	0.000 000 000 053
eV		
Hartree energy	4.359 744 722 2071 e-18	0.000 000 000 0085 e-18
J		
Hartree energy in eV	27.211 386 245 988	0.000 000 000 053
eV		
hartree-hertz relationship	6.579 683 920 502 e15	0.000 000 000 013 e15
Hz		
hartree-inverse meter relationship	2.194 746 313 6320 e7	0.000 000 000 0043 e7
m^-1		
hartree-joule relationship	4.359 744 722 2071 e-18	0.000 000 000 0085 e-18
J		
hartree-kelvin relationship	3.157 750 248 0407 e5	0.000 000 000 0061 e5
K		
hartree-kilogram relationship	4.850 870 209 5432 e-35	0.000 000 000 0094 e-35
kg		
helion-electron mass ratio	5495.885 280 07	0.000 000 24
helion g factor	-4.255 250 615	0.000 000 050
helion mag. mom.	-1.074 617 532 e-26	0.000 000 013 e-26
J T^-1		
helion mag. mom. to Bohr magneton ratio	-1.158 740 958 e-3	0.000 000 014 e-3
helion mag. mom. to nuclear magneton ratio	-2.127 625 307	0.000 000 025
helion mass	5.006 412 7796 e-27	0.000 000 0015 e-27
kg		
helion mass energy equivalent	4.499 539 4125 e-10	0.000 000 0014 e-10
J		
helion mass energy equivalent in MeV	2808.391 607 43	0.000 000 85
MeV		
helion mass in u	3.014 932 247 175	0.000 000 000 097
u		
helion molar mass	3.014 932 246 13 e-3	0.000 000 000 91 e-3
kg mol^-1		
helion-proton mass ratio	2.993 152 671 67	0.000 000 000 13
helion relative atomic mass	3.014 932 247 175	0.000 000 000 097
helion shielding shift	5.996 743 e-5	0.000 010 e-5
hertz-atomic mass unit relationship	4.439 821 6652 e-24	0.000 000 0013 e-24
u		
hertz-electron volt relationship	4.135 667 696... e-15	(exact)
eV		
hertz-hartree relationship	1.519 829 846 0570 e-16	0.000 000 000 0029 e-16
E_h		
hertz-inverse meter relationship	3.335 640 951... e-9	(exact)
m^-1		
hertz-joule relationship	6.626 070 15 e-34	(exact)
J		
hertz-kelvin relationship	4.799 243 073... e-11	(exact)
K		
hertz-kilogram relationship	7.372 497 323... e-51	(exact)

kg		
hyperfine transition frequency of Cs-133	9 192 631 770	(exact)
Hz		
inverse fine-structure constant	137.035 999 084	0.000 000 021
inverse meter-atomic mass unit relationship	1.331 025 050 10 e-15	0.000 000 000 40 e-15
u		
inverse meter-electron volt relationship	1.239 841 984... e-6	(exact)
eV		
inverse meter-hartree relationship	4.556 335 252 9120 e-8	0.000 000 000 0088 e-8
E_h		
inverse meter-hertz relationship	299 792 458	(exact)
Hz		
inverse meter-joule relationship	1.986 445 857... e-25	(exact)
J		
inverse meter-kelvin relationship	1.438 776 877... e-2	(exact)
K		
inverse meter-kilogram relationship	2.210 219 094... e-42	(exact)
kg		
inverse of conductance quantum	12 906.403 72...	(exact)
ohm		
Josephson constant	483 597.848 4... e9	(exact)
Hz V^-1		
joule-atomic mass unit relationship	6.700 535 2565 e9	0.000 000 0020 e9
u		
joule-electron volt relationship	6.241 509 074... e18	(exact)
eV		
joule-hartree relationship	2.293 712 278 3963 e17	0.000 000 000 0045 e17
E_h		
joule-hertz relationship	1.509 190 179... e33	(exact)
Hz		
joule-inverse meter relationship	5.034 116 567... e24	(exact)
m^-1		
joule-kelvin relationship	7.242 970 516... e22	(exact)
K		
joule-kilogram relationship	1.112 650 056... e-17	(exact)
kg		
kelvin-atomic mass unit relationship	9.251 087 3014 e-14	0.000 000 0028 e-14
u		
kelvin-electron volt relationship	8.617 333 262... e-5	(exact)
eV		
kelvin-hartree relationship	3.166 811 563 4556 e-6	0.000 000 000 0061 e-6
E_h		
kelvin-hertz relationship	2.083 661 912... e10	(exact)
Hz		
kelvin-inverse meter relationship	69.503 480 04...	(exact)
m^-1		
kelvin-joule relationship	1.380 649 e-23	(exact)
J		
kelvin-kilogram relationship	1.536 179 187... e-40	(exact)
kg		
kilogram-atomic mass unit relationship	6.022 140 7621 e26	0.000 000 0018 e26
u		
kilogram-electron volt relationship	5.609 588 603... e35	(exact)
eV		
kilogram-hartree relationship	2.061 485 788 7409 e34	0.000 000 000 0040 e34
E_h		
kilogram-hertz relationship	1.356 392 489... e50	(exact)
Hz		
kilogram-inverse meter relationship	4.524 438 335... e41	(exact)
m^-1		
kilogram-joule relationship	8.987 551 787... e16	(exact)
J		
kilogram-kelvin relationship	6.509 657 260... e39	(exact)
K		
lattice parameter of silicon	5.431 020 511 e-10	0.000 000 089 e-10
m		
lattice spacing of ideal Si (220)	1.920 155 716 e-10	0.000 000 032 e-10
m		
Loschmidt constant (273.15 K, 100 kPa)	2.651 645 804... e25	(exact)
m^-3		
Loschmidt constant (273.15 K, 101.325 kPa)	2.686 780 111... e25	(exact)
m^-3		
luminous efficacy	683	(exact)
lm W^-1		
mag. flux quantum	2.067 833 848... e-15	(exact)
Wb		
molar gas constant	8.314 462 618...	(exact)
J mol^-1 K^-1		
molar mass constant	0.999 999 999 65 e-3	0.000 000 000 30 e-3
kg mol^-1		
molar mass of carbon-12	11.999 999 9958 e-3	0.000 000 0036 e-3
kg mol^-1		
molar Planck constant	3.990 312 712... e-10	(exact)
J Hz^-1 mol^-1		
molar volume of ideal gas (273.15 K, 100 kPa)	22.710 954 64... e-3	(exact)
m^3 mol^-1		

molar volume of ideal gas (273.15 K, 101.325 kPa)	22.413 969 54... e-3	(exact)
m^3 mol^-1		
molar volume of silicon	1.205 883 199 e-5	0.000 000 060 e-5
m^3 mol^-1		
Molybdenum x unit	1.002 099 52 e-13	0.000 000 53 e-13
m		
muon Compton wavelength	1.173 444 110 e-14	0.000 000 026 e-14
m		
muon-electron mass ratio	206.768 2830	0.000 0046
muon g factor	-2.002 331 8418	0.000 000 0013
muon mag. mom.	-4.490 448 30 e-26	0.000 000 10 e-26
J T^-1		
muon mag. mom. anomaly	1.165 920 89 e-3	0.000 000 63 e-3
muon mag. mom. to Bohr magneton ratio	-4.841 970 47 e-3	0.000 000 11 e-3
muon mag. mom. to nuclear magneton ratio	-8.890 597 03	0.000 000 20
muon mass	1.883 531 627 e-28	0.000 000 042 e-28
kg		
muon mass energy equivalent	1.692 833 804 e-11	0.000 000 038 e-11
J		
muon mass energy equivalent in MeV	105.658 3755	0.000 0023
MeV		
muon mass in u	0.113 428 9259	0.000 000 0025
u		
muon molar mass	1.134 289 259 e-4	0.000 000 025 e-4
kg mol^-1		
muon-neutron mass ratio	0.112 454 5170	0.000 000 0025
muon-proton mag. mom. ratio	-3.183 345 142	0.000 000 071
muon-proton mass ratio	0.112 609 5264	0.000 000 0025
muon-tau mass ratio	5.946 35 e-2	0.000 40 e-2
natural unit of action	1.054 571 817... e-34	(exact)
J s		
natural unit of action in eV s	6.582 119 569... e-16	(exact)
eV s		
natural unit of energy	8.187 105 7769 e-14	0.000 000 0025 e-14
J		
natural unit of energy in MeV	0.510 998 950 00	0.000 000 000 15
MeV		
natural unit of length	3.861 592 6796 e-13	0.000 000 0012 e-13
m		
natural unit of mass	9.109 383 7015 e-31	0.000 000 0028 e-31
kg		
natural unit of momentum	2.730 924 530 75 e-22	0.000 000 000 82 e-22
kg m s^-1		
natural unit of momentum in MeV/c	0.510 998 950 00	0.000 000 000 15
MeV/c		
natural unit of time	1.288 088 668 19 e-21	0.000 000 000 39 e-21
s		
natural unit of velocity	299 792 458	(exact)
m s^-1		
neutron Compton wavelength	1.319 590 905 81 e-15	0.000 000 000 75 e-15
m		
neutron-electron mag. mom. ratio	1.040 668 82 e-3	0.000 000 25 e-3
neutron-electron mass ratio	1838.683 661 73	0.000 000 89
neutron g factor	-3.826 085 45	0.000 000 90
neutron gyromag. ratio	1.832 471 71 e8	0.000 000 43 e8
s^-1 T^-1		
neutron gyromag. ratio in MHz/T	29.164 6931	0.000 0069
MHz T^-1		
neutron mag. mom.	-9.662 3651 e-27	0.000 0023 e-27
J T^-1		
neutron mag. mom. to Bohr magneton ratio	-1.041 875 63 e-3	0.000 000 25 e-3
neutron mag. mom. to nuclear magneton ratio	-1.913 042 73	0.000 000 45
neutron mass	1.674 927 498 04 e-27	0.000 000 000 95 e-27
kg		
neutron mass energy equivalent	1.505 349 762 87 e-10	0.000 000 000 86 e-10
J		
neutron mass energy equivalent in MeV	939.565 420 52	0.000 000 54
MeV		
neutron mass in u	1.008 664 915 95	0.000 000 000 49
u		
neutron molar mass	1.008 664 915 60 e-3	0.000 000 000 57 e-3
kg mol^-1		
neutron-muon mass ratio	8.892 484 06	0.000 000 20

neutron-proton mag. mom. ratio	-0.684 979 34	0.000 000 16
neutron-proton mass difference	2.305 574 35 e-30	0.000 000 82 e-30
kg		
neutron-proton mass difference energy equivalent	2.072 146 89 e-13	0.000 000 74 e-13
J		
neutron-proton mass difference energy equivalent in MeV	1.293 332 36	0.000 000 46
MeV		
neutron-proton mass difference in u	1.388 449 33 e-3	0.000 000 49 e-3
u		
neutron-proton mass ratio	1.001 378 419 31	0.000 000 000 49
neutron relative atomic mass	1.008 664 915 95	0.000 000 000 49
neutron-tau mass ratio	0.528 779	0.000 036
neutron to shielded proton mag. mom. ratio	-0.684 996 94	0.000 000 16
Newtonian constant of gravitation	6.674 30 e-11	0.000 15 e-11
m^3 kg^-1 s^-2		
Newtonian constant of gravitation over h-bar c	6.708 83 e-39	0.000 15 e-39
(GeV/c^2)^-2		
nuclear magneton	5.050 783 7461 e-27	0.000 000 0015 e-27
J T^-1		
nuclear magneton in eV/T	3.152 451 258 44 e-8	0.000 000 000 96 e-8
eV T^-1		
nuclear magneton in inverse meter per tesla	2.542 623 413 53 e-2	0.000 000 000 78 e-2
m^-1 T^-1		
nuclear magneton in K/T	3.658 267 7756 e-4	0.000 000 0011 e-4
K T^-1		
nuclear magneton in MHz/T	7.622 593 2291	0.000 000 0023
MHz T^-1		
Planck constant	6.626 070 15 e-34	(exact)
J Hz^-1		
Planck constant in eV/Hz	4.135 667 696... e-15	(exact)
eV Hz^-1		
Planck length	1.616 255 e-35	0.000 018 e-35
m		
Planck mass	2.176 434 e-8	0.000 024 e-8
kg		
Planck mass energy equivalent in GeV	1.220 890 e19	0.000 014 e19
GeV		
Planck temperature	1.416 784 e32	0.000 016 e32
K		
Planck time	5.391 247 e-44	0.000 060 e-44
s		
proton charge to mass quotient	9.578 833 1560 e7	0.000 000 0029 e7
C kg^-1		
proton Compton wavelength	1.321 409 855 39 e-15	0.000 000 000 40 e-15
m		
proton-electron mass ratio	1836.152 673 43	0.000 000 11
proton g factor	5.585 694 6893	0.000 000 0016
proton gyromag. ratio	2.675 221 8744 e8	0.000 000 0011 e8
s^-1 T^-1		
proton gyromag. ratio in MHz/T	42.577 478 518	0.000 000 018
MHz T^-1		
proton mag. mom.	1.410 606 797 36 e-26	0.000 000 000 60 e-26
J T^-1		
proton mag. mom. to Bohr magneton ratio	1.521 032 202 30 e-3	0.000 000 000 46 e-3
proton mag. mom. to nuclear magneton ratio	2.792 847 344 63	0.000 000 000 82
proton mag. shielding correction	2.5689 e-5	0.0011 e-5
proton mass	1.672 621 923 69 e-27	0.000 000 000 51 e-27
kg		
proton mass energy equivalent	1.503 277 615 98 e-10	0.000 000 000 46 e-10
J		
proton mass energy equivalent in MeV	938.272 088 16	0.000 000 29
MeV		
proton mass in u	1.007 276 466 621	0.000 000 000 053
u		
proton molar mass	1.007 276 466 27 e-3	0.000 000 000 31 e-3
kg mol^-1		
proton-muon mass ratio	8.880 243 37	0.000 000 20
proton-neutron mag. mom. ratio	-1.459 898 05	0.000 000 34
proton-neutron mass ratio	0.998 623 478 12	0.000 000 000 49
proton relative atomic mass	1.007 276 466 621	0.000 000 000 053
proton rms charge radius	8.414 e-16	0.019 e-16
m		

proton-tau mass ratio	0.528 051	0.000 036
quantum of circulation m^2 s^-1	3.636 947 5516 e-4	0.000 000 0011 e-4
quantum of circulation times 2 m^2 s^-1	7.273 895 1032 e-4	0.000 000 0022 e-4
reduced Compton wavelength m	3.861 592 6796 e-13	0.000 000 0012 e-13
reduced muon Compton wavelength m	1.867 594 306 e-15	0.000 000 042 e-15
reduced neutron Compton wavelength m	2.100 194 1552 e-16	0.000 000 0012 e-16
reduced Planck constant J s	1.054 571 817... e-34	(exact)
reduced Planck constant in eV s eV s	6.582 119 569... e-16	(exact)
reduced Planck constant times c in MeV fm MeV fm	197.326 980 4...	(exact)
reduced proton Compton wavelength m	2.103 089 103 36 e-16	0.000 000 000 64 e-16
reduced tau Compton wavelength m	1.110 538 e-16	0.000 075 e-16
Rydberg constant m^-1	10 973 731.568 160	0.000 021
Rydberg constant times c in Hz Hz	3.289 841 960 2508 e15	0.000 000 000 0064 e15
Rydberg constant times hc in eV eV	13.605 693 122 994	0.000 000 000 026
Rydberg constant times hc in J J	2.179 872 361 1035 e-18	0.000 000 000 0042 e-18
Sackur-Tetrode constant (1 K, 100 kPa)	-1.151 707 537 06	0.000 000 000 45
Sackur-Tetrode constant (1 K, 101.325 kPa)	-1.164 870 523 58	0.000 000 000 45
second radiation constant m K	1.438 776 877... e-2	(exact)
shielded helion gyromag. ratio s^-1 T^-1	2.037 894 569 e8	0.000 000 024 e8
shielded helion gyromag. ratio in MHz/T MHz T^-1	32.434 099 42	0.000 000 38
shielded helion mag. mom. J T^-1	-1.074 553 090 e-26	0.000 000 013 e-26
shielded helion mag. mom. to Bohr magneton ratio	-1.158 671 471 e-3	0.000 000 014 e-3
shielded helion mag. mom. to nuclear magneton ratio	-2.127 497 719	0.000 000 025
shielded helion to proton mag. mom. ratio	-0.761 766 5618	0.000 000 0089
shielded helion to shielded proton mag. mom. ratio	-0.761 786 1313	0.000 000 0033
shielded proton gyromag. ratio s^-1 T^-1	2.675 153 151 e8	0.000 000 029 e8
shielded proton gyromag. ratio in MHz/T MHz T^-1	42.576 384 74	0.000 000 46
shielded proton mag. mom. J T^-1	1.410 570 560 e-26	0.000 000 015 e-26
shielded proton mag. mom. to Bohr magneton ratio	1.520 993 128 e-3	0.000 000 017 e-3
shielded proton mag. mom. to nuclear magneton ratio	2.792 775 599	0.000 000 030
shielding difference of d and p in HD	2.0200 e-8	0.0020 e-8
shielding difference of t and p in HT	2.4140 e-8	0.0020 e-8
speed of light in vacuum m s^-1	299 792 458	(exact)
standard acceleration of gravity m s^-2	9.806 65	(exact)
standard atmosphere Pa	101 325	(exact)
standard-state pressure Pa	100 000	(exact)
Stefan-Boltzmann constant W m^-2 K^-4	5.670 374 419... e-8	(exact)
tau Compton wavelength m	6.977 71 e-16	0.000 47 e-16
tau-electron mass ratio	3477.23	0.23
tau energy equivalent MeV	1776.86	0.12
tau mass kg	3.167 54 e-27	0.000 21 e-27
tau mass energy equivalent J	2.846 84 e-10	0.000 19 e-10
tau mass in u u	1.907 54	0.000 13
tau molar mass	1.907 54 e-3	0.000 13 e-3

kg mol^-1		
tau-muon mass ratio	16.8170	0.0011
tau-neutron mass ratio	1.891 15	0.000 13
tau-proton mass ratio	1.893 76	0.000 13
Thomson cross section	6.652 458 7321 e-29	0.000 000 0060 e-29
m^2		
triton-electron mass ratio	5496.921 535 73	0.000 000 27
triton g factor	5.957 924 931	0.000 000 012
triton mag. mom.	1.504 609 5202 e-26	0.000 000 0030 e-26
J T^-1		
triton mag. mom. to Bohr magneton ratio	1.622 393 6651 e-3	0.000 000 0032 e-3
triton mag. mom. to nuclear magneton ratio	2.978 962 4656	0.000 000 0059
triton mass	5.007 356 7446 e-27	0.000 000 0015 e-27
kg		
triton mass energy equivalent	4.500 387 8060 e-10	0.000 000 0014 e-10
J		
triton mass energy equivalent in MeV	2808.921 132 98	0.000 000 85
MeV		
triton mass in u	3.015 500 716 21	0.000 000 000 12
u		
triton molar mass	3.015 500 715 17 e-3	0.000 000 000 92 e-3
kg mol^-1		
triton-proton mass ratio	2.993 717 034 14	0.000 000 000 15
triton relative atomic mass	3.015 500 716 21	0.000 000 000 12
triton to proton mag. mom. ratio	1.066 639 9191	0.000 000 0021
unified atomic mass unit	1.660 539 066 60 e-27	0.000 000 000 50 e-27
kg		
vacuum electric permittivity	8.854 187 8128 e-12	0.000 000 0013 e-12
F m^-1		
vacuum mag. permeability	1.256 637 062 12 e-6	0.000 000 000 19 e-6
N A^-2		
von Klitzing constant	25 812.807 45...	(exact)
ohm		
weak mixing angle	0.222 90	0.000 30
Wien frequency displacement law constant	5.878 925 757... e10	(exact)
Hz K^-1		
Wien wavelength displacement law constant	2.897 771 955... e-3	(exact)
m K		
W to Z mass ratio	0.881 53	0.000 17

Chapter 9

requirements

gcc>=4.6 or msvc>=14

gfortran>=4.6 or ifort>=18

cmake>=3.20

Chapter 10

Modules Index

10.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 11

Data Type Index

11.1 Data Types List

Here are the data types with brief descriptions:

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

Chapter 13

Module Documentation

13.1 codata Module Reference

Codata constants.

Functions/Subroutines

- pure character(len=4) function, public codata_get_year ()

  Get the set year for the codata constants.
- integer function, public codata_get_number_constants ()

  Get the number of constants.
- subroutine, public codata_print ()

  Display all constants.
- character(len=:) function, allocatable, public codata_get_name_by_index (index)

  Get the name of the constant by index.
- real(real64) function, public codata_get_value_by_index (index)

  Get the value of the constant by index.
- real(real64) function, public codata_get_uncertainty_by_index (index)

  Get the vauncertaintylue of the constant by index.
- character(len=:) function, allocatable, public codata_get_unit_by_index (index)

  Get the unit of the constant by index.
- real(real64) function, public codata_get_value (name)

  Get the value of the constant by name.
- real(real64) function, public codata_get_uncertainty (name)

  Get the uncertainty of the constant by name.
- character(len=25) function, public codata_get_unit (name)

  Get the unit of the constant by name.

13.1.1 Detailed Description

Codata constants.

Codata constants wrapped in an array of derived type with members name, value, uncertainty and unit. Methods for getting the member values are available.

13.1.2 Function/Subroutine Documentation

13.1.2.1 codata_get_name_by_index()

```
character(len=:)  function, allocatable, public codata::codata_get_name_by_index (
            integer, intent(in) index )
```

Get the name of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

name or empty if not found.

13.1.2.2 codata_get_number_constants()

```
integer function, public codata::codata_get_number_constants
```

Get the number of constants.

Returns

Number of constants

13.1.2.3 codata_get_uncertainty()

```
real(real64) function, public codata::codata_get_uncertainty (
            character(len=*), intent(in) name )
```

Get the uncertainty of the constant by name.

Parameters

in	*name*	Name of the constant

Returns

value or NaN if not found

13.1.2.4 codata_get_uncertainty_by_index()

```
real(real64) function, public codata::codata_get_uncertainty_by_index (
             integer, intent(in) index )
```

Get the vauncertaintylue of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

13.1.2.5 codata_get_unit()

```
character(len=25) function, public codata::codata_get_unit (
             character(len=*), intent(in) name )
```

Get the unit of the constant by name.

Parameters

in	*name*	Name of the constant

Returns

unit or None if not found

13.1.2.6 codata_get_unit_by_index()

```
character(len=:)  function, allocatable, public codata::codata_get_unit_by_index (
             integer, intent(in) index )
```

Get the unit of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

name or empty if not found.

13.1.2.7 codata_get_value()

```
real(real64) function, public codata::codata_get_value (
             character(len=*), intent(in) name )
```

Get the value of the constant by name.

Parameters

in	*name*	Name of the constant

Returns

value or NaN if not found

13.1.2.8 codata_get_value_by_index()

```
real(real64) function, public codata::codata_get_value_by_index (
             integer, intent(in) index )
```

Get the value of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

13.1.2.9 codata_get_year()

```
pure character(len=4) function, public codata::codata_get_year
```

Get the set year for the codata constants.

Returns

Year of the codata constants

13.1.2.10 codata_print()

```
subroutine, public codata::codata_print
```

Display all constants.

13.2 codata_capi Module Reference

C API for the codata constants.

Functions/Subroutines

- type(c_ptr) function codata_capi_get_year ()
  Get the set year for the codata constants return Year of the codata constants.
- integer(c_int) function codata_capi_get_number_constants ()
  Get the number of constants.
- subroutine codata_capi_print ()
  Display all constants.
- type(c_ptr) function codata_capi_get_name_by_index (index)
  Get the name of the constant by index.
- real(c_double) function codata_capi_get_value_by_index (index)
  Get the value of the constant by index.
- real(c_double) function codata_capi_get_uncertainty_by_index (index)
  Get the uncertainty of the constant by index.
- type(c_ptr) function codata_capi_get_unit_by_index (index)
  Get the unit of the constant by index.
- real(c_double) function codata_capi_get_value (char_p, length)
  Get the value of the constant by name.
- real(c_double) function codata_capi_get_uncertainty (char_p, length)
  Get the uncertainty of the constant by name.
- type(c_ptr) function codata_capi_get_unit (char_p, length)
  Get the unit of the constant by name.

Variables

- character(len=:), allocatable, target capi_name
  Allocatable for a Fortran string representing the name of the constant.
- character(len=:), pointer capi_name_ptr
  Fortran pointer to the string representing the name of the constant.
- character(len=:), allocatable, target capi_unit
  Allocatable for a Fortran string representing the unit of the constant.
- character(len=:), pointer capi_unit_ptr
  Fortran pointer to the string representing the unit of the constant.
- character(len=:), allocatable, target capi_year
  Allocatable for a Fortran string representing the codata year.
- character(len=:), pointer capi_year_ptr
  Fortran pointer to the string representing the codata year.

13.2.1 Detailed Description

C API for the codata constants.

Provide C compatible getters and setters for accessing the codata constants.

13.2.2 Function/Subroutine Documentation

13.2.2.1 codata_capi_get_name_by_index()

```
type(c_ptr) function codata_capi::codata_capi_get_name_by_index (
             integer(c_int), intent(in), value index )
```

Get the name of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

name or empty if not found.

13.2.2.2 codata_capi_get_number_constants()

```
integer(c_int) function codata_capi::codata_capi_get_number_constants
```

Get the number of constants.

Returns

Number of constants

13.2.2.3 codata_capi_get_uncertainty()

```
real(c_double) function codata_capi::codata_capi_get_uncertainty (
             type(c_ptr), intent(in), value char_p,
             integer(c_int), intent(in), value length )
```

Get the uncertainty of the constant by name.

Parameters

in	*char_p*	Name of the constant
in	*length*	Length of the string

Returns

value or NaN if not found

13.2.2.4 codata_capi_get_uncertainty_by_index()

```
real(c_double) function codata_capi::codata_capi_get_uncertainty_by_index (
            integer(c_int), intent(in), value index )
```

Get the uncertainty of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

13.2.2.5 codata_capi_get_unit()

```
type(c_ptr) function codata_capi::codata_capi_get_unit (
            type(c_ptr), intent(in), value char_p,
            integer(c_int), intent(in), value length )
```

Get the unit of the constant by name.

Parameters

in	*char_p*	Name of the constant
in	*length*	Length of the string

Returns

unit or None if not found

13.2.2.6 codata_capi_get_unit_by_index()

```
type(c_ptr) function codata_capi::codata_capi_get_unit_by_index (
             integer(c_int), intent(in), value index )
```

Get the unit of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

name or empty if not found.

13.2.2.7 codata_capi_get_value()

```
real(c_double) function codata_capi::codata_capi_get_value (
             type(c_ptr), intent(in), value char_p,
             integer(c_int), intent(in), value length )
```

Get the value of the constant by name.

Parameters

in	*char_p*	Name of the constant
in	*length*	Length of the string

Returns

value or NaN if not found

13.2.2.8 codata_capi_get_value_by_index()

```
real(c_double) function codata_capi::codata_capi_get_value_by_index (
             integer(c_int), intent(in), value index )
```

Get the value of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

13.2.2.9 codata_capi_get_year()

```
type(c_ptr) function codata_capi::codata_capi_get_year
```

Get the set year for the codata constants return Year of the codata constants.

13.2.2.10 codata_capi_print()

```
subroutine codata_capi::codata_capi_print
```

Display all constants.

13.2.3 Variable Documentation

13.2.3.1 capi_name

```
character(len=:), allocatable, target codata_capi::capi_name
```

Allocatable for a Fortran string representing the name of the constant.

It is used for interoperability Fortran and C strings.

13.2.3.2 capi_name_ptr

```
character(len=:), pointer codata_capi::capi_name_ptr
```

Fortran pointer to the string representing the name of the constant.

It is used for interoperability Fortran and C strings.

13.2.3.3 capi_unit

```
character(len=:), allocatable, target codata_capi::capi_unit
```

Allocatable for a Fortran string representing the unit of the constant.

It is used for interoperability Fortran and C strings.

13.2.3.4 capi_unit_ptr

```
character(len=:), pointer codata_capi::capi_unit_ptr
```

Fortran pointer to the string representing the unit of the constant.

It is used for interoperability Fortran and C strings.

13.2.3.5 capi_year

```
character(len=:), allocatable, target codata_capi::capi_year
```

Allocatable for a Fortran string representing the codata year.

It is used for interoperability Fortran and C strings.

13.2.3.6 capi_year_ptr

```
character(len=:), pointer codata_capi::capi_year_ptr
```

Fortran pointer to the string representing the codata year.

It is used for interoperability Fortran and C strings.

13.3 codata_data Module Reference

Codata constants - autogenerated.

Variables

- type(codata_t_constant), dimension(10), parameter codata10 = [codata_t_constant("alpha particle-electron mass ratio", 7294.29954142d0, 0.00000024d0, " ") , codata_t_constant("alpha particle mass", 6.6446573357d-27, 0.0000000020d-27, "kg") , codata_t_constant("alpha particle mass energy equivalent", 5.9719201914d-10, 0.0000000018d-10, "J") , codata_t_constant("alpha particle mass energy equivalent in MeV", 3727.3794066d0, 0.0000011d0, "MeV") , codata_t_constant("alpha particle mass in u", 4.001506179127d0, 0.000000000063d0, "u") , codata_t_constant("alpha particle molar mass", 4.↩ 0015061777d-3, 0.0000000012d-3, "kg mol^-1") , codata_t_constant("alpha particle-proton mass ratio", 3.97259969009d0, 0.00000000022d0, " ") , codata_t_constant("alpha particle relative atomic mass", 4.↩ 001506179127d0, 0.000000000063d0, " ") , codata_t_constant("Angstrom star", 1.00001495d-10, 0.↩ 00000090d-10, "m") , codata_t_constant("atomic mass constant", 1.66053906660d-27, 0.00000000050d-27, "kg")]
- type(codata_t_constant), dimension(354), public codata_constants = [codata10 ,codata20 ,codata30 ,codata40 ,codata50 ,codata60 ,codata70 ,codata80 ,codata90 ,codata100 ,codata110 ,codata120 ,codata130 ,codata140 ,codata150 ,codata160 ,codata170 ,codata180 ,codata190 ,codata200 ,codata210 ,codata220 ,codata230 ,codata240 ,codata250 ,codata260 ,codata270 ,codata280 ,codata290 ,codata300 ,codata310 ,codata320 ,codata330 ,codata340 ,codata350 ,codata354]

13.3.1 Detailed Description

Codata constants - autogenerated.

13.3.2 Variable Documentation

13.3.2.1 codata10

```
type(codata_t_constant), dimension(10), parameter codata_data::codata10 = [ codata_t_constant("alpha
particle-electron mass ratio", 7294.29954142d0, 0.00000024d0, " ") , codata_t_constant("alpha
particle mass", 6.6446573357d-27, 0.0000000020d-27, "kg") , codata_t_constant("alpha particle
mass energy equivalent", 5.9719201914d-10, 0.0000000018d-10, "J") , codata_t_constant("alpha
particle mass energy equivalent in MeV", 3727.3794066d0, 0.0000011d0, "MeV") , codata_t↩
_constant("alpha particle mass in u", 4.001506179127d0, 0.000000000063d0, "u") , codata↩
_t_constant("alpha particle molar mass", 4.0015061777d-3, 0.0000000012d-3, "kg mol^-1") ,
codata_t_constant("alpha particle-proton mass ratio", 3.97259969009d0, 0.00000000022d0, " ")
, codata_t_constant("alpha particle relative atomic mass", 4.001506179127d0, 0.000000000063d0,
" ") , codata_t_constant("Angstrom star", 1.00001495d-10, 0.00000090d-10, "m") , codata_t_↩
constant("atomic mass constant", 1.66053906660d-27, 0.00000000050d-27, "kg") ]
```

13.3.2.2 codata_constants

```
type(codata_t_constant), dimension(354), public codata_data::codata_constants = [codata10
,codata20 ,codata30 ,codata40 ,codata50 ,codata60 ,codata70 ,codata80 ,codata90 ,codata100
,codata110 ,codata120 ,codata130 ,codata140 ,codata150 ,codata160 ,codata170 ,codata180 ,codata190
,codata200 ,codata210 ,codata220 ,codata230 ,codata240 ,codata250 ,codata260 ,codata270 ,codata280
,codata290 ,codata300 ,codata310 ,codata320 ,codata330 ,codata340 ,codata350 ,codata354 ]
```

Chapter 14

Data Type Documentation

14.1 codata_file_props Struct Reference

Properties of the file for the codata raw data.

Data Fields

- int n
- int index_header_end
- char codata_path [18]
- char year [5]
- char fmodule_path [18]

14.1.1 Detailed Description

Properties of the file for the codata raw data.

14.1.2 Field Documentation

14.1.2.1 codata_path

```
char codata_file_props::codata_path[18]
```

Filepath to the raw codata constants.

14.1.2.2 fmodule_path

```
char codata_file_props::fmodule_path[18]
```

Filepath of the generated Fortran module.

14.1.2.3 index_header_end

```
int codata_file_props::index_header_end
```

Number of lines for the header.

14.1.2.4 n

```
int codata_file_props::n
```

Number of lines.

14.1.2.5 year

```
char codata_file_props::year[5]
```

Year of release of the codata constants.

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

- src/generator.c

Chapter 15

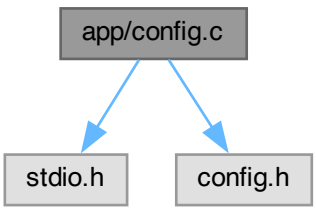
File Documentation

15.1 app/config.c File Reference

Provides the configuration of the codata library.

```
#include <stdio.h>
#include "config.h"
```

Include dependency graph for config.c:



Functions

- int main (int argc, char **argv)

15.1.1 Detailed Description

Provides the configuration of the codata library.

15.1.2 Function Documentation

15.1.2.1 main()

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

15.2 doxygen/introduction/install.md File Reference

15.3 doxygen/introduction/license.md File Reference

15.4 doxygen/introduction/raw_codata.md File Reference

15.5 doxygen/introduction/requirements.md File Reference

15.6 doxygen/releases/0.1.0-notes.md File Reference

15.7 doxygen/releases/0.2.0-notes.md File Reference

15.8 doxygen/releases/0.2.1-notes.md File Reference

15.9 doxygen/releases/0.3.0-notes.md File Reference

15.10 README.md File Reference

15.11 src/codata.f90 File Reference

Codata module.

Modules

- module codata

  Codata constants.

Functions/Subroutines

- pure character(len=4) function, public codata::codata_get_year ()
  Get the set year for the codata constants.
- integer function, public codata::codata_get_number_constants ()
  Get the number of constants.
- subroutine, public codata::codata_print ()
  Display all constants.
- character(len=:) function, allocatable, public codata::codata_get_name_by_index (index)
  Get the name of the constant by index.
- real(real64) function, public codata::codata_get_value_by_index (index)
  Get the value of the constant by index.
- real(real64) function, public codata::codata_get_uncertainty_by_index (index)
  Get the vauncertaintylue of the constant by index.
- character(len=:) function, allocatable, public codata::codata_get_unit_by_index (index)
  Get the unit of the constant by index.
- real(real64) function, public codata::codata_get_value (name)
  Get the value of the constant by name.
- real(real64) function, public codata::codata_get_uncertainty (name)
  Get the uncertainty of the constant by name.
- character(len=25) function, public codata::codata_get_unit (name)
  Get the unit of the constant by name.

15.11.1 Detailed Description

Codata module.

15.12 src/codata.h File Reference

C header for the codata library.

Functions

- char * codata_capi_get_year ()
  Get the set year for the codata constants return Year of the codata constants.
- int codata_capi_get_number_constants ()
  Get the number of constants.
- void codata_capi_print ()
  Display all constants.
- double codata_capi_get_value (char *name, int length)
  Get the value of the constant by name.
- double codata_capi_get_uncertainty (char *name, int length)
  Get the uncertainty of the constant by name.
- char * codata_capi_get_unit (char *name, int length)
  Get the unit of the constant by name.
- double codata_capi_get_value_by_index (int index)
  Get the value of the constant by index.
- double codata_capi_get_uncertainty_by_index (int index)
  Get the uncertainty of the constant by index.
- char * codata_capi_get_name_by_index (int index)
  Get the name of the constant by index.
- char * codata_capi_get_unit_by_index (int index)
  Get the unit of the constant by index.

15.12.1 Detailed Description

C header for the codata library.

15.12.2 Function Documentation

15.12.2.1 codata_capi_get_name_by_index()

```
char * codata_capi_get_name_by_index (
            int index )
```

Get the name of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

name or None if not found.

15.12.2.2 codata_capi_get_number_constants()

```
int codata_capi_get_number_constants ( )
```

Get the number of constants.

Returns

Number of the constants.

15.12.2.3 codata_capi_get_uncertainty()

```
double codata_capi_get_uncertainty (
            char * name,
            int length )
```

Get the uncertainty of the constant by name.

Parameters

in	*name*	Name of the constant
in	*length*	Length of the string

Returns

value or NaN if not found

15.12.2.4 codata_capi_get_uncertainty_by_index()

```
double codata_capi_get_uncertainty_by_index (
            int index )
```

Get the uncertainty of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

15.12.2.5 codata_capi_get_unit()

```
char * codata_capi_get_unit (
            char * name,
            int length )
```

Get the unit of the constant by name.

Parameters

in	*name*	Name of the constant
in	*length*	Length of the string

Returns

unit or None if not found

15.12.2.6 codata_capi_get_unit_by_index()

```
char * codata_capi_get_unit_by_index (
            int index )
```

Get the unit of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

unit or None if not found.

15.12.2.7 codata_capi_get_value()

```
double codata_capi_get_value (
            char * name,
            int length )
```

Get the value of the constant by name.

Parameters

in	*name*	Name of the constant
in	*length*	Length of the string

Returns

value or NaN if not found

Examples

example_in_c.c.

15.12.2.8 codata_capi_get_value_by_index()

```
double codata_capi_get_value_by_index (
            int index )
```

Get the value of the constant by index.

Parameters

in	*index*	Index of the position.

Returns

value or NaN if not found.

15.12.2.9 codata_capi_get_year()

```
char * codata_capi_get_year ( )
```

Get the set year for the codata constants return Year of the codata constants.

Examples

example_in_c.c.

15.12.2.10 codata_capi_print()

```
void codata_capi_print ( )
```

Display all constants.

Examples

example_in_c.c.

15.13 codata.h

Go to the documentation of this file.

```
00001
00013 extern char *codata_capi_get_year();
00014
00019 extern int codata_capi_get_number_constants();
00020
00021
00025 extern void codata_capi_print();
00026
00027
00034 extern double codata_capi_get_value(char *name, int length);
00035
00042 extern double codata_capi_get_uncertainty(char *name, int length);
00043
00050 extern char * codata_capi_get_unit(char *name, int length);
00051
00057 extern double codata_capi_get_value_by_index(int index);
00058
00064 extern double codata_capi_get_uncertainty_by_index(int index);
00065
00066
00072 extern char* codata_capi_get_name_by_index(int index);
00073
00079 extern char* codata_capi_get_unit_by_index(int index);
```

15.14 src/codata_capi.f90 File Reference

Codata module - C API.

Modules

- module codata_capi

  C API for the codata constants.

Functions/Subroutines

- type(c_ptr) function codata_capi::codata_capi_get_year ()

  Get the set year for the codata constants return Year of the codata constants.
- integer(c_int) function codata_capi::codata_capi_get_number_constants ()

  Get the number of constants.
- subroutine codata_capi::codata_capi_print ()

  Display all constants.
- type(c_ptr) function codata_capi::codata_capi_get_name_by_index (index)

  Get the name of the constant by index.
- real(c_double) function codata_capi::codata_capi_get_value_by_index (index)

  Get the value of the constant by index.
- real(c_double) function codata_capi::codata_capi_get_uncertainty_by_index (index)

  Get the uncertainty of the constant by index.
- type(c_ptr) function codata_capi::codata_capi_get_unit_by_index (index)

  Get the unit of the constant by index.
- real(c_double) function codata_capi::codata_capi_get_value (char_p, length)

  Get the value of the constant by name.
- real(c_double) function codata_capi::codata_capi_get_uncertainty (char_p, length)

  Get the uncertainty of the constant by name.
- type(c_ptr) function codata_capi::codata_capi_get_unit (char_p, length)

  Get the unit of the constant by name.

Variables

- character(len=:), allocatable, target codata_capi::capi_name

  Allocatable for a Fortran string representing the name of the constant.
- character(len=:), pointer codata_capi::capi_name_ptr

  Fortran pointer to the string representing the name of the constant.
- character(len=:), allocatable, target codata_capi::capi_unit

  Allocatable for a Fortran string representing the unit of the constant.
- character(len=:), pointer codata_capi::capi_unit_ptr

  Fortran pointer to the string representing the unit of the constant.
- character(len=:), allocatable, target codata_capi::capi_year

  Allocatable for a Fortran string representing the codata year.
- character(len=:), pointer codata_capi::capi_year_ptr

  Fortran pointer to the string representing the codata year.

15.14.1 Detailed Description

Codata module - C API.

15.15 src/codata_data.f90 File Reference

Codata module - autogenerated.

Modules

- module codata_data

  Codata constants - autogenerated.

Variables

- type(codata_t_constant), dimension(10), parameter codata_data::codata10 = [codata_t_constant("alpha particle-electron mass ratio", 7294.29954142d0, 0.00000024d0, " ") , codata_t_constant("alpha particle mass", 6.6446573357d-27, 0.0000000020d-27, "kg") , codata_t_constant("alpha particle mass energy equivalent", 5.9719201914d-10, 0.0000000018d-10, "J") , codata_t_constant("alpha particle mass energy equivalent in MeV", 3727.3794066d0, 0.0000011d0, "MeV") , codata_t_constant("alpha particle mass in u", 4.001506179127d0, 0.000000000063d0, "u") , codata_t_constant("alpha particle molar mass", 4.↩ 0015061777d-3, 0.0000000012d-3, "kg mol^-1") , codata_t_constant("alpha particle-proton mass ratio", 3.97259969009d0, 0.00000000022d0, " ") , codata_t_constant("alpha particle relative atomic mass", 4.↩ 001506179127d0, 0.000000000063d0, " ") , codata_t_constant("Angstrom star", 1.00001495d-10, 0.↩ 00000090d-10, "m") , codata_t_constant("atomic mass constant", 1.66053906660d-27, 0.00000000050d-27, "kg")]
- type(codata_t_constant), dimension(354), public codata_data::codata_constants = [codata10 ,codata20 ,codata30 ,codata40 ,codata50 ,codata60 ,codata70 ,codata80 ,codata90 ,codata100 ,codata110 ,codata120 ,codata130 ,codata140 ,codata150 ,codata160 ,codata170 ,codata180 ,codata190 ,codata200 ,codata210 ,codata220 ,codata230 ,codata240 ,codata250 ,codata260 ,codata270 ,codata280 ,codata290 ,codata300 ,codata310 ,codata320 ,codata330 ,codata340 ,codata350 ,codata354]

15.15.1 Detailed Description

Codata module - autogenerated.

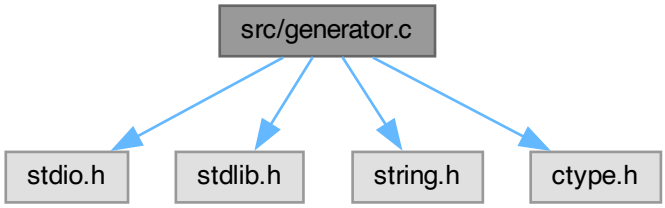
15.16 src/generator.c File Reference

Generator for Fortran module.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <ctype.h>
```

Include dependency graph for generator.c:



Data Structures

- struct codata_file_props

  Properties of the file for the codata raw data.

Functions

- void format_names (char *line, char *name)

  Format names simply by copying them.
- void format_values (char *line, char *value)

  Format values to be conform to Fortran double precision.
- void format_uncertainties (char *line, char *uncertainty)

  Format the uncertainties to be conform to Fortran double precsion.
- void format_units (char *line, char *unit)

  Format the units to be conform to Fortran strings.
- void clean_line (char *buf, size_t buffer_size)

  Fill the buffer with white space.
- int read_line (FILE *f, char *buf, size_t buffer_size)

  Read the line from f and copy in buf.
- void ltrim (char *buf, size_t buffer_size)

  Remove all white space from the left.
- void rtrim (char *buf, size_t buffer_size)

  Remove all white space from the right.
- int is_blank_line (char *buf, size_t buffer_size)

  Test if the line is a blank line.
- void get_props (struct codata_file_props *props)

  Get the properties of the codata file.
- void print_props (struct codata_file_props *props)

  Print the codata file properties.
- void write_file_doc (FILE *fcode)

  Generate the Fortran file documentation.
- void write_module_doc (FILE *fcode)

  Generate the Fortran module documentation.

- void write_module_declaration (FILE *fcode, struct codata_file_props *props)

  Generate the Fortran module declaration.
- void write_all_constants (FILE *fcodata, FILE *fcode, struct codata_file_props *props)

  Generate all constants in the Fortran module.
- void write_module_end (FILE *fcode, struct codata_file_props *props)

  Generate the end of the Fortran module.
- int main (int argc, char **argv)

  Generated Fortran module.

15.16.1 Detailed Description

Generator for Fortran module.

The raw data from NIST are parsed line by line where the columns name, value, uncertainty and unit are formatted to be conform to Fortran. The formatted (as strings) names, values, uncertainties and units are then inserted in a derived type in the generated Fortran module. The raw codata from http://physics.nist.gov/constants are converted into Fortran code.

15.16.2 Function Documentation

15.16.2.1 clean_line()

```
void clean_line (
            char * buf,
            size_t buffer_size )
```

Fill the buffer with white space.

Parameters

buf	Line to be cleaned
buffer_size	Size of the line.

15.16.2.2 format_names()

```
void format_names (
            char * line,
            char * name )
```

Format names simply by copying them.

Parameters

line	Line to be parsed.
name	String where the name will be copied.

15.16.2.3 format_uncertainties()

```
void format_uncertainties (
            char * line,
            char * uncertainty )
```

Format the uncertainties to be conform to Fortran double precsion.

Parameters

line	Line to be parsed.
uncertainty	String where the uncertainty will be copied.

15.16.2.4 format_units()

```
void format_units (
            char * line,
            char * unit )
```

Format the units to be conform to Fortran strings.

Parameters

line	Line to be parsed.
unit	String where the unit will be copied.

15.16.2.5 format_values()

```
void format_values (
            char * line,
            char * value )
```

Format values to be conform to Fortran double precision.

Parameters

line	Line to be parsed.
value	String where the value will be copied.

15.16.2.6 get_props()

```
void get_props (
            struct codata_file_props * props )
```

Get the properties of the codata file.

Parameters

props	Properties of the codata file.

15.16.2.7 is_blank_line()

```
int is_blank_line (
            char * buf,
            size_t buffer_size )
```

Test if the line is a blank line.

Parameters

buf	Line to be tested.
buffer_size	Size of the line.

Returns

int Flag indicating if blank(=1) or not (=0).

15.16.2.8 ltrim()

```
void ltrim (
            char * buf,
            size_t buffer_size )
```

Remove all white space from the left.

Parameters

buf	Line to be left trimmed.
buffer_size	Size of the line.

15.16.2.9 main()

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

Generated Fortran module.

Parameters

argc	Number of arguments
argv	List of arguments

Returns

int Exit flag.

Examples

example_in_c.c.

15.16.2.10 print_props()

```
void print_props (
            struct codata_file_props * props )
```

Print the codata file properties.

Parameters

props	Properties of the codata file.

15.16.2.11 read_line()

```
int read_line (
            FILE * f,
            char * buf,
            size_t buffer_size )
```

Read the line from f and copy in buf.

Parameters

f	File pointer where the line will be parsed.
buf	String where the line will be copied.
buffer_size	Size of the buffer.

Returns

int Flag if the line is empty(=1) or not empty(=0).

15.16.2.12 rtrim()

```
void rtrim (
            char * buf,
            size_t buffer_size )
```

Remove all white space from the right.

Parameters

buf	Line to be right trimmed.
buffer_size	Size of the line.

15.16.2.13 write_all_constants()

```
void write_all_constants (
            FILE * fcodata,
            FILE * fcode,
            struct codata_file_props * props )
```

Generate all constants in the Fortran module.

Parameters

fcodata	File pointer to the codata file.
fcode	File pointer to the Fortran module.
props	Properties of the codata file.

15.16.2.14 write_file_doc()

```
void write_file_doc (
            FILE * fcode )
```

Generate the Fortran file documentation.

Parameters

fcode	File pointer of the Fortran module.

15.16.2.15 write_module_declaration()

```
void write_module_declaration (
            FILE * fcode,
            struct codata_file_props * props )
```

Generate the Fortran module declaration.

Parameters

fcode	File pointer of the Fortran module.
props	Properties of the codata file.

15.16.2.16 write_module_doc()

```
void write_module_doc (
            FILE * fcode )
```

Generate the Fortran module documentation.

Parameters

fcode	File pointer of the Fortran module.

15.16.2.17 write_module_end()

```
void write_module_end (
            FILE * fcode,
            struct codata_file_props * props )
```

Generate the end of the Fortran module.

Parameters

fcode	File pointer to the Fortran module.
props	Properties of the codata file.

Chapter 16

Example Documentation

16.1 example_in_fortran.f90

```
00001 program example_in_fortran
00002     use codata
00003     implicit none
00004
00005     character(len=4) :: year = "2014"
00006
00007     ! call directly codata, the values used will be the last i.e. 2018
00008     call codata_print()
00009     print *, "Codata ", codata_get_year(), codata_get_value("alpha particle mass")
00010     print *, codata_get_year()
00011
00012 end program
```

16.2 example_in_c.c

```
#include <stdio.h>
#include <string.h>
#include "codata.h"

int main(int argc, char **argv){

    char year[5] = "2014";

    // avoid compiler complaining
    if (argc>1){
        printf("%d %s", argc, argv[1]);
    }

    /* call directly codata, the values used will be the last i.e. 2018 */
    codata_capi_print();
    char name[] = "alpha particle mass";
    printf("Codata %s: %+23.16e\n", codata_capi_get_year(), codata_capi_get_value(name, strlen(name)));
    printf("%s\n", codata_capi_get_year());


    return 0;

}
```

Index