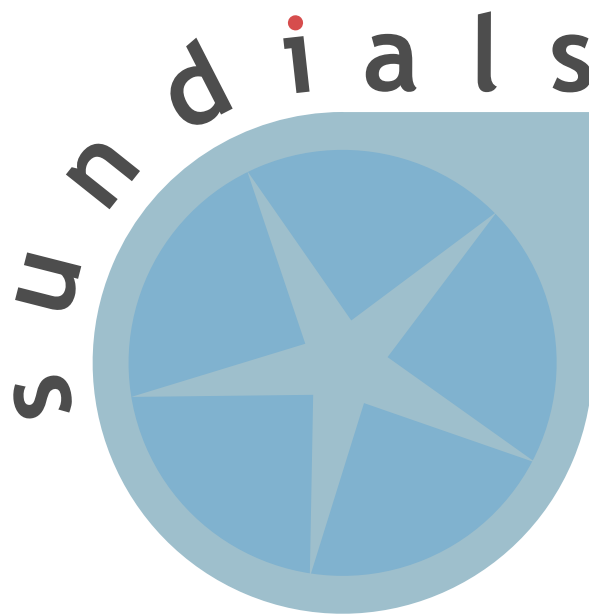


Installation Guide for SUNDIALS v3.2.0

Eddy Banks, Aaron M. Collier, David J. Gardner, Alan C. Hindmarsh,
Radu Serban, and Carol S. Woodward
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

September 28, 2018



UCRL-SM-208116

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Contents

1	SUNDIALS Package Installation Procedure	1
1.1	CMake-based installation	2
1.1.1	Configuring, building, and installing on Unix-like systems	2
1.1.2	Configuration options (Unix/Linux)	4
1.1.3	Configuration examples	10
1.1.4	Working with external Libraries	11
1.1.5	Testing the build and installation	13
1.2	Building and Running Examples	13
1.3	Configuring, building, and installing on Windows	14
1.4	Installed libraries and exported header files	14

Chapter 1

SUNDIALS Package Installation Procedure

The installation of any SUNDIALS package is accomplished by installing the SUNDIALS suite as a whole, according to the instructions that follow. The same procedure applies whether or not the downloaded file contains one or all solvers in SUNDIALS.

The SUNDIALS suite (or individual solvers) are distributed as compressed archives (`.tar.gz`). The name of the distribution archive is of the form `solver-x.y.z.tar.gz`, where *solver* is one of: `sundials`, `cvode`, `cvodes`, `arkode`, `ida`, `idas`, or `kinsol`, and `x.y.z` represents the version number (of the SUNDIALS suite or of the individual solver). To begin the installation, first uncompress and expand the sources, by issuing

```
% tar xzf solver-x.y.z.tar.gz
```

This will extract source files under a directory `solver-x.y.z`.

Starting with version 2.6.0 of SUNDIALS, CMake is the only supported method of installation. The explanations of the installation procedure begins with a few common observations:

- The remainder of this chapter will follow these conventions:

solverdir is the directory `solver-x.y.z` created above; i.e., the directory containing the SUNDIALS sources.

builddir is the (temporary) directory under which SUNDIALS is built.

instdir is the directory under which the SUNDIALS exported header files and libraries will be installed. Typically, header files are exported under a directory `instdir/include` while libraries are installed under `instdir/lib`, with *instdir* specified at configuration time.

- For SUNDIALS CMake-based installation, in-source builds are prohibited; in other words, the build directory *builddir* can **not** be the same as *solverdir* and such an attempt will lead to an error. This prevents “polluting” the source tree and allows efficient builds for different configurations and/or options.
- The installation directory *instdir* can **not** be the same as the source directory *solverdir*.
- By default, only the libraries and header files are exported to the installation directory *instdir*. If enabled by the user (with the appropriate toggle for CMake), the examples distributed with SUNDIALS will be built together with the solver libraries but the installation step will result in exporting (by default in a subdirectory of the installation directory) the example sources and sample outputs together with automatically generated configuration files that reference the *installed* SUNDIALS headers and libraries. As such, these configuration files for the SUNDIALS examples can be used as “templates” for your own problems. CMake installs `CMakeLists.txt` files and also (as an option available only under Unix/Linux) `Makefile` files. Note this installation



approach also allows the option of building the SUNDIALS examples without having to install them. (This can be used as a sanity check for the freshly built libraries.)

- Even if generation of shared libraries is enabled, only static libraries are created for the FCMIX modules. (Because of the use of fixed names for the Fortran user-provided subroutines, FCMIX shared libraries would result in “undefined symbol” errors at link time.)

1.1 CMake-based installation

CMake-based installation provides a platform-independent build system. CMake can generate Unix and Linux Makefiles, as well as KDevelop, Visual Studio, and (Apple) XCode project files from the same configuration file. In addition, CMake also provides a GUI front end and which allows an interactive build and installation process.

The SUNDIALS build process requires CMake version 3.1.3 or higher and a working C compiler. On Unix-like operating systems, it also requires Make (and **curses**, including its development libraries, for the GUI front end to CMake, **ccmake**), while on Windows it requires Visual Studio. CMake is continually adding new features, and the latest version can be downloaded from <http://www.cmake.org>. Build instructions for CMake (only necessary for Unix-like systems) can be found on the CMake website. Once CMake is installed, Linux/Unix users will be able to use **ccmake**, while Windows users will be able to use **CMakeSetup**.

As previously noted, when using CMake to configure, build and install SUNDIALS, it is always required to use a separate build directory. While in-source builds are possible, they are explicitly prohibited by the SUNDIALS CMake scripts (one of the reasons being that, unlike autotools, CMake does not provide a **make distclean** procedure and it is therefore difficult to clean-up the source tree after an in-source build). By ensuring a separate build directory, it is an easy task for the user to clean-up all traces of the build by simply removing the build directory. CMake does generate a **make clean** which will remove files generated by the compiler and linker.

1.1.1 Configuring, building, and installing on Unix-like systems

The default CMake configuration will build all included solvers and associated examples and will build static and shared libraries. The *installdir* defaults to */usr/local* and can be changed by setting the **CMAKE_INSTALL_PREFIX** variable. Support for FORTRAN and all other options are disabled.

CMake can be used from the command line with the **cmake** command, or from a **curses**-based GUI by using the **ccmake** command. Examples for using both methods will be presented. For the examples shown it is assumed that there is a top level SUNDIALS directory with appropriate source, build and install directories:

```
% mkdir (...)sundials/instldir
% mkdir (...)sundials/buildldir
% cd (...)sundials/buildldir
```

Building with the GUI

Using CMake with the GUI follows this general process:

- Select and modify values, run configure (c key)
- New values are denoted with an asterisk
- To set a variable, move the cursor to the variable and press enter
 - If it is a boolean (ON/OFF) it will toggle the value
 - If it is string or file, it will allow editing of the string
 - For file and directories, the <tab> key can be used to complete

- Repeat until all values are set as desired and the generate option is available (g key)
- Some variables (advanced variables) are not visible right away
- To see advanced variables, toggle to advanced mode (t key)
- To search for a variable press / key, and to repeat the search, press the n key

To build the default configuration using the GUI, from the *builddir* enter the `ccmake` command and point to the *solverdir*:

```
% ccmake ../solverdir
```

The default configuration screen is shown in Figure 1.1.

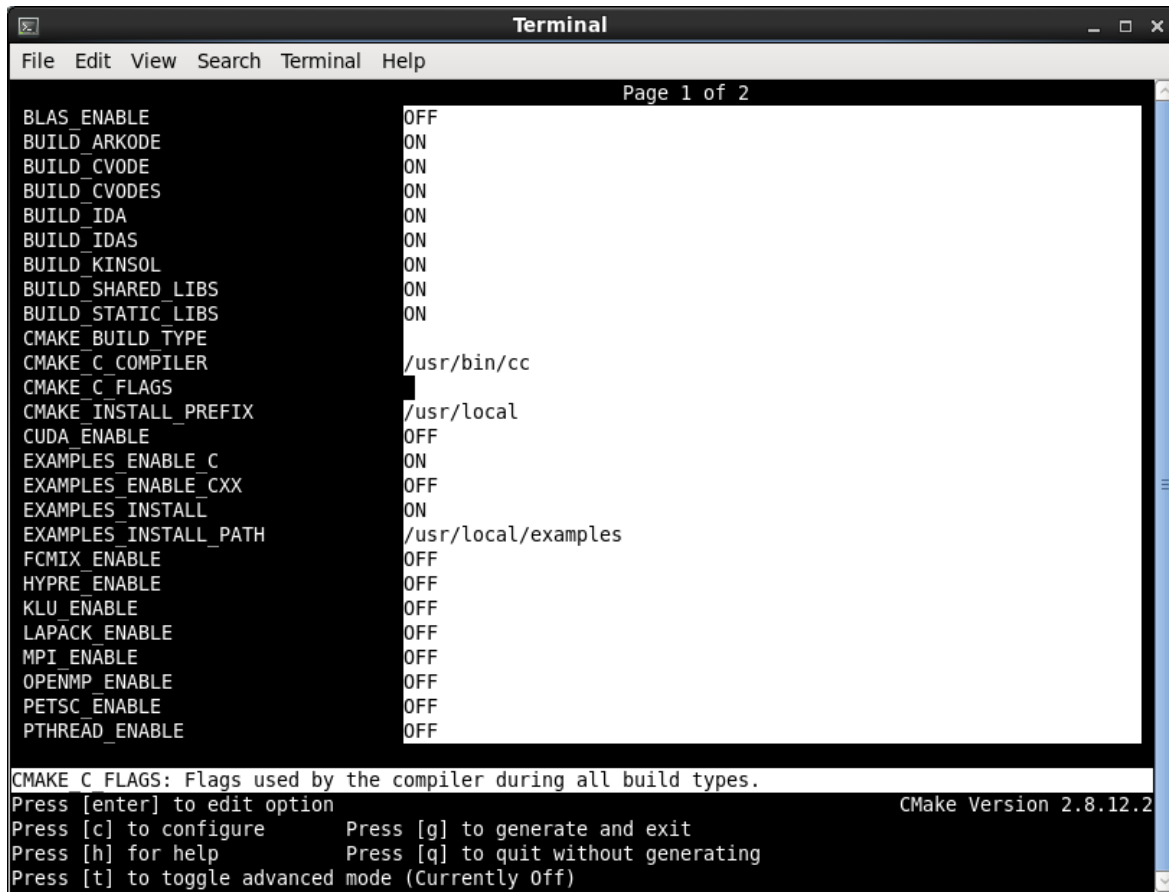


Figure 1.1: Default configuration screen. Note: Initial screen is empty. To get this default configuration, press 'c' repeatedly (accepting default values denoted with asterisk) until the 'g' option is available.

The default *instldir* for both SUNDIALS and corresponding examples can be changed by setting the `CMAKE_INSTALL_PREFIX` and the `EXAMPLES_INSTALL_PATH` as shown in figure 1.2.

Pressing the (g key) will generate makefiles including all dependencies and all rules to build SUNDIALS on this system. Back at the command prompt, you can now run:

```
% make
```

To install SUNDIALS in the installation directory specified in the configuration, simply run:

```
% make install
```

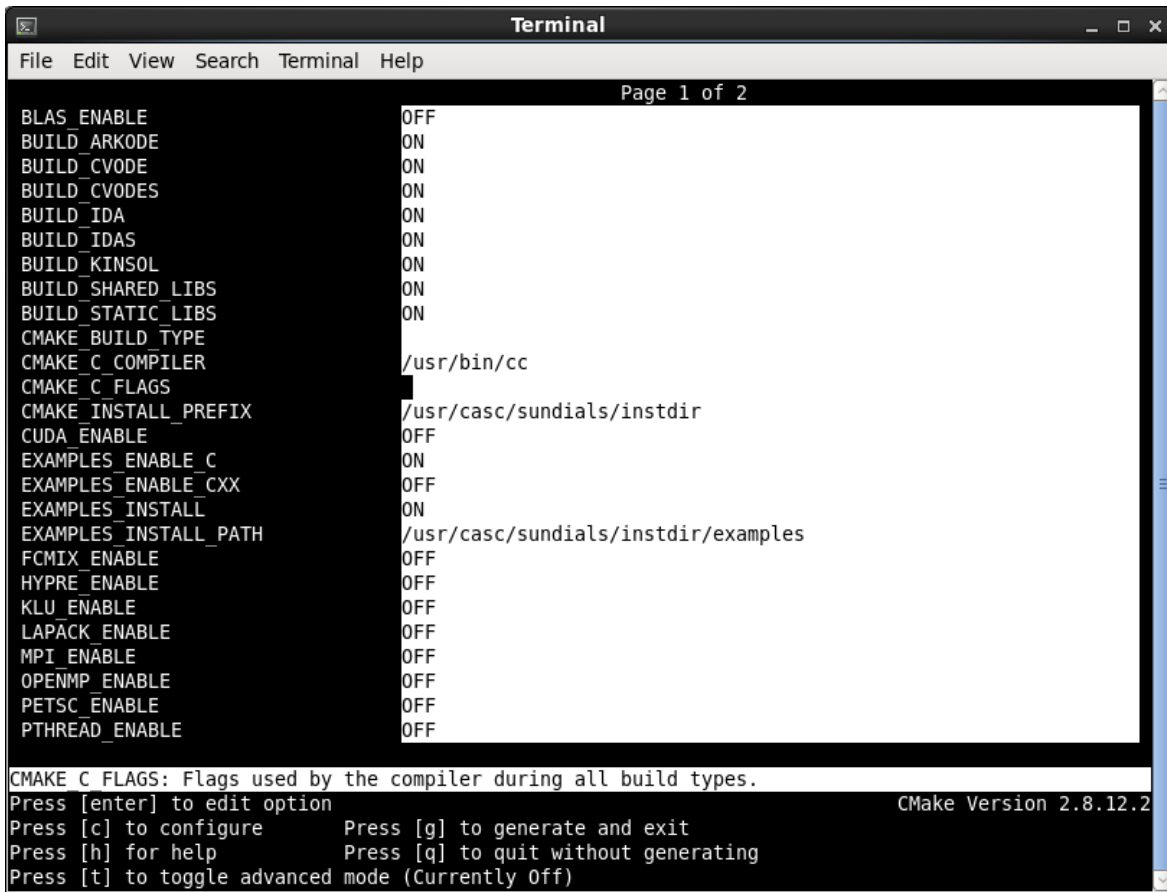


Figure 1.2: Changing the *instdir* for SUNDIALS and corresponding examples

Building from the command line

Using CMake from the command line is simply a matter of specifying CMake variable settings with the `cmake` command. The following will build the default configuration:

```
% cmake -DCMAKE_INSTALL_PREFIX=/home/myname/sundials/instdir \
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instdir/examples \
> ../solverdir
% make
% make install
```

1.1.2 Configuration options (Unix/Linux)

A complete list of all available options for a CMake-based SUNDIALS configuration is provide below. Note that the default values shown are for a typical configuration on a Linux system and are provided as illustration only.

BLAS_ENABLE - Enable BLAS support

Default: OFF

Note: Setting this option to ON will trigger additional CMake options. See additional information on building with BLAS enabled in [1.1.4](#).

BLAS_LIBRARIES - BLAS library

Default: /usr/lib/libblas.so

Note: CMake will search for libraries in your `LD_LIBRARY_PATH` prior to searching default system paths.

`BUILD_ARKODE` - Build the ARKODE library
Default: ON

`BUILD_CVODE` - Build the CVODE library
Default: ON

`BUILD_CVODES` - Build the CVODES library
Default: ON

`BUILD_IDA` - Build the IDA library
Default: ON

`BUILD_IDAS` - Build the IDAS library
Default: ON

`BUILD_KINSOL` - Build the KINSOL library
Default: ON

`BUILD_SHARED_LIBS` - Build shared libraries
Default: ON

`BUILD_STATIC_LIBS` - Build static libraries
Default: ON

`CMAKE_BUILD_TYPE` - Choose the type of build, options are: `None` (`CMAKE_C_FLAGS` used), `Debug`, `Release`, `RelWithDebInfo`, and `MinSizeRel`
Default:
Note: Specifying a build type will trigger the corresponding build type specific compiler flag options below which will be appended to the flags set by `CMAKE_<language>_FLAGS`.

`CMAKE_C_COMPILER` - C compiler
Default: `/usr/bin/cc`

`CMAKE_C_FLAGS` - Flags for C compiler
Default:

`CMAKE_C_FLAGS_DEBUG` - Flags used by the C compiler during debug builds
Default: `-g`

`CMAKE_C_FLAGS_MINSIZEREL` - Flags used by the C compiler during release minsize builds
Default: `-Os -DNDEBUG`

`CMAKE_C_FLAGS_RELEASE` - Flags used by the C compiler during release builds
Default: `-O3 -DNDEBUG`

`CMAKE_CXX_COMPILER` - C++ compiler
Default: `/usr/bin/c++`
Note: A C++ compiler (and all related options) are only triggered if C++ examples are enabled (`EXAMPLES_ENABLE_CXX` is ON). All SUNDIALS solvers can be used from C++ applications by default without setting any additional configuration options.

`CMAKE_CXX_FLAGS` - Flags for C++ compiler
Default:

`CMAKE_CXX_FLAGS_DEBUG` - Flags used by the C++ compiler during debug builds
Default: `-g`

CMAKE_CXX_FLAGS_MINSIZEREL - Flags used by the C++ compiler during release minsize builds
 Default: -Os -DNDEBUG

CMAKE_CXX_FLAGS_RELEASE - Flags used by the C++ compiler during release builds
 Default: -O3 -DNDEBUG

CMAKE_Fortran_COMPILER - Fortran compiler
 Default: /usr/bin/gfortran
 Note: Fortran support (and all related options) are triggered only if either Fortran-C support is enabled (**FCMIX_ENABLE** is ON) or BLAS/LAPACK support is enabled (**BLAS_ENABLE** or **LAPACK_ENABLE** is ON).

CMAKE_Fortran_FLAGS - Flags for Fortran compiler
 Default:

CMAKE_Fortran_FLAGS_DEBUG - Flags used by the Fortran compiler during debug builds
 Default: -g

CMAKE_Fortran_FLAGS_MINSIZEREL - Flags used by the Fortran compiler during release minsize builds
 Default: -Os

CMAKE_Fortran_FLAGS_RELEASE - Flags used by the Fortran compiler during release builds
 Default: -O3

CMAKE_INSTALL_PREFIX - Install path prefix, prepended onto install directories
 Default: /usr/local
 Note: The user must have write access to the location specified through this option. Exported SUNDIALS header files and libraries will be installed under subdirectories **include** and **lib** of **CMAKE_INSTALL_PREFIX**, respectively.

CUDA_ENABLE - Build the SUNDIALS CUDA vector module.
 Default: OFF

EXAMPLES_ENABLE_C - Build the SUNDIALS C examples
 Default: ON

EXAMPLES_ENABLE_CUDA - Build the SUNDIALS CUDA examples
 Default: OFF
 Note: You need to enable CUDA support to build these examples.

EXAMPLES_ENABLE_CXX - Build the SUNDIALS C++ examples
 Default: OFF

EXAMPLES_ENABLE_RAJA - Build the SUNDIALS RAJA examples
 Default: OFF
 Note: You need to enable CUDA and RAJA support to build these examples.

EXAMPLES_ENABLE_F77 - Build the SUNDIALS Fortran77 examples
 Default: ON (if **FCMIX_ENABLE** is ON)

EXAMPLES_ENABLE_F90 - Build the SUNDIALS Fortran90 examples
 Default: OFF

EXAMPLES_INSTALL - Install example files
 Default: ON
 Note: This option is triggered when any of the SUNDIALS example programs are enabled (**EXAMPLES_ENABLE_<language>** is ON). If the user requires installation of example programs then the sources and sample output files for all SUNDIALS modules that are currently enabled will be exported to the directory specified by **EXAMPLES_INSTALL_PATH**. A CMake configuration

script will also be automatically generated and exported to the same directory. Additionally, if the configuration is done under a Unix-like system, makefiles for the compilation of the example programs (using the installed SUNDIALS libraries) will be automatically generated and exported to the directory specified by `EXAMPLES_INSTALL_PATH`.

`EXAMPLES_INSTALL_PATH` - Output directory for installing example files

Default: `/usr/local/examples`

Note: The actual default value for this option will be an `examples` subdirectory created under `CMAKE_INSTALL_PREFIX`.

`FCMIX_ENABLE` - Enable Fortran-C support

Default: `OFF`

`HYPRE_ENABLE` - Enable *hypre* support

Default: `OFF`

Note: See additional information on building with *hypre* enabled in [1.1.4](#).

`HYPRE_INCLUDE_DIR` - Path to *hypre* header files

`HYPRE_LIBRARY_DIR` - Path to *hypre* installed library files

`KLU_ENABLE` - Enable KLU support

Default: `OFF`

Note: See additional information on building with KLU enabled in [1.1.4](#).

`KLU_INCLUDE_DIR` - Path to SuiteSparse header files

`KLU_LIBRARY_DIR` - Path to SuiteSparse installed library files

`LAPACK_ENABLE` - Enable LAPACK support

Default: `OFF`

Note: Setting this option to `ON` will trigger additional CMake options. See additional information on building with LAPACK enabled in [1.1.4](#).

`LAPACK_LIBRARIES` - LAPACK (and BLAS) libraries

Default: `/usr/lib/liblapack.so;/usr/lib/libblas.so`

Note: CMake will search for libraries in your `LD_LIBRARY_PATH` prior to searching default system paths.

`MPI_ENABLE` - Enable MPI support (build the parallel nvector).

Default: `OFF`

Note: Setting this option to `ON` will trigger several additional options related to MPI.

`MPI_C_COMPILER` - `mpicc` program

Default:

`MPI_CXX_COMPILER` - `mpicxx` program

Default:

Note: This option is triggered only if MPI is enabled (`MPI_ENABLE` is `ON`) and C++ examples are enabled (`EXAMPLES_ENABLE_CXX` is `ON`). All SUNDIALS solvers can be used from C++ MPI applications by default without setting any additional configuration options other than `MPI_ENABLE`.

`MPI_Fortran_COMPILER` - `mpif77` or `mpif90` program

Default:

Note: This option is triggered only if MPI is enabled (`MPI_ENABLE` is `ON`), Fortran-C support is enabled (`FCMIX_ENABLE` is `ON`), and Fortran77 or Fortran90 examples are enabled (`EXAMPLES_ENABLE_F77` or `EXAMPLES_ENABLE_F90` are `ON`).

MPIEXEC_EXECUTABLE - Specify the executable for running MPI programs
 Default: `mpirun`
 Note: This option is triggered only if MPI is enabled (`MPI_ENABLE` is ON).

OPENMP_ENABLE - Enable OpenMP support (build the OpenMP nvector).
 Default: OFF

PETSC_ENABLE - Enable PETSc support
 Default: OFF
 Note: See additional information on building with PETSc enabled in [1.1.4](#).

PETSC_INCLUDE_DIR - Path to PETSc header files

PETSC_LIBRARY_DIR - Path to PETSc installed library files

PTHREAD_ENABLE - Enable Pthreads support (build the Pthreads nvector).
 Default: OFF

RAJA_ENABLE - Enable RAJA support (build the RAJA nvector).
 Default: OFF
 Note: You need to enable CUDA in order to build the RAJA vector module.

SUNDIALS_F77_FUNC_CASE - **advanced option** - Specify the case to use in the Fortran name-mangling scheme, options are: `lower` or `upper`
 Default:
 Note: The build system will attempt to infer the Fortran name-mangling scheme using the Fortran compiler. This option should only be used if a Fortran compiler is not available or to override the inferred or default (`lower`) scheme if one can not be determined. If used, `SUNDIALS_F77_FUNC_UNDERSCORES` must also be set.

SUNDIALS_F77_FUNC_UNDERSCORES - **advanced option** - Specify the number of underscores to append in the Fortran name-mangling scheme, options are: `none`, `one`, or `two`
 Default:
 Note: The build system will attempt to infer the Fortran name-mangling scheme using the Fortran compiler. This option should only be used if a Fortran compiler is not available or to override the inferred or default (`one`) scheme if one can not be determined. If used, `SUNDIALS_F77_FUNC_CASE` must also be set.

SUNDIALS_INDEX_TYPE - **advanced**
 Integer type used for SUNDIALS indices. The size must match the size provided for the `SUNDIALS_INDEX_SIZE` option.
 Default:
 Note: In past SUNDIALS versions, a user could set this option to `INT64_T` to use 64-bit integers, or `INT32_T` to use 32-bit integers. Starting in SUNDIALS 3.2.0, these special values are deprecated. For SUNDIALS 3.2.0 and up, a user will only need to use the `SUNDIALS_INDEX_SIZE` option in most cases.

SUNDIALS_INDEX_SIZE - Integer size (in bits) used for indices in SUNDIALS, options are: `32` or `64`
 Default: `64`
 Note: The build system tries to find an integer type of appropriate size. Candidate 64-bit integer types are (in order of preference): `int64_t`, `__int64`, `long long`, and `long`. Candidate 32-bit integers are (in order of preference): `int32_t`, `int`, and `long`. The advanced option, `SUNDIALS_INDEX_TYPE` can be used to provide a type not listed here.

SUNDIALS_PRECISION - Precision used in SUNDIALS, options are: `double`, `single`, or `extended`
 Default: `double`

SUPERLUMT_ENABLE - Enable SuperLU_MT support

Default: OFF

Note: See additional information on building with SuperLU_MT enabled in [1.1.4](#).

SUPERLUMT_INCLUDE_DIR - Path to SuperLU_MT header files (typically SRC directory)

SUPERLUMT_LIBRARY_DIR - Path to SuperLU_MT installed library files

SUPERLUMT_THREAD_TYPE - Must be set to Pthread or OpenMP

Default: Pthread

USE_GENERIC_MATH - Use generic (stdc) math libraries

Default: ON

xSDK Configuration Options

SUNDIALS supports CMake configuration options defined by the Extreme-scale Scientific Software Development Kit (xSDK) community policies (see <https://xsdk.info> for more information). xSDK CMake options are unused by default but may be activated by setting USE_XSDK_DEFAULTS to ON.

When xSDK options are active, they will overwrite the corresponding SUNDIALS option and may have different default values (see details below). As such the equivalent SUNDIALS options should not be used when configuring with xSDK options. In the GUI front end to CMake (ccmake), setting USE_XSDK_DEFAULTS to ON will hide the corresponding SUNDIALS options as advanced CMake variables. During configuration, messages are output detailing which xSDK flags are active and the equivalent SUNDIALS options that are replaced. Below is a complete list xSDK options and the corresponding SUNDIALS options if applicable.



TPL_BLAS_LIBRARIES - BLAS library

Default: /usr/lib/libblas.so

SUNDIALS equivalent: BLAS_LIBRARIES

Note: CMake will search for libraries in your LD_LIBRARY_PATH prior to searching default system paths.

TPL_ENABLE_BLAS - Enable BLAS support

Default: OFF

SUNDIALS equivalent: BLAS_ENABLE

TPL_ENABLE_HYPRE - Enable *hypre* support

Default: OFF

SUNDIALS equivalent: HYPRE_ENABLE

TPL_ENABLE_KLU - Enable KLU support

Default: OFF

SUNDIALS equivalent: KLU_ENABLE

TPL_ENABLE_PETSC - Enable PETSc support

Default: OFF

SUNDIALS equivalent: PETSC_ENABLE

TPL_ENABLE_LAPACK - Enable LAPACK support

Default: OFF

SUNDIALS equivalent: LAPACK_ENABLE

TPL_ENABLE_SUPERLUMT - Enable SuperLU_MT support

Default: OFF

SUNDIALS equivalent: SUPERLUMT_ENABLE

TPL_HYPRE_INCLUDE_DIRS - Path to *hypre* header files

SUNDIALS equivalent: HYPRE_INCLUDE_DIR

TPL_HYPRE_LIBRARIES - *hypre* library
SUNDIALS equivalent: N/A

TPL_KLU_INCLUDE_DIRS - Path to KLU header files
SUNDIALS equivalent: **KLU_INCLUDE_DIR**

TPL_KLU_LIBRARIES - KLU library
SUNDIALS equivalent: N/A

TPL_LAPACK_LIBRARIES - LAPACK (and BLAS) libraries
Default: `/usr/lib/liblapack.so;/usr/lib/libblas.so`
SUNDIALS equivalent: **LAPACK_LIBRARIES**
Note: CMake will search for libraries in your **LD_LIBRARY_PATH** prior to searching default system paths.

TPL_PETSC_INCLUDE_DIRS - Path to PETSc header files
SUNDIALS equivalent: **PETSC_INCLUDE_DIR**

TPL_PETSC_LIBRARIES - PETSc library
SUNDIALS equivalent: N/A

TPL_SUPERLUMT_INCLUDE_DIRS - Path to SuperLU_MT header files
SUNDIALS equivalent: **SUPERLUMT_INCLUDE_DIR**

TPL_SUPERLUMT_LIBRARIES - SuperLU_MT library
SUNDIALS equivalent: N/A

TPL_SUPERLUMT_THREAD_TYPE - SuperLU_MT library thread type
SUNDIALS equivalent: **SUPERLUMT_THREAD_TYPE**

USE_XSDK_DEFAULTS - Enable xSDK default configuration settings
Default: OFF
SUNDIALS equivalent: N/A
Note: Enabling xSDK defaults also sets **CMAKE_BUILD_TYPE** to Debug

XSDK_ENABLE_FORTRAN - Enable SUNDIALS Fortran interface
Default: OFF
SUNDIALS equivalent: **FCMIX_ENABLE**

XSDK_INDEX_SIZE - Integer size (bits) used for indices in SUNDIALS, options are: 32 or 64
Default: 32
SUNDIALS equivalent: **SUNDIALS_INDEX_SIZE**

XSDK_PRECISION - Precision used in SUNDIALS, options are: **double**, **single**, or **quad**
Default: **double**
SUNDIALS equivalent: **SUNDIALS_PRECISION**

1.1.3 Configuration examples

The following examples will help demonstrate usage of the CMake configure options. To configure SUNDIALS using the default C and Fortran compilers, and default **mpicc** and **mpif77** parallel compilers, enable compilation of examples, and install libraries, headers, and example sources under subdirectories of `/home/myname/sundials/`, use:

```
% cmake \  
> -DCMAKE_INSTALL_PREFIX=/home/myname/sundials/instdir \  
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instdir/examples \  
> -DMPI_ENABLE=ON \  
> -DFCMIX_ENABLE=ON \  

```

```

> /home/myname/sundials/solverdir
%
% make install
%

```

To disable installation of the examples, use:

```

% cmake \
> -DCMAKE_INSTALL_PREFIX=/home/myname/sundials/instdir \
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instdir/examples \
> -DMPI_ENABLE=ON \
> -DFCMIX_ENABLE=ON \
> -DEXAMPLES_INSTALL=OFF \
> /home/myname/sundials/solverdir
%
% make install
%

```

1.1.4 Working with external Libraries

The SUNDIALS suite contains many options to enable implementation flexibility when developing solutions. The following are some notes addressing specific configurations when using the supported third party libraries. When building SUNDIALS as a shared library external libraries any used with SUNDIALS must also be build as a shared library or as a static library compiled with the `-fPIC` flag.



Building with BLAS

SUNDIALS does not utilize BLAS directly but it may be needed by other external libraries that SUNDIALS can be built with (e.g. LAPACK, PETSc, SuperLU_MT, etc.). To enable BLAS, set the `BLAS_ENABLE` option to `ON`. If the directory containing the BLAS library is in the `LD_LIBRARY_PATH` environment variable, CMake will set the `BLAS_LIBRARIES` variable accordingly, otherwise CMake will attempt to find the BLAS library in standard system locations. To explicitly tell CMake what libraries to use, the `BLAS_LIBRARIES` variable can be set to the desired library. Example:

```

% cmake \
> -DCMAKE_INSTALL_PREFIX=/home/myname/sundials/instdir \
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instdir/examples \
> -DBLAS_ENABLE=ON \
> -DBLAS_LIBRARIES=/myblaspath/lib/libblas.so \
> -DSUPERLUMT_ENABLE=ON \
> -DSUPERLUMT_INCLUDE_DIR=/mysuperlumtpath/SRC
> -DSUPERLUMT_LIBRARY_DIR=/mysuperlumtpath/lib
> /home/myname/sundials/solverdir
%
% make install
%

```

When allowing CMake to automatically locate the LAPACK library, CMake *may* also locate the corresponding BLAS library.



If a working Fortran compiler is not available to infer the Fortran name-mangling scheme, the options `SUNDIALS_F77_FUNC.CASE` and `SUNDIALS_F77_FUNC.UNDERSCORES` *must* be set in order to bypass the check for a Fortran compiler and define the name-mangling scheme. The defaults for these options in earlier versions of SUNDIALS were `lower` and `one` respectively.

Building with LAPACK

To enable LAPACK, set the `LAPACK_ENABLE` option to `ON`. If the directory containing the LAPACK library is in the `LD_LIBRARY_PATH` environment variable, CMake will set the `LAPACK_LIBRARIES` variable accordingly, otherwise CMake will attempt to find the LAPACK library in standard system locations. To explicitly tell CMake what library to use, the `LAPACK_LIBRARIES` variable can be set to the desired libraries. When setting the LAPACK location explicitly the location of the corresponding BLAS library will also need to be set. Example:



```
% cmake \  
> -DCMAKE_INSTALL_PREFIX=/home/myname/sundials/instdir \  
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instdir/examples \  
> -DBLAS_ENABLE=ON \  
> -DBLAS_LIBRARIES=/mylapackpath/lib/libblas.so \  
> -DLAPACK_ENABLE=ON \  
> -DLAPACK_LIBRARIES=/mylapackpath/lib/liblapack.so \  
> /home/myname/sundials/solverdir  
%  
% make install  
%
```



When allowing CMake to automatically locate the LAPACK library, CMake *may* also locate the corresponding BLAS library.

If a working Fortran compiler is not available to infer the Fortran name-mangling scheme, the options `SUNDIALS_F77_FUNC_CASE` and `SUNDIALS_F77_FUNC_UNDERSCORES` *must* be set in order to bypass the check for a Fortran compiler and define the name-mangling scheme. The defaults for these options in earlier versions of SUNDIALS were `lower` and `one` respectively.

Building with KLU

The KLU libraries are part of SuiteSparse, a suite of sparse matrix software, available from the Texas A&M University website: <http://faculty.cse.tamu.edu/davis/suitesparse.html>. SUNDIALS has been tested with SuiteSparse version 4.5.3. To enable KLU, set `KLU_ENABLE` to `ON`, set `KLU_INCLUDE_DIR` to the `include` path of the KLU installation and set `KLU_LIBRARY_DIR` to the `lib` path of the KLU installation. The CMake configure will result in populating the following variables: `AMD_LIBRARY`, `AMD_LIBRARY_DIR`, `BTF_LIBRARY`, `BTF_LIBRARY_DIR`, `COLAMD_LIBRARY`, `COLAMD_LIBRARY_DIR`, and `KLU_LIBRARY`.

Building with SuperLU_MT

The SuperLU_MT libraries are available for download from the Lawrence Berkeley National Laboratory website: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/#superlu_mt. SUNDIALS has been tested with SuperLU_MT version 3.1. To enable SuperLU_MT, set `SUPERLUMT_ENABLE` to `ON`, set `SUPERLUMT_INCLUDE_DIR` to the `SRC` path of the SuperLU_MT installation, and set the variable `SUPERLUMT_LIBRARY_DIR` to the `lib` path of the SuperLU_MT installation. At the same time, the variable `SUPERLUMT_THREAD_TYPE` must be set to either `Pthread` or `OpenMP`.

Do not mix thread types when building SUNDIALS solvers. If threading is enabled for SUNDIALS by having either `OPENMP_ENABLE` or `PTHREAD_ENABLE` set to `ON` then SuperLU_MT should be set to use the same threading type.



Building with PETSc

The PETSc libraries are available for download from the Argonne National Laboratory website: <http://www.mcs.anl.gov/petsc>. SUNDIALS has been tested with PETSc version 3.7.2. To enable PETSc, set `PETSC_ENABLE` to `ON`, set `PETSC_INCLUDE_DIR` to the `include` path of the PETSc installation, and set the variable `PETSC_LIBRARY_DIR` to the `lib` path of the PETSc installation.

Building with *hypre*

The *hypre* libraries are available for download from the Lawrence Livermore National Laboratory website: <http://computation.llnl.gov/projects/hypre>. SUNDIALS has been tested with *hypre* version 2.11.1. To enable *hypre*, set `HYPRE_ENABLE` to `ON`, set `HYPRE_INCLUDE_DIR` to the `include` path of the *hypre* installation, and set the variable `HYPRE_LIBRARY_DIR` to the `lib` path of the *hypre* installation.

Building with CUDA

SUNDIALS CUDA modules and examples have been tested with version 8.0 of the CUDA toolkit. To build them, you need to install the Toolkit and compatible NVIDIA drivers. Both are available for download from the NVIDIA website: <https://developer.nvidia.com/cuda-downloads>. To enable CUDA, set `CUDA_ENABLE` to `ON`. If CUDA is installed in a nonstandard location, you may be prompted to set the variable `CUDA_TOOLKIT_ROOT_DIR` with your CUDA Toolkit installation path. To enable CUDA examples, set `EXAMPLES_ENABLE_CUDA` to `ON`.

Building with RAJA

RAJA is a performance portability layer developed by Lawrence Livermore National Laboratory and can be obtained from <https://github.com/LLNL/RAJA>. SUNDIALS RAJA modules and examples have been tested with RAJA version 0.3. Building SUNDIALS RAJA modules requires a CUDA-enabled RAJA installation. To enable RAJA, set `CUDA_ENABLE` and `RAJA_ENABLE` to `ON`. If RAJA is installed in a nonstandard location you will be prompted to set the variable `RAJA_DIR` with the path to the RAJA CMake configuration file. To enable building the RAJA examples set `EXAMPLES_ENABLE_RAJA` to `ON`.

1.1.5 Testing the build and installation

If SUNDIALS was configured with `EXAMPLES_ENABLE_<language>` options to `ON`, then a set of regression tests can be run after building with the `make` command by running:

```
% make test
```

Additionally, if `EXAMPLES_INSTALL` was also set to `ON`, then a set of smoke tests can be run after installing with the `make install` command by running:

```
% make test_install
```

1.2 Building and Running Examples

Each of the SUNDIALS solvers is distributed with a set of examples demonstrating basic usage. To build and install the examples, set at least of the `EXAMPLES_ENABLE_<language>` options to `ON`, and set `EXAMPLES_INSTALL` to `ON`. Specify the installation path for the examples with the variable `EXAMPLES_INSTALL_PATH`. CMake will generate `CMakeLists.txt` configuration files (and `Makefile` files if on Linux/Unix) that reference the *installed* SUNDIALS headers and libraries.

Either the `CMakeLists.txt` file or the traditional `Makefile` may be used to build the examples as well as serve as a template for creating user developed solutions. To use the supplied `Makefile` simply run `make` to compile and generate the executables. To use CMake from within the installed example directory, run `cmake` (or `ccmake` to use the GUI) followed by `make` to compile the example code. Note that if CMake is used, it will overwrite the traditional `Makefile` with a new CMake-generated `Makefile`. The resulting output from running the examples can be compared with example output bundled in the SUNDIALS distribution.

NOTE: There will potentially be differences in the output due to machine architecture, compiler versions, use of third party libraries etc.



1.3 Configuring, building, and installing on Windows

CMake can also be used to build SUNDIALS on Windows. To build SUNDIALS for use with Visual Studio the following steps should be performed:

1. Unzip the downloaded tar file(s) into a directory. This will be the *solverdir*
2. Create a separate *builddir*
3. Open a Visual Studio Command Prompt and cd to *builddir*
4. Run `cmake-gui ../solverdir`
 - (a) Hit Configure
 - (b) Check/Uncheck solvers to be built
 - (c) Change CMAKE_INSTALL_PREFIX to *instdir*
 - (d) Set other options as desired
 - (e) Hit Generate
5. Back in the VS Command Window:
 - (a) Run `msbuild ALL_BUILD.vcxproj`
 - (b) Run `msbuild INSTALL.vcxproj`

The resulting libraries will be in the *instdir*. The SUNDIALS project can also now be opened in Visual Studio. Double click on the ALL_BUILD.vcxproj file to open the project. Build the whole *solution* to create the SUNDIALS libraries. To use the SUNDIALS libraries in your own projects, you must set the include directories for your project, add the SUNDIALS libraries to your project solution, and set the SUNDIALS libraries as dependencies for your project.

1.4 Installed libraries and exported header files

Using the CMake SUNDIALS build system, the command

```
% make install
```

will install the libraries under *libdir* and the public header files under *includedir*. The values for these directories are *instdir/lib* and *instdir/include*, respectively. The location can be changed by setting the CMake variable CMAKE_INSTALL_PREFIX. Although all installed libraries reside under *libdir/lib*, the public header files are further organized into subdirectories under *includedir/include*.

The installed libraries and exported header files are listed for reference in Table 1.1. The file extension *.lib* is typically *.so* for shared libraries and *.a* for static libraries. Note that, in the Tables, names are relative to *libdir* for libraries and to *includedir* for header files.

A typical user program need not explicitly include any of the shared SUNDIALS header files from under the *includedir/include/sundials* directory since they are explicitly included by the appropriate solver header files (*e.g.*, *cvode_dense.h* includes *sundials_dense.h*). However, it is both legal and safe to do so, and would be useful, for example, if the functions declared in *sundials_dense.h* are to be used in building a preconditioner.

Table 1.1: SUNDIALS libraries and header files

SHARED	Libraries	n/a
<i>continued on next page</i>		

<i>continued from last page</i>			
	Header files	sundials/sundials_config.h sundials/sundials_types.h sundials/sundials_nvector.h sundials/sundials_iterative.h sundials/sundials_dense.h sundials/sundials_matrix.h sundials/sundials_linear solver.h	sundials/sundials_fconfig.h sundials/sundials_math.h sundials/sundials_fvector.h sundials/sundials_direct.h sundials/sundials_band.h sundials/sundials_version.h sundials/sundials_mpi_types.h
NVECTOR_SERIAL	Libraries	libsundials_nvecserial. <i>lib</i>	libsundials_fvecserial.a
	Header files	nvector/nvector_serial.h	
NVECTOR_PARALLEL	Libraries	libsundials_nvecparallel. <i>lib</i>	libsundials_fvecparallel.a
	Header files	nvector/nvector_parallel.h	
NVECTOR_OPENMP	Libraries	libsundials_nvecopenmp. <i>lib</i>	libsundials_fvecopenmp.a
	Header files	nvector/nvector_openmp.h	
NVECTOR_PTHREADS	Libraries	libsundials_nvecpthreads. <i>lib</i>	libsundials_fvecpthreads.a
	Header files	nvector/nvector_pthreads.h	
NVECTOR_PARHYP	Libraries	libsundials_nvecparhyp. <i>lib</i>	
	Header files	nvector/nvector_parhyp.h	
NVECTOR_PETSC	Libraries	libsundials_nvecpetsc. <i>lib</i>	
	Header files	nvector/nvector_petsc.h	
NVECTOR_CUDA	Libraries	libsundials_nveccuda. <i>lib</i>	
	Libraries	libsundials_nvecmpicuda. <i>lib</i>	
	Header files	nvector/nvector_cuda.h	
		nvector/nvector_mpicuda.h nvector/cuda/ThreadPartitioning.hpp nvector/cuda/Vector.hpp nvector/cuda/VectorKernels.cuh	
NVECTOR_RAJA	Libraries	libsundials_nveccudaraja. <i>lib</i>	
	Libraries	libsundials_nveccudampiraja. <i>lib</i>	
	Header files	nvector/nvector_raja.h nvector/nvector_mpiraja.h nvector/raja/Vector.hpp	
SUNMATRIX_BAND	Libraries	libsundials_sunmatrixband. <i>lib</i> libsundials_fsunmatrixband.a	
	Header files	sunmatrix/sunmatrix_band.h	
SUNMATRIX_DENSE	Libraries	libsundials_sunmatrixdense. <i>lib</i> libsundials_fsunmatrixdense.a	
	Header files	sunmatrix/sunmatrix_dense.h	
SUNMATRIX_SPARSE	Libraries	libsundials_sunmatrixsparse. <i>lib</i> libsundials_fsunmatrixsparse.a	
	Header files	sunmatrix/sunmatrix_sparse.h	
SUNLINSOL_BAND	Libraries	libsundials_sunlinsolband. <i>lib</i>	
<i>continued on next page</i>			

<i>continued from last page</i>			
		libsundials_fsunlinsolband.a	
	Header files	sunlinsol/sunlinsol_band.h	
SUNLINSOL_DENSE	Libraries	libsundials_sunlinsoldense. <i>lib</i> libsundials_fsunlinsoldense.a	
	Header files	sunlinsol/sunlinsol_dense.h	
SUNLINSOL_KLU	Libraries	libsundials_sunlinsolklu. <i>lib</i> libsundials_fsunlinsolklu.a	
	Header files	sunlinsol/sunlinsol_klu.h	
SUNLINSOL_LAPACKBAND	Libraries	libsundials_sunlinsollapackband. <i>lib</i> libsundials_fsunlinsollapackband.a	
	Header files	sunlinsol/sunlinsol_lapackband.h	
SUNLINSOL_LAPACKDENSE	Libraries	libsundials_sunlinsollapackdense. <i>lib</i> libsundials_fsunlinsollapackdense.a	
	Header files	sunlinsol/sunlinsol_lapackdense.h	
SUNLINSOL_PCG	Libraries	libsundials_sunlinsolpcg. <i>lib</i> libsundials_fsunlinsolpcg.a	
	Header files	sunlinsol/sunlinsol_pcg.h	
SUNLINSOL_SPBCGS	Libraries	libsundials_sunlinsolspbcgs. <i>lib</i> libsundials_fsunlinsolspbcgs.a	
	Header files	sunlinsol/sunlinsol_spbcgs.h	
SUNLINSOL_SPFQMR	Libraries	libsundials_sunlinsolspfqr. <i>lib</i> libsundials_fsunlinsolspfqr.a	
	Header files	sunlinsol/sunlinsol_spfqmr.h	
SUNLINSOL_SPGMR	Libraries	libsundials_sunlinsolspgmr. <i>lib</i> libsundials_fsunlinsolspgmr.a	
	Header files	sunlinsol/sunlinsol_spgmr.h	
SUNLINSOL_SPTFQMR	Libraries	libsundials_sunlinsolsptfqmr. <i>lib</i> libsundials_fsunlinsolsptfqmr.a	
	Header files	sunlinsol/sunlinsol_sptfqmr.h	
SUNLINSOL_SUPERLUMT	Libraries	libsundials_sunlinsolsuperlumt. <i>lib</i> libsundials_fsunlinsolsuperlumt.a	
	Header files	sunlinsol/sunlinsol_superlumt.h	
CVODE	Libraries	libsundials_cvode. <i>lib</i>	libsundials_fcvcde.a
	Header files	cvcde/cvcde.h	cvcde/cvcde_impl.h
		cvcde/cvcde_direct.h	cvcde/cvcde_spils.h
		cvcde/cvcde_bandpre.h	cvcde/cvcde_bbdpre.h
CVODES	Libraries	libsundials_cvodes. <i>lib</i>	
	Header files	cvodes/cvodes.h	cvodes/cvodes_impl.h
		cvodes/cvodes_direct.h	cvodes/cvodes_spils.h
		cvodes/cvodes_bandpre.h	cvodes/cvodes_bbdpre.h
ARKODE	Libraries	libsundials_arkode. <i>lib</i>	libsundials_farkode.a
<i>continued on next page</i>			

<i>continued from last page</i>			
	Header files	arkode/arkode.h arkode/arkode_direct.h arkode/arkode_bandpre.h	arkode/arkode_impl.h arkode/arkode_spils.h arkode/arkode_bbdpre.h
IDA	Libraries	libsundials_ida. <i>lib</i>	libsundials_fida.a
	Header files	ida/ida.h ida/ida_direct.h ida/ida_bbdpre.h	ida/ida_impl.h ida/ida_spils.h
IDAS	Libraries	libsundials_idas. <i>lib</i>	
	Header files	idas/idas.h idas/idas_direct.h idas/idas_bbdpre.h	idas/idas_impl.h idas/idas_spils.h
KINSOL	Libraries	libsundials_kinsol. <i>lib</i>	libsundials_fkinsol.a
	Header files	kinsol/kinsol.h kinsol/kinsol_direct.h kinsol/kinsol_bbdpre.h	kinsol/kinsol_impl.h kinsol/kinsol_spils.h

