

Installation Guide for SUNDIALS v5.6.0

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

December 11, 2020



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.

CONTRIBUTORS

The SUNDIALS library has been developed over many years by a number of contributors. The current SUNDIALS team consists of Cody J. Balos, David J. Gardner, Alan C. Hindmarsh, Daniel R. Reynolds, and Carol S. Woodward. We thank Radu Serban for significant and critical past contributions.

Other contributors to SUNDIALS include: James Almgren-Bell, Lawrence E. Banks, Peter N. Brown, George Byrne, Rujeko Chinomona, Scott D. Cohen, Aaron Collier, Keith E. Grant, Steven L. Lee, Shelby L. Lockhart, John Loffeld, Daniel McGreer, Slaven Peles, Cosmin Petra, H. Hunter Schwartz, Jean M. Sexton, Dan Shumaker, Steve G. Smith, Allan G. Taylor, Hilari C. Tiedeman, Chris White, Ting Yan, and Ulrike M. Yang.

Contents

1	SUNDIALS Package Installation Procedure	1
1.1	CMake-based installation	2
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/CMAKE_INSTALL_LIBDIR`, with *instdir* and `CMAKE_INSTALL_LIBDIR` 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 *instdir* 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/instdir
% mkdir (...)sundials/builddir
% cd (...)sundials/builddir
```

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.

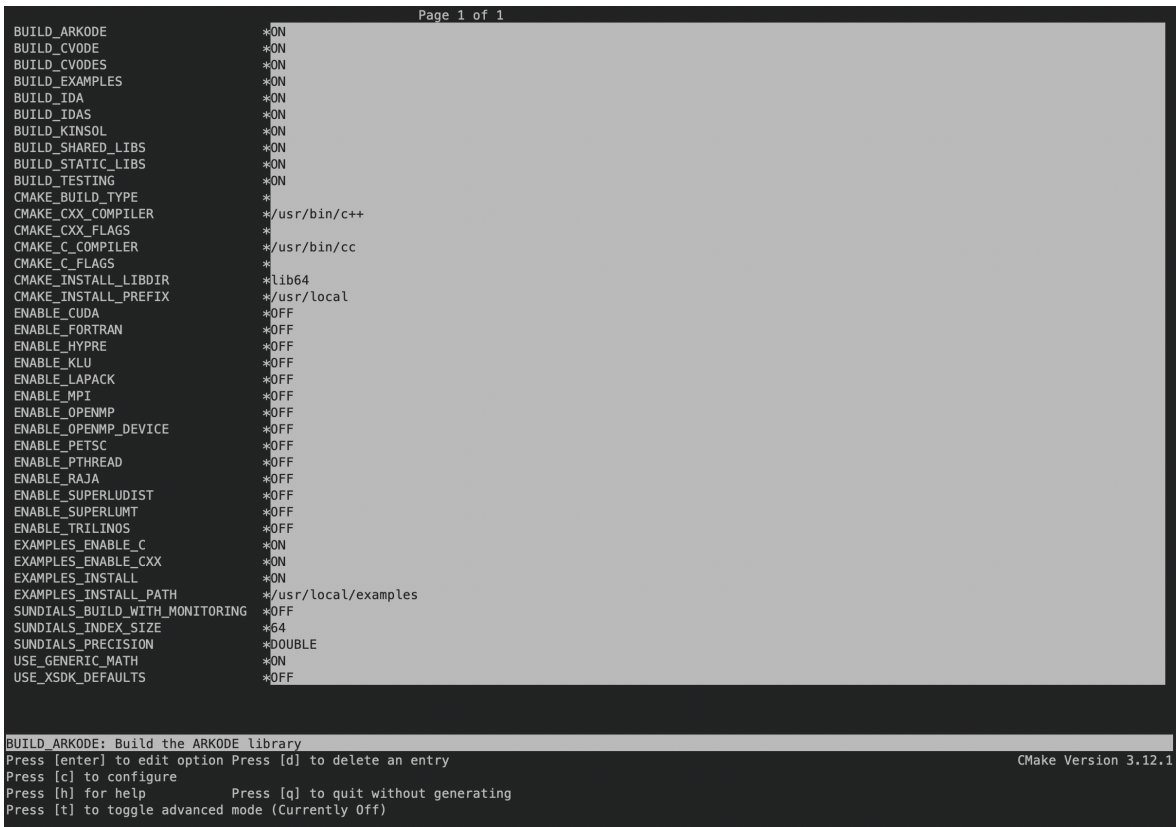


Figure 1.1: Default configuration screen. Note: Initial screen is empty. To get this default configuration, press 'c' repeatedly (accepting default values denoted with asterisk) until the 'g' option is available.

The default *instdir* 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
```

```

Page 1 of 1
BUILD_ARKODE      *ON
BUILD_CVODE       *ON
BUILD_CVODES      *ON
BUILD_EXAMPLES    *ON
BUILD_IDA         *ON
BUILD_IDAS        *ON
BUILD_KINSOL      *ON
BUILD_SHARED_LIBS *ON
BUILD_STATIC_LIBS *ON
BUILD_TESTING     *ON
CMAKE_BUILD_TYPE  *
CMAKE_CXX_COMPILER */usr/bin/c++
CMAKE_CXX_FLAGS    *
CMAKE_C_COMPILER  */usr/bin/cc
CMAKE_C_FLAGS      *
CMAKE_INSTALL_LIBDIR */lib64
CMAKE_INSTALL_PREFIX */usr/casc/sundials/instdir
ENABLE_CUDA        *OFF
ENABLE_FORTRAN     *OFF
ENABLE_HYPRE       *OFF
ENABLE_KLU         *OFF
ENABLE_LAPACK      *OFF
ENABLE_MPI         *OFF
ENABLE_OPENMP      *OFF
ENABLE_OPENMP_DEVICE *OFF
ENABLE_PETSC       *OFF
ENABLE_PTHREAD     *OFF
ENABLE_RAJA        *OFF
ENABLE_SUPERLUDIST *OFF
ENABLE_SUPERLUMT   *OFF
ENABLE_TRILINOS    *OFF
EXAMPLES_ENABLE_C  *ON
EXAMPLES_ENABLE_CXX *ON
EXAMPLES_INSTALL   *ON
EXAMPLES_INSTALL_PATH */usr/casc/sundials/instdir/examples
SUNDIALS_BUILD_WITH_MONITORING *OFF
SUNDIALS_INDEX_SIZE *64
SUNDIALS_PRECISION *DOUBLE
USE_GENERIC_MATH   *ON
USE_XSDK_DEFAULTS  *OFF

EXAMPLES_INSTALL_PATH: Output directory for installing example files
Press [enter] to edit option Press [d] to delete an entry
Press [c] to configure
Press [h] for help
Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
CMake Version 3.12.1

```

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.

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 LAPACK support is enabled (ENABLE_LAPACK 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 **CMAKE_INSTALL_LIBDIR** of **CMAKE_INSTALL_PREFIX**, respectively.

CMAKE_INSTALL_LIBDIR - Library installation directory

Default:

Note: This is the directory within **CMAKE_INSTALL_PREFIX** that the SUNDIALS libraries will be installed under. The default is automatically set based on the operating system using the GNUInstallDirs CMake module.

Fortran_INSTALL_MOODIR - Fortran module installation directory

Default: fortran

ENABLE_CUDA - Build the SUNDIALS CUDA modules.

Default: OFF

CUDA_ARCH - Specifies the CUDA architecture to compile for.

Default: sm_30

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 unless **ENABLE_TRILINOS** is ON.

EXAMPLES_ENABLE_F77 - Build the SUNDIALS Fortran77 examples

Default: ON (if **F77_INTERFACE_ENABLE** is ON)

EXAMPLES_ENABLE_F90 - Build the SUNDIALS Fortran90 examples

Default: ON (if **F77_INTERFACE_ENABLE** is ON)

EXAMPLES_ENABLE_F2003 - Build the SUNDIALS Fortran2003 examples

Default: ON (if **BUILD_FORTRAN_MODULE_INTERFACE** is ON)

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`.

`F77_INTERFACE_ENABLE` - Enable Fortran-C support via the Fortran 77 interfaces

Default: `OFF`

`BUILD_FORTRAN_MODULE_INTERFACE` - Enable Fortran-C support via the Fortran 2003 interfaces

Default: `OFF`

`ENABLE_HYPRE` - 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

`ENABLE_KLU` - 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

`ENABLE_LAPACK` - 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.

`ENABLE_MPI` - Enable MPI support. This will build the parallel `NVECTOR` and the MPI-aware version of the `ManyVector` library.

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 (`ENABLE_MPI` 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 `ENABLE_MPI`.

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

Default:

Note: This option is triggered only if MPI is enabled (`ENABLE_MPI` is `ON`) and Fortran-C support is enabled (`F77_INTERFACE_ENABLE` or `BUILD_FORTRAN_MODULE_INTERFACE` is `ON`).

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

ENABLE_OPENMP - Enable OpenMP support (build the OpenMP NVECTOR).
 Default: OFF

OPENMP_DEVICE_ENABLE - Enable OpenMP device offloading (build the OpenMPDEV nvector) if supported by the provided compiler.
 Default: OFF

OPENMP_DEVICE_WORKS - **advanced option** - Skip the check done to see if the OpenMP provided by the compiler supports OpenMP device offloading.
 Default: OFF

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

PETSC_DIR - Path to PETSc installation
 Default:

PETSC_LIBRARIES - **advanced option** - Semi-colon separated list of PETSc link libraries. Unless provided by the user, this is autopopulated based on the PETSc installation found in **PETSC_DIR**.
 Default:

PETSC_INCLUDES - **advanced option** - Semi-colon separated list of PETSc include directories. Unless provided by the user, this is autopopulated based on the PETSc installation found in **PETSC_DIR**.
 Default:

ENABLE_PTHREAD - Enable Pthreads support (build the Pthreads NVECTOR).
 Default: OFF

ENABLE_RAJA - Enable RAJA support.
 Default: OFF
 Note: You need to enable CUDA or HIP in order to build the RAJA vector module.

SUNDIALS_RAJA_BACKENDS - If building SUNDIALS with RAJA support, this sets the RAJA backend to target. Values supported are CUDA and HIP.
 Default: CUDA

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

SUPERLUDIST_INCLUDE_DIR - Path to SuperLU_DIST header files (typically SRC directory)

SUPERLUDIST_LIBRARY_DIR - Path to SuperLU_DIST installed library files

SUPERLUDIST_LIBRARIES - Semi-colon separated list of libraries needed for SuperLU_DIST

SUPERLUDIST_OpenMP - Enable SUNDIALS support for SuperLU_DIST built with OpenMP
 Default: OFF
 Note: SuperLU_DIST must be built with OpenMP support for this option to function properly. Additionally the environment variable **OMP_NUM_THREADS** must be set to the desired number of threads.

ENABLE_SUPERLUMT - Enable SUPERLUMT support
 Default: OFF
 Note: See additional information on building with SUPERLUMT 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_LIBRARIES` - Semi-colon separated list of libraries needed for SuperLU_MT

`SUPERLUMT_THREAD_TYPE` - Must be set to Pthread or OpenMP
Default: Pthread

`ENABLE_TRILINOS` - Enable Trilinos support (build the Tpetra NVECTOR).
Default: OFF

`Trilinos_DIR` - Path to the Trilinos install directory.
Default:

`TRILINOS_INTERFACE_C_COMPILER` - **advanced option** - Set the C compiler for building the Trilinos interface (i.e., NVECTOR_TRILINOS and the examples that use it).
Default: The C compiler exported from the found Trilinos installation if `USE_XSDK_DEFAULTS=OFF`.
`CMAKE_C_COMPILER` or `MPI_C_COMPILER` if `USE_XSDK_DEFAULTS=ON`.
Note: It is recommended to use the same compiler that was used to build the Trilinos library.

`TRILINOS_INTERFACE_C_COMPILER_FLAGS` - **advanced option** - Set the C compiler flags for Trilinos interface (i.e., NVECTOR_TRILINOS and the examples that use it).
Default: The C compiler flags exported from the found Trilinos installation if `USE_XSDK_DEFAULTS=OFF`.
`CMAKE_C_FLAGS` if `USE_XSDK_DEFAULTS=ON`.
Note: It is recommended to use the same flags that were used to build the Trilinos library.

`TRILINOS_INTERFACE_CXX_COMPILER` - **advanced option** - Set the C++ compiler for building Trilinos interface (i.e., NVECTOR_TRILINOS and the examples that use it).
Default: The C++ compiler exported from the found Trilinos installation if `USE_XSDK_DEFAULTS=OFF`.
`CMAKE_CXX_COMPILER` or `MPI_CXX_COMPILER` if `USE_XSDK_DEFAULTS=ON`.
Note: It is recommended to use the same compiler that was used to build the Trilinos library.

`TRILINOS_INTERFACE_CXX_COMPILER_FLAGS` - **advanced option** - Set the C++ compiler flags for Trilinos interface (i.e., NVECTOR_TRILINOS and the examples that use it).
Default: The C++ compiler flags exported from the found Trilinos installation if `USE_XSDK_DEFAULTS=OFF`.
`CMAKE_CXX_FLAGS` if `USE_XSDK_DEFAULTS=ON`.
Note: It is recommended to use the same flags that were used to build the Trilinos library.

`SUNDIALS_BUILD_WITH_MONITORING` - Build SUNDIALS with capabilities for fine-grained monitoring of solver progress and statistics. This is primarily useful for debugging.
Default: OFF
Note: Building with monitoring may result in minor performance degradation even if monitoring is not utilized.

`SUNDIALS_BUILD_PACKAGE_FUSED_KERNELS` - Build specialized fused kernels inside CVOID.
Default: OFF

Note: This option is currently only available when building with `CUDA_ENABLE = ON`. Building with fused kernels requires linking to either `libsundials_cvoid_fused_cuda.lib` or `libsundials_cvoid_fused_stub.lib` where the latter provides CPU-only placeholders for the fused routines, in addition to `libsundials_cvoid.lib`.

`CMAKE_CXX_STANDARD` - The C++ standard to build C++ parts of SUNDIALS with.
Default: 11

Note: Options are 99, 11, 14, 17. This option only used when a C++ compiler is required.

`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 option** - 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**

`SUNDIALS_INSTALL_CMAKEDIR` - Installation directory for the SUNDIALS cmake files (relative to `CMAKE_INSTALL_PREFIX`).
Default: `CMAKE_INSTALL_PREFIX/cmake/sundials`

`USE_GENERIC_MATH` - Use generic (stdc) math libraries
Default: **ON**

`USE_XSDK_DEFAULTS` - Enable xSDK (see [for](#) more information) default configuration settings. This sets `CMAKE_BUILD_TYPE` to **Debug**, `SUNDIALS_INDEX_SIZE` to 32 and `SUNDIALS_PRECISION` to **double**.
Default: **OFF**

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/instldir \  
> -DEXAMPLES_INSTALL_PATH=/home/myname/sundials/instldir/examples \  
> -DENABLE_MPI=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 \  
> -DENABLE_MPI=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 any external libraries used with SUNDIALS must also be build as a shared library or as a static library compiled with the `-fPIC` flag.



Building with LAPACK

To enable LAPACK, set the `ENABLE_LAPACK` 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 required for LAPACK.

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

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 5.7.2. To enable KLU, set `ENABLE_KLU` 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 `ENABLE_SUPERLUMT` 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_LIBRARIES` must be set to a semi-colon separated list of other libraries SuperLU_MT depends on. For example, if SuperLU_MT was built with an external blas library, then include the full path to the blas library in this list. Additionally, 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 `ENABLE_OPENMP` or `ENABLE_PTHREAD` set to `ON` then SuperLU_MT should be set to use the same threading type.

Building with SuperLU_DIST

The SuperLU_DIST libraries are available for download from the Lawrence Berkeley National Laboratory website: <http://crd-legacy.lbl.gov/~xiaoye/SuperLU/#superlu-dist>. SUNDIALS has been tested with SuperLU_DIST 6.1.1. To enable SuperLU_DIST, set `ENABLE_SUPERLUDIST` to `ON`, set `SUPERLUDIST_INCLUDE_DIR` to the include directory of the SuperLU_DIST installation (typically `SRC`), and set the variable

`SUPERLUDIST_LIBRARY_DIR` to the path to library directory of the SuperLU_DIST installation (typically `lib`). At the same time, the variable `SUPERLUDIST_LIBRARIES` must be set to a semi-colon separated list of other libraries SuperLU_DIST depends on. For example, if SuperLU_DIST was built with LAPACK, then include the LAPACK library in this list. If SuperLU_DIST was built with OpenMP support, then you may set `SUPERLUDIST_OPENMP` to `ON` to utilize the OpenMP functionality of SuperLU_DIST.



Do not mix thread types when building SUNDIALS solvers. If threading is enabled for SUNDIALS by having `ENABLE_PTHREAD` set to `ON` then SuperLU_DIST should not be set to use OpenMP.

Building with PETSc

The PETSc libraries are available for download from the Argonne National Laboratory website: <http://www.mcs.anl.gov/>. SUNDIALS has been tested with PETSc version 3.10.0–3.14.0. To enable PETSc, set `ENABLE_PETSC` to `ON` and then set `PETSC_DIR` to the path of the PETSc installation. Alternatively, a user can provide a list of include paths in `PETSC_INCLUDES`, and a list of complete paths to the libraries needed in `PETSC_LIBRARIES`.

Building with hypre

The *hypre* libraries are available for download from the Lawrence Livermore National Laboratory website: <http://computing.llnl.gov/projects/hypre>. SUNDIALS has been tested with *hypre* version 2.14.0–2.19.0. To enable *hypre*, set `ENABLE_HYPRE` 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.

Note: SUNDIALS must be configured so that `SUNDIALS_INDEX_SIZE` (or equivalently, `XSDK_INDEX_SIZE`) equals the precision of `HYPRE_BigInt` in the corresponding *hypre* installation.

Building with CUDA

SUNDIALS CUDA modules and examples have been tested with versions 9 through 11.0.2 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 `ENABLE_CUDA` 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 up to version 0.12.1. Building SUNDIALS RAJA modules requires a CUDA-enabled RAJA installation. To enable RAJA, set `ENABLE_CUDA` and `ENABLE_RAJA` 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_CUDA` to `ON`.

Building with Trilinos

Trilinos is a suite of numerical libraries developed by Sandia National Laboratories. It can be obtained at <https://github.com/trilinos/Trilinos>. SUNDIALS Trilinos modules and examples have been tested with Trilinos version 12.14.1 – 12.18.1. To enable Trilinos, set `ENABLE_TRILINOS` to `ON`. If Trilinos is installed in a nonstandard location you will be prompted to set the variable `Trilinos_DIR` with the path to the Trilinos CMake configuration file. It is desirable to build the Trilinos vector interface with same compiler and options that were used to build Trilinos. CMake will try to find the correct compiler settings automatically from the Trilinos configuration file. If that is not successful, the compilers and options can be manually set with the following CMake variables:

- `Trilinos_INTERFACE_C_COMPILER`
- `Trilinos_INTERFACE_C_COMPILER_FLAGS`
- `Trilinos_INTERFACE_CXX_COMPILER`
- `Trilinos_INTERFACE_CXX_COMPILER_FLAGS`

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*/`CMAKE_INSTALL_LIBDIR` and *instdir*/`include`, respectively. The location can be changed by setting the CMake variable `CMAKE_INSTALL_PREFIX`. Although all installed libraries reside under *libdir*/`CMAKE_INSTALL_LIBDIR`, 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.

1.4.1 Using SUNDIALS as a Third Party Library in other CMake Projects

The `make install` command will also install a [CMake package configuration file](#) that other CMake projects can load to get all the information needed to build against SUNDIALS. In the consuming project's CMake code, the `find_package` command may be used to search for the configuration file, which will be installed to `instdir/SUNDIALS_INSTALL_CMAKEDIR/SUNDIALSConfig.cmake` alongside a package version file `instdir/SUNDIALS_INSTALL_CMAKEDIR/SUNDIALSConfigVersion.cmake`. Together these files contain all the information the consuming project needs to use SUNDIALS, including exported CMake targets. The SUNDIALS exported CMake targets follow the same naming convention as the generated library binaries, e.g. the exported target for CVODE is `SUNDIALS::cvcde`. The CMake code snipped below shows how a consuming project might leverage the SUNDIALS package configuration file to build against SUNDIALS in their own CMake project.

```
project(MyProject)

# Set the variable SUNDIALS_DIR to the SUNDIALS instdir.
# When using the cmake CLI command, this can be done like so:
#   cmake -D SUNDIALS_DIR=/path/to/sundials/installation

find_project(SUNDIALS REQUIRED)

add_executable(myexec main.c)

# Link to SUNDIALS libraries through the exported targets.
# This is just an example, users should link to the targets appropriate
# for their use case.
target_link_libraries(myexec PUBLIC SUNDIALS::cvcde SUNDIALS::nvecpetsc)
```

Table 1.1: SUNDIALS libraries and header files

SHARED	Libraries	n/a
	Header files	sundials/sundials_config.h sundials/sundials_fconfig.h sundials/sundials_types.h sundials/sundials_math.h sundials/sundials_nvector.h sundials/sundials_fnvector.h sundials/sundials_matrix.h sundials/sundials_linearsolver.h sundials/sundials_iterative.h sundials/sundials_direct.h sundials/sundials_dense.h sundials/sundials_band.h sundials/sundials_nonlinearsolver.h sundials/sundials_version.h sundials/sundials_mpi_types.h sundials/sundials_cuda_policies.hpp
NVECTOR_SERIAL	Libraries	libsundials_nvecserial. <i>lib</i> libsundials_fnvecserial_mod. <i>lib</i> libsundials_fnvecserial.a
	Header files	nvector/nvector_serial.h
	Module files	fnvector_serial_mod.mod
NVECTOR_PARALLEL	Libraries	libsundials_nvecparallel. <i>lib</i> libsundials_fnvecparallel.a libsundials_fnvecparallel_mod. <i>lib</i>
	Header files	nvector/nvector_parallel.h
	Module files	fnvector_parallel_mod.mod
NVECTOR_MANYVECTOR	Libraries	libsundials_nvecmanyvector. <i>lib</i> libsundials_nvecmanyvector_mod. <i>lib</i>
	Header files	nvector/nvector_manyvector.h
	Module files	fnvector_manyvector_mod.mod
NVECTOR_MPIMANYVECTOR	Libraries	libsundials_nvecmpimanyvector. <i>lib</i> libsundials_nvecmpimanyvector_mod. <i>lib</i>
	Header files	nvector/nvector_mpimanyvector.h
	Module files	fnvector_mpimanyvector_mod.mod
continued on next page		

<i>continued from last page</i>		
NVECTOR_MPIPLUSX	Libraries	libsundials_nvecmpiplusx. <i>lib</i> libsundials_nvecmpiplusx_mod. <i>lib</i>
	Header files	nvector/nvector_mpiplusx.h
	Module files	fnvector_mpiplusx_mod.mod
NVECTOR_OPENMP	Libraries	libsundials_nvecopenmp. <i>lib</i> libsundials_fnvecopenmp_mod. <i>lib</i> libsundials_fnvecopenmp.a
	Header files	nvector/nvector_openmp.h
	Module files	fnvector_openmp_mod.mod
NVECTOR_OPENMPDEV	Libraries	libsundials_nvecopenmpdev. <i>lib</i>
	Header files	nvector/nvector_openmpdev.h
NVECTOR_PTHREADS	Libraries	libsundials_nvecpthreads. <i>lib</i> libsundials_fnvecpthreads_mod. <i>lib</i> libsundials_fnvecpthreads.a
	Header files	nvector/nvector_pthreads.h
	Module files	fnvector_pthreads_mod.mod
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>
	Header files	nvector/nvector_cuda.h
NVECTOR_RAJA	Libraries	libsundials_nveccudaraja. <i>lib</i> libsundials_nvechipraja. <i>lib</i>
	Header files	nvector/nvector_raja.h
NVECTOR_TRILINOS	Libraries	libsundials_nvectrilinos. <i>lib</i>
	Header files	nvector/nvector_trilinos.h nvector/trilinos/SundialsTpetraVectorInterface.hpp nvector/trilinos/SundialsTpetraVectorKernels.hpp
SUNMATRIX_BAND	Libraries	libsundials_sunmatrixband. <i>lib</i> libsundials_fsunmatrixband_mod. <i>lib</i> libsundials_fsunmatrixband.a
	Header files	sunmatrix/sunmatrix_band.h
	Module files	fsunmatrix_band_mod.mod
SUNMATRIX_DENSE	Libraries	libsundials_sunmatrixdense. <i>lib</i> libsundials_fsunmatrixdense_mod. <i>lib</i> libsundials_fsunmatrixdense.a
	Header files	sunmatrix/sunmatrix_dense.h
<i>continued on next page</i>		

<i>continued from last page</i>		
	Module files	fsunmatrix_dense_mod.mod
SUNMATRIX_SPARSE	Libraries	libsundials_sunmatrixsparse.lib libsundials_fsunmatrixsparse_mod.lib libsundials_fsunmatrixsparse.a
	Header files	sunmatrix/sunmatrix_sparse.h
	Module files	fsunmatrix_sparse_mod.mod
SUNMATRIX_SLUNRLOC	Libraries	libsundials_sunmatrixslunrloc.lib
	Header files	sunmatrix/sunmatrix_slunrloc.h
SUNLINSOL_CUSPARSE	Libraries	libsundials_sunmatrixcusparse.lib
	Header files	sunmatrix/sunmatrix_cusparse.h
SUNLINSOL_BAND	Libraries	libsundials_sunlinsolband.lib libsundials_fsunlinsolband_mod.lib libsundials_fsunlinsolband.a
	Header files	sunlinsol/sunlinsol_band.h
	Module files	fsunlinsol_band_mod.mod
SUNLINSOL_DENSE	Libraries	libsundials_sunlinsoldense.lib libsundials_fsunlinsoldense_mod.lib libsundials_fsunlinsoldense.a
	Header files	sunlinsol/sunlinsol_dense.h
	Module files	fsunlinsol_dense_mod.mod
SUNLINSOL_KLU	Libraries	libsundials_sunlinsolklu.lib libsundials_fsunlinsolklu_mod.lib libsundials_fsunlinsolklu.a
	Header files	sunlinsol/sunlinsol_klu.h
	Module files	fsunlinsol_klu_mod.mod
SUNLINSOL_LAPACKBAND	Libraries	libsundials_sunlinsollapackband.lib libsundials_fsunlinsollapackband.a
	Header files	sunlinsol/sunlinsol_lapackband.h
SUNLINSOL_LAPACKDENSE	Libraries	libsundials_sunlinsollapackdense.lib libsundials_fsunlinsollapackdense.a
	Header files	sunlinsol/sunlinsol_lapackdense.h
SUNLINSOL_PCG	Libraries	libsundials_sunlinsolpcg.lib libsundials_fsunlinsolpcg_mod.lib libsundials_fsunlinsolpcg.a
	Header files	sunlinsol/sunlinsol_pcg.h
	Module files	fsunlinsol_pcg_mod.mod
<i>continued on next page</i>		

<i>continued from last page</i>		
SUNLINSOL_SPBCGS	Libraries	libsundials_sunlinsolspbcgs. <i>lib</i> libsundials_fsunlinsolspbcgs_mod. <i>lib</i> libsundials_fsunlinsolspbcgs.a
	Header files	sunlinsol/sunlinsol_spbcgs.h
	Module files	fsunlinsol_spbcgs_mod.mod
SUNLINSOL_SPFGMR	Libraries	libsundials_sunlinsolspfgmr. <i>lib</i> libsundials_fsunlinsolspfgmr_mod. <i>lib</i> libsundials_fsunlinsolspfgmr.a
	Header files	sunlinsol/sunlinsol_spfgmr.h
	Module files	fsunlinsol_spfgmr_mod.mod
SUNLINSOL_SPGMR	Libraries	libsundials_sunlinsolspgmr. <i>lib</i> libsundials_fsunlinsolspgmr_mod. <i>lib</i> libsundials_fsunlinsolspgmr.a
	Header files	sunlinsol/sunlinsol_spgmr.h
	Module files	fsunlinsol_spgmr_mod.mod
SUNLINSOL_SPTFQMR	Libraries	libsundials_sunlinsolsptfqmr. <i>lib</i> libsundials_fsunlinsolsptfqmr_mod. <i>lib</i> libsundials_fsunlinsolsptfqmr.a
	Header files	sunlinsol/sunlinsol_sptfqmr.h
	Module files	fsunlinsol_sptfqmr_mod.mod
SUNLINSOL_SUPERLUMT	Libraries	libsundials_sunlinsolsuperlumt. <i>lib</i> libsundials_fsunlinsolsuperlumt.a
	Header files	sunlinsol/sunlinsol_superlumt.h
SUNLINSOL_SUPERLUDIST	Libraries	libsundials_sunlinsolsuperludist. <i>lib</i>
	Header files	sunlinsol/sunlinsol_superludist.h
SUNLINSOL_CUSOLVERSP_BATCHQR	Libraries	libsundials_sunlinsolcusolversp. <i>lib</i>
	Header files	sunlinsol/sunlinsol_cusolverp_batchqr.h
SUNNONLINSOL_NEWTON	Libraries	libsundials_sunnonlinsolnewton. <i>lib</i> libsundials_fsunnonlinsolnewton_mod. <i>lib</i> libsundials_fsunnonlinsolnewton.a
	Header files	sunnonlinsol/sunnonlinsol_newton.h
	Module files	fsunnonlinsol_newton_mod.mod
SUNNONLINSOL_FIXEDPOINT	Libraries	libsundials_sunnonlinsolfixedpoint. <i>lib</i> libsundials_fsunnonlinsolfixedpoint.a libsundials_fsunnonlinsolfixedpoint_mod. <i>lib</i>
	Header files	sunnonlinsol/sunnonlinsol_fixedpoint.h
	Module files	fsunnonlinsol_fixedpoint_mod.mod
<i>continued on next page</i>		

<i>continued from last page</i>		
SUNNONLINSOL_PETSCSNES	Libraries	libsundials_sunnonlinsolpetscsnes. <i>lib</i>
	Header files	sunnonlinsol/sunnonlinsol_petscsnes.h
CVODE	Libraries	libsundials_cvode. <i>lib</i> libsundials_fcvcde.a libsundials_fcvcde_mod. <i>lib</i>
	Header files	cvode/cvode.h cvode/cvode_direct.h cvode/cvode_spils.h cvode/cvode_bbdpre.h
		cvode/cvode_impl.h cvode/cvode_ls.h cvode/cvode_bandpre.h
	Module files	fcvcde_mod.mod
CVODES	Libraries	libsundials_cvodes. <i>lib</i> libsundials_fcvcodes_mod. <i>lib</i>
	Header files	cvodes/cvodes.h cvodes/cvodes_direct.h cvodes/cvodes_spils.h cvodes/cvodes_bbdpre.h
		cvodes/cvodes_impl.h cvodes/cvodes_ls.h cvodes/cvodes_bandpre.h
	Module files	fcvcodes_mod.mod
ARKODE	Libraries	libsundials_arkode. <i>lib</i> libsundials_farkode.a libsundials_farkode_mod. <i>lib</i>
	Header files	arkode/arkode.h arkode/arkode_ls.h arkode/arkode_bbdpre.h
		arkode/arkode_impl.h arkode/arkode_bandpre.h
	Module files	farkode_mod.mod farkode_erkstep_mod.mod farkode_mrstep_mod.mod
IDA	Libraries	libsundials_ida. <i>lib</i> libsundials_fida.a libsundials_fida_mod. <i>lib</i>
	Header files	ida/ida.h ida/ida_direct.h ida/ida_spils.h
		ida/ida_impl.h ida/ida_ls.h ida/ida_bbdpre.h
	Module files	fida_mod.mod
IDAS	Libraries	libsundials_idas. <i>lib</i> libsundials_fidas_mod. <i>lib</i>
	Header files	idas/idas.h idas/idas_direct.h idas/idas_spils.h
		idas/idas_impl.h idas/idas_ls.h idas/idas_bbdpre.h
	Module files	fidas_mod.mod
<i>continued on next page</i>		

<i>continued from last page</i>		
KINSOL	Libraries	libsundials_kinsol. <i>lib</i> libsundials_fkinsol.a libsundials_fkinsol_mod. <i>lib</i>
	Header files	kinsol/kinsol.h kinsol/kinsol_impl.h kinsol/kinsol_direct.h kinsol/kinsol_ls.h kinsol/kinsol_spils.h kinsol/kinsol_bbdpre.h
	Module files	fkinsol_mod.mod

