Xcode Build Settings Reference

All variables are prefixed with "$" to uniquely identify them on this page. Submitting updates to this page.

$ACTION

Description	Identifies the type of build to perform on the target.
Type	String
Values	• `build` : Build the product and place it in the product build directory `$(CONFIGURATION_BUILD_DIR)` . • `clean` : Remove the product and build files in the product build directory `$(CONFIGURATION_BUILD_DIR)` and the intermediate build files directory `$(CONFIGURATION_TEMP_DIR)` . • `install` : Build the product and place it in its installation destination `$(INSTALL_PATH)` . • `installhdrs` : Copy the product's public and private header files into the public headers directory `$(PUBLIC_HEADERS_FOLDER_PATH)` and the private headers directory `$(PRIVATE_HEADERS_FOLDER_PATH)` , respectively. • `installsrc` : Copy the target's source files into the project directory `$(SRCROOT)` .
Default Value	`build`

$AD_HOC_CODE_SIGNING_ALLOWED

Description	Internal setting used by Xcode to determine if ad-hoc signing identities can be used.
Type	Boolean
Default Value	`NO`

$ADDITIONAL_SDKS

Description	The locations of any sparse SDKs that should be layered on top of the one specified by `$(SDKROOT)` . If more than one SDK is listed, the first one has highest precedence. Every SDK specified in this setting should be a "sparse" SDK, i.e. not an SDK for an entire OS X release.
Type	String
Default Value	empty string

$ALTERNATE_GROUP

Description	The group name or gid for the files listed under the `$(ALTERNATE_PERMISSIONS_FILES)` setting.
Type	String
Default Value	`$(INSTALL_GROUP)`

$ALTERNATE_MODE

Description	Permissions used for the the files listed under the `$(ALTERNATE_PERMISSIONS_FILES)` setting.
Type	String
Default Value	`$(INSTALL_MODE_FLAG)`

$ALTERNATE_OWNER

Description	The owner name or uid for the files listed under the `$(ALTERNATE_PERMISSIONS_FILES)` setting.
Type	String
Default Value	`$(INSTALL_OWNER)`

$ALTERNATE_PERMISSIONS_FILES

Description	List of files to which the alternate owner, group and permissions are applied.
Type	StringList
Default Value	empty string

$ALWAYS_SEARCH_USER_PATHS

Description	Specifies whether the compiler searches for headers in the project directory before searching system directories.
Type	Boolean
Values	• `YES` : Search project directory first. • `NO` : Search system directories first.
Default Value	`YES`

$ALWAYS_USE_SEPARATE_HEADERMAPS

Description	
Type	Boolean
Values	• •
Default Value	

$APPLE_INTERNAL_DEVELOPER_DIR

Type	Path
Default Value	`$(APPLE_INTERNAL_DIR)/Developer`

$APPLE_INTERNAL_DIR

Type	Path
Default Value	`/AppleInternal`

$APPLE_INTERNAL_DOCUMENTATION_DIR

Type	Path
Default Value	`$(APPLE_INTERNAL_DIR)/Documentation`

$APPLE_INTERNAL_LIBRARY_DIR

Type	Path
Default Value	`$(APPLE_INTERNAL_DIR)/Library`

$APPLE_INTERNAL_TOOLS

Type	Path
Default Value	`$(APPLE_INTERNAL_DEVELOPER_DIR)/Tools`

$APPLICATION_EXTENSION_API_ONLY

Description	When enabled, this causes the compiler and linker to disallow use of APIs that are not available to app extensions and to disallow linking to frameworks that have not been built with this setting enabled. If enabled, passes flag `-fapplication-extension` .
Type	Boolean
Default Value	`NO`

$APPLY_RULES_IN_COPY_FILES

Description	Files copied with a 'Copy Files Build Phase' are unchanged by default.
Type	Boolean
Values	• Setting this to YES causes Property Lists (.plist) and Strings files to be converted as specified by 'Property List Output Encoding' and 'Strings file Output Encoding'.
Default Value	`NO`

$ARCHS

Description	Space-separated list of identifiers. Specifies the architectures (ABIs, processor models) to which the binary is targeted. When this build setting specifies more than one architecture, the generated binary may contain object code for each of the specified architectures.
Type	String
Default Value	`$(ARCHS_STANDARD)`
Example Value	`x86_64`

$ARCHS_STANDARD

Description	Standard Architectures.
Type	StringList
Values	• When `$(SDKROOT)=iphoneos` : `armv7 arm64` • When `$(SDKROOT)=iphonesimulator` : `i386 x86_64` • When `$(SDKROOT)=macosx` : `x86_64`
Example Value	`$(ARCHS_STANDARD_INCLUDING_64_BIT)`

$ARCHS_STANDARD_32_64_BIT

Description	Universal (32/64-bit)
Type	StringList
Default Value	`$(ARCHS_STANDARD_32_BIT) $(ARCHS_STANDARD_64_BIT)`
Example Value	`i386 x86_64`

$ARCHS_STANDARD_32_BIT

Description	32-bit Architecture
Type	StringList
Values	• When `sdk=macosx` : `i386` • When `sdk=iphoneos` : `armv7` • When `sdk=iphonesimulator` : `i386`

$ARCHS_STANDARD_64_BIT

Description	64-bit Architecture
Type	StringList
Values	• When `sdk=macosx` : `x86_64` • When `sdk=iphoneos` : `arm64` • When `sdk=iphonesimulator` : `x86_64`

$ARCHS_STANDARD_INCLUDING_64_BIT

Description	Standard Architectures, and 64-bit Architectures
Type	StringList
Values	• When `sdk=macosx` : `x86_64` • When `sdk=iphoneos` : `arm64` • When `sdk=iphonesimulator` : `x86_64`

$ARCHS_UNIVERSAL_IPHONE_OS

Description	Universal Architectures for iPhoneOS
Type	StringList
Values	• When `sdk=iphoneos` : `armv7 arm64` • When `sdk=iphonesimulator` : `i386 x86_64`

$ASSETCATALOG_COMPILER_APPICON_NAME

Description	Name of the asset catalog app icon set whose contents will be merged into the Info.plist. Passes flag `--app-icon` .
Type	String
Default Value	empty string

$ASSETCATALOG_COMPILER_LAUNCHIMAGE_NAME

Description	Name of the asset catalog launch image set whose contents will be merged into the Info.plist. Passes flag `--launch-image` .
Type	String
Default Value	empty string

$ASSETCATALOG_NOTICES

Description	Show notices encountered during the compilation of asset catalogs. Passes flag `--notices` .
Type	Boolean
Default Value	`YES`

$ASSETCATALOG_OTHER_FLAGS

Description	Pass additional flags through to the asset catalog compiler.
Type	StringList
Default Value	empty string

$ASSETCATALOG_WARNINGS

Description	Show warnings encountered during the compilation of asset catalogs. Passes flag `--warnings` .
Type	Boolean
Default Value	`YES`

$AVAILABLE_PLATFORMS

Description	Space-separated list of platform bundles installed in Xcode's Developer directory.
Type	String
Default Value	`iphonesimulator macosx iphoneos`

$BUILD_COMPONENTS

Description	Space-separated list of identifiers. Specifies subsets of the product.
Type	String
Values	• When `$(ACTION)=build` : `headers build` • When `$(ACTION)=install` : `headers build` • When `$(ACTION)=installhdrs` : `headers` • When `$(ACTION)=installsrc` : empty

$BUILD_DIR

Description	Alias for `$(SYMROOT)`
Type	Path
Default Value	`$(SYMROOT)`

$BUILD_ROOT

Description	Alias for `$(SYMROOT)`
Type	Path
Default Value	`$(SYMROOT)`

$BUILD_STYLE

Description	Name of current build style.
Type	String
Default Value	empty string

$BUILD_VARIANTS

Description	Space-separated list of identifiers. Specifies the binary variants of the product. You can create additional variant names for special purposes. For example, you can use the name of a build configuration as a variant name to create highly customized binaries.
Type	String
Values	• `normal` : Use to produce a normal binary. • `profile` : Use to produce a binary that generates profile information. • `debug` : Use to produce a binary with debug symbols, additional assertions, and diagnostic code.
Default Value	`normal`

$BUILT_PRODUCTS_DIR

Description	Identifies the directory under which all the product's files can be found. This directory contains either product files or symbolic links to them. Run Script build phases can use the value of this build setting as a convenient way to refer to the product files built by one or more targets even when these files are scattered throughout a directory hierarchy (for example, when `$(DEPLOYMENT_LOCATION)` is set to YES).
Type	Path
Values	• When `$(DEPLOYMENT_LOCATION)=YES` and `$(RETAIN_RAW_BINARIES)=YES` : `$(SYMROOT)/BuiltProducts` • Otherwise: `$(CONFIGURATION_BUILD_DIR)`

$BUNDLE_LOADER

Description	Passes flag `-bundle_loader` to the linker.
Type	String
Default Value	empty string

$CACHE_ROOT

Description	Alias for `$(CCHROOT)`.
Type	Path
Default Value	`$(CCHROOT)`
Example Value	`/var/folders/m5/j4zdc7f9157659_pd5p0_n980000gn/C/com.apple.DeveloperTools/6.2-6C131e/Xcode`

$CCHROOT

Description	The file used to cache build-time information that must persist between launches of the Xcode application.
Type	Path
Default Value	`confstr('CS_DARWIN_USER_CACHE_DIR')/com.apple.DeveloperTools/$(XCODE_PRODUCT_VERSION)-$(XCODE_PRODUCT_BUILD_VERSIO`

$CHMOD

Description	Path to `chmod` tool.
Type	Path
Default Value	`/bin/chmod`

$CHOWN

Description	Path to `chown` tool.
Type	Path
Default Value	`/usr/sbin/chown`

$CLANG_ALLOW_NON_MODULAR_INCLUDES_IN_FRAMEWORK_MODULES

Description	Enabling this setting allows non-modular includes to be used from within framework modules. This is inherently unsafe, as such headers might cause duplicate definitions when used by any client that imports both the framework and the non-modular includes themselves. If disabled, passes flags: • `-Wnon-modular-include-in-framework-module` • `-Werror=non-modular-include-in-framework-module`
Type	Boolean
Default Value	`NO`

$CLANG_ANALYZER_DEADCODE_DEADSTORES

Description	Check for values stored to variables that are never read afterwards. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang deadcode.DeadStores`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_GCD

Description	Check for misuses of Grand Central Dispatch API. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.API`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_MALLOC

Description	See `$(CLANG_ANALYZER_MEMORY_MANAGEMENT)`.
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_MEMORY_MANAGEMENT

Description	Warn about memory leaks, use-after-free, as well as other API misuse. If disabled passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang unix.Malloc` • `-Xclang -analyzer-disable-checker` • `-Xclang unix.MallocSizeof` • `-Xclang -analyzer-disable-checker` • `-Xclang unix.MismatchedDeallocator` • `-Xclang -analyzer-disable-checker` • `-Xclang cplusplus.NewDelete`
Type	Boolean
Default Value	`$(CLANG_ANALYZER_MALLOC)`

$CLANG_ANALYZER_OBJC_ATSYNC

Description	Warn on nil pointers used as mutexes for @synchronized. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.AtSync`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_COLLECTIONS

Description	Warn if CF collections are created with non-pointer-size values. Check if NS collections are initialized with non-Objective-C type elements. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.coreFoundation.containers.PointerSizedValues` • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.VariadicMethodTypes` • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.NilArg`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_INCOMP_METHOD_TYPES

Description	Warn about Objective-C method signatures with type incompatibilities. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.IncompatibleMethodTypes`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_NSCFERROR

Description	Warn if functions accepting CFErrorRef or NSError cannot indicate if error occured. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.coreFoundation.CFError` • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.NSError`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_RETAIN_COUNT

Description	Warn on leaks and improper reference count management. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.RetainCount` • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.ClassRelease`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_SELF_INIT

Description	Check that [super init] is properly called within an Objective-C initialization method. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.SelfInit`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_OBJC_UNUSED_IVARS

Description	Warn about private ivars that are never used. If disabled, passes flags: • `-Xclang -analyzer-disable-checker` • `-Xclang osx.cocoa.UnusedIvars`
Type	Boolean
Default Value	`YES`

$CLANG_ANALYZER_SECURITY_FLOATLOOPCOUNTER

Description	Warn on using a floating point value as a loop counter (CERT: FLP30-C, FLP30-CPP). If disabled, passes flags: • `-Xclang -analyzer-checker` • `-Xclang security.FloatLoopCounter`
Type	Boolean
Default Value	`NO`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_GETPW_GETS

Description	Warn on uses of 'getpw' and 'gets'. The functions are dangerous as they may trigger a buffer overflow.
Type	Boolean
Values	• `YES` : * `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.getpw` * `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.gets` • `NO` : `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.getpw` * `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.gets`
Default Value	`YES`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_MKSTEMP

Description	Warn on uses of 'mktemp', which produces predictable temporay files. It is obsoleted by 'mktemps'. Warn when 'mkstemp' is passed fewer than 6 X's in the format string.
Type	Boolean
Values	• `YES` :* `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.mkstemp` * `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.mktemp` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.mkstemp` * `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.mktemp`
Default Value	`YES`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_RAND

Description	Warn on uses of 'rand', 'random', and related functions which produce predictable random number sequences. Use arc4random instead.
Type	Boolean
Values	• `YES` :* `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.rand` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.rand`
Default Value	`NO`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_STRCPY

Description	Warn on uses of the 'strcpy' and 'strcat' functions, which can result in buffer overflows. Use 'strlcpy' or 'strlcat' instead.
Type	Boolean
Values	• `YES` :* `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.strcpy` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.strcpy`
Default Value	`NO`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_UNCHECKEDRETURN

Description	Warn on uses of sensitive functions whose return values must be always checked.
Type	Boolean
Values	• `YES` :* `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.UncheckedReturn` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.UncheckedReturn`
Default Value	`YES`

$CLANG_ANALYZER_SECURITY_INSECUREAPI_VFORK

Description	Warn on uses of the 'vfork' function, which is inherently insecure. Use the safer 'posix_spawn' function instead.
Type	Boolean
Values	• `YES` :* `-Xclang -analyzer-checker` * `-Xclang security.insecureAPI.vfork` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang security.insecureAPI.vfork`
Default Value	`YES`

$CLANG_ANALYZER_SECURITY_KEYCHAIN_API

Description	Check for misuse of Keychain Services API.
Type	Boolean
Values	• `YES` • `NO` :* `-Xclang -analyzer-disable-checker` * `-Xclang osx.SecKeychainAPI`
Default Value	`YES`

$CLANG_ARC_MIGRATE_DIR

Description	Passes flag `-ccc-arcmt-migrate`
Type	Path
Default Value	empty string

$CLANG_ARC_MIGRATE_EMIT_ERROR

Description	If enabled, passes flag `-arcmt-migrate-emit-errors`
Type	Boolean
Default Value	`NO`

$CLANG_ARC_MIGRATE_PRECHECK

Description	Used by Xcode's migration tool, do not edit.
Type	Enumeration
Values	• `donothing` : Does nothing • `precheck` : Passes flag `-ccc-arcmt-check`
Default Value	`donothing`

$CLANG_ARC_MIGRATE_REPORT_OUTPUT

Description	Passes flag `-arcmt-migrate-report-output`
Type	Path
Default Value	empty string

$CLANG_COLOR_DIAGNOSTICS

Description	If enabled passes flag `-fcolor-diagnostics` . Note: this flag does NOT impact PCH or compilation. The build system makes special efforts to ignore for dependency tracking.
Type	Boolean
Default Value	`$(COLOR_DIAGNOSTICS)`

$CLANG_CXX_LANGUAGE_STANDARD

Description	Choose a standard or non-standard C++ language dialect.
Type	Enumeration
Values	• `c++98` : Passes flag `-std=c++98` C++98: Accept ISO C++ 1998 with amendments, but not GNU extensions. • `gnu++98` : Passes flag `-std=gnu++98` GNU++98: Accept ISO C++ 1998 with amendments and GNU extensions. • `c++0x` : Passes flag `-std=c++11` C++11: Accept the ISO C++ 2011 standard with amendments, but not GNU extensions. • `gnu++0x` : Passes flag `-std=gnu++11` GNU++11: Accept the ISO C++ 2011 standard with amendments and GNU extensions. • `c++14` : Passes flag `-std=c++1y` C++14: Accept the ISO C++ 2014 standard with amendments, but not GNU extensions. • `gnu++14` : Passes flag `-std=gnu++1y` GNU++14: Accept the ISO C++ 2014 standard with amendments and GNU extensions. • `compiler-default` : Compiler Default: Tells the compiler to use its default C++ language dialect. This is normally the best choice unless you have specific needs. (Currently equivalent to GNU++98.)
Default Value	`compiler-default`

$CLANG_CXX_LIBRARY

Description	Choose a version of the C++ standard library to use.
Type	Enumeration
Values	• `libstdc++` : Passes flag `-stdlib=libstdlibc++` traditional C++ standard library that works with GCC and the LLVM Compiler • `libc++` : Passes flag `-stdlib=libc++` highly optimized C++ standard library that works only with the LLVM Compiler, and is designed to support new C++11 features. • `compiler-default` : Uses `libstdc++`
Default Value	`compiler-default`

$CLANG_DEBUG_INFORMATION_LEVEL

Description	Toggles the amount of debug information emitted when debug symbols are enabled. This can impact the size of the generated debug information, which can matter in some cases for large projects (such as when using LTO).
Type	Enumeration
Values	• `default` : • `line-tables-only` : Passes flag `-gline-tables-only`
Default Value	`default`

$CLANG_ENABLE_APP_EXTENSION

Description	If enabled, passes flag `-fapplication-extension`
Type	Boolean
Default Value	`$(APPLICATION_EXTENSION_API_ONLY)`

$CLANG_ENABLE_MODULES

Description	If enabled, passes flag `-fmodules` .
Type	Boolean
Default Value	`NO`

$CLANG_ENABLE_MODULE_IMPLEMENTATION_OF

Description	If enabled, passes flag `-fmodule-implementation-of $(PRODUCT_MODULE_NAME)`
Type	Boolean
Default Value	`YES`

$CLANG_ENABLE_OBJC_ARC

Description	If enabled, passes flag `-fobjc-arc` .
Type	Boolean
Default Value	`YES`

$CLANG_INSTRUMENT_FOR_OPTIMIZATION_PROFILING

Description	If enabled, passes flag `-fprofile-instr-generate` and `-fprofile-instr-generate`
Type	Boolean
Default Value	`NO`

$CLANG_LINK_OBJC_RUNTIME

Description	If enabled, passes flag `-fobjc-link-runtime`. This option is passed for linking to inform the compiler that the ObjC runtime must be linked in (with possible backwards compatibility libraries linked in).
Type	Boolean
Default Value	`YES`

$CLANG_MACRO_BACKTRACE_LIMIT

Description	`-fmacro-backtrace-limit=$(value)`
Type	Integer
Default Value	`0`

$CLANG_MODULES_AUTOLINK

Description	If disabled, passes flag `-fno-autolink`.
Type	Boolean
Default Value	`YES`

$CLANG_MODULES_IGNORE_MACROS

Description	If `$(CLANG_ENABLE_MODULES)` is enabled, passes flag `-fmodules-ignore-macro=$(value)`.
Type	StringList
Default Value	`$(GCC_PREPROCESSOR_DEFINITIONS_NOT_USED_IN_PRECOMPS)`

$CLANG_MODULES_VALIDATE_SYSTEM_HEADERS

Description	If `$(CLANG_ENABLE_MODULES)` is enabled, if enabled passes flag `-fmodules-validate-system-headers`.
Type	Boolean
Default Value	`NO`

$CLANG_MODULES_VALIDATION_TIMESTAMP

Description	If not empty string: Passes flags `-fbuild-session-timestamp=$(value)` and `-fmodules-validate-once-per-build-session`
Type	String
Default Value	empty string

$CLANG_MODULE_CACHE_PATH

Description	Passes flag `-fmodules-cache-path=$(CLANG_MODULE_CACHE_PATH)` if `$(CLANG_ENABLE_MODULES)` is enabled.
Type	Path
Default Value	`$(MODULE_CACHE_DIR)`
Example Value	`/Users/genica/Library/Developer/Xcode/DerivedData/ModuleCache`

$CLANG_OBJC_MIGRATE_DIR

Description	Passes flag `-ccc-objcmt-migrate`
Type	Path
Default Value	empty string

$CLANG_OPTIMIZATION_PROFILE_FILE

Description	The path to the file of the profile data to use when 'Use Optimization Profile' is enabled.
Type	Path
Default Value	`$(SRCROOT)/OptimizationProfiles/$(PROJECT_NAME).profdata`
Example Value	`/Users/genica/MyProject/OptimizationProfiles/MyProject.profdata`

$CLANG_RETAIN_COMMENTS_FROM_SYSTEM_HEADERS

Description	If enabled, passes flag `-fretain-comments-from-system-headers`. Note: this flag impacts PCH.
Type	Boolean
Default Value	`NO`

$CLANG_STATIC_ANALYZER_MODE

Description	The depth the static analyzer uses during the Build action.
Type	Enumeration
Values	• `shallow` : Use Shallow for faster analysis. Passes flags: * `-Xclang -analyzer-config` * `-Xclang mode=shallow` • `deep` : Use Deep to exercise the full power of the analyzer.
Default Value	`shallow`

$CLANG_STATIC_ANALYZER_MODE_ON_ANALYZE_ACTION

Description	The depth the static analyzer uses during the Analyze action.
Type	Enumeration
Values	• `shallow` : Use Shallow for faster analysis. • `deep` : Use Deep to exercise the full power of the analyzer.
Default Value	`deep`

$CLANG_USE_OPTIMIZATION_PROFILE

Description	When this setting is enabled, clang will use the optimization profile collected for a target when building it.
Type	Boolean
Values	• `YES` : Passes flag: `-fprofile-instr-use=$(CLANG_OPTIMIZATION_PROFILE_FILE)`
Default Value	`NO`

$CLANG_WARN_ASSIGN_ENUM

Description	Warn about assigning integer constants to enum values that are out of the range of the enumerated type. If enabled, passes flag `-Wassign-enum`
Type	Boolean
Default Value	`NO`

$CLANG_WARN_BOOL_CONVERSION

Description	Warn about implicit conversions to boolean values that are suspicious. For example, writing 'if (foo)' with 'foo' being the name a function will trigger a warning.
Type	Boolean
Values	• `YES` : Passes flag `-Wbool-conversion` • `NO` : Passes flag `-Wno-bool-conversion`
Default Value	`$(CLANG_WARN_SUSPICIOUS_IMPLICIT_CONVERSION)`

$CLANG_WARN_CONSTANT_CONVERSION

Description	Warn about implicit conversions of constant values that cause the constant value to change, either through a loss of precision, or entirely in its meaning.
Type	Boolean
Values	• `YES` : Passes flag `-Wconstant-conversion` • `NO` : Passes flag `-Wno-constant-conversion`
Default Value	`$(CLANG_WARN_SUSPICIOUS_IMPLICIT_CONVERSION)`

$CLANG_WARN_CXX0X_EXTENSIONS

Description	When compiling C++ code using a language standard older than C++11, warn about the use of C++11 extensions.
Type	Boolean
Values	• `YES` : Passes flag `-Wc++11-extensions` • `NO` : Passes flag `-Wno-c++11-extensions`
Default Value	`NO`

$CLANG_WARN_DEPRECATED_OBJC_IMPLEMENTATIONS

Description	Warn if an Objective-C class either subclasses a deprecated class or overrides a method that has been marked deprecated.
Type	Boolean
Values	• `YES` : Passes flag `-Wdeprecated-implementations` • `NO` : Passes flag `-Wno-deprecated-implementations`
Default Value	`NO`

$CLANG_WARN_DIRECT_OBJC_ISA_USAGE

Description	Warn about direct accesses to the Objective-C 'isa' pointer instead of using a runtime API.
Type	Enumeration
Values	• `YES` : • `YES_ERROR` : Passes flag `-Werror=deprecated-objc-isa-usage` • `NO` : Passes flag `-Wno-deprecated-objc-isa-usage`
Default Value	`YES_ERROR`

$CLANG_WARN_DOCUMENTATION_COMMENTS

Description	Warns about issues in documentation comments (doxygen-style) such as missing or incorrect documentation tags. If enabled, passes flag: `-Wdocumentation`
Type	Boolean
Default Value	`NO`

$CLANG_WARN_EMPTY_BODY

Description	Warn about loop bodies that are suspiciously empty.
Type	Boolean
Values	• `YES` : Passes flag `-Wempty-body` • `NO` : Passes flag `-Wno-empty-body`
Default Value	`NO`

$CLANG_WARN_ENUM_CONVERSION

Description	Warn about implicit conversions between different kinds of enum values. For example, this can catch issues when using the wrong enum flag as an argument to a function or method.
Type	Boolean
Values	• `YES` : Passes flag `-Wenum-conversion` • `NO` : Passes flag `-Wno-enum-conversion`
Default Value	`$(CLANG_WARN_SUSPICIOUS_IMPLICIT_CONVERSION)`

$CLANG_WARN_IMPLICIT_SIGN_CONVERSION

Description	Warn about implicit integer conversions that change the signedness of an integer value.
Type	Boolean
Values	• `YES` : Passes flag `-Wsign-conversion` • `NO` : Passes flag `-Wno-sign-conversion`
Default Value	`NO`

$CLANG_WARN_INT_CONVERSION

Description	Warn about implicit conversions between pointers and integers. For example, this can catch issues when one incorrectly intermixes using NSNumber*'s and raw integers.
Type	Boolean
Values	• `YES` : Passes flag `-Wint-conversion` • `NO` : Passes flag `-Wno-int-conversion`
Default Value	`$(CLANG_WARN_SUSPICIOUS_IMPLICIT_CONVERSION)`

$CLANG_WARN_OBJC_EXPLICIT_OWNERSHIP_TYPE

Description	If enabled, passes flag `-Wexplicit-ownership-type`
Type	Boolean
Default Value	`NO`

$CLANG_WARN_OBJC_IMPLICIT_ATOMIC_PROPERTIES

Description	Warn about @property declarations that are implicitly atomic.
Type	Boolean
Values	• `YES` : Passes flag `-Wimplicit-atomic-properties` • `NO` : Passes flag `-Wno-implicit-atomic-properties`
Default Value	`NO`

$CLANG_WARN_OBJC_IMPLICIT_RETAIN_SELF

Description	Warn about implicit retains of 'self' within blocks, which can create a retain-cycle. If enabled, passes flag `-Wimplicit-retain-self`
Type	Boolean
Default Value	`NO`

$CLANG_WARN_OBJC_MISSING_PROPERTY_SYNTHESIS

Description	If enabled, passes flag `-Wobjc-missing-property-synthesis`
Type	Boolean
Default Value	`NO`

$CLANG_WARN_OBJC_RECEIVER_WEAK

Description	Warn about sending messages to Objective-C pointers that are `__weak`. This aids in avoiding situations (e.g., race conditions) when the last strong reference goes away and a client is messaging a `__weak` pointer that can suddenly (and unexpectedly) become nil.
Type	Boolean
Values	• `YES` : Passes flag `-Wreceiver-is-weak` • `NO` : Passes flag `-Wno-receiver-is-weak`
Default Value	`NO`

$CLANG_WARN_OBJC_REPEATED_USE_OF_WEAK

Description	Warn about repeatedly using a weak reference without assigning the weak reference to a strong reference. This is often symptomatic of a race condition where the weak reference can become nil between accesses, resulting in unexpected behavior. Assigning to temporary strong reference ensures the object stays alive during the related accesses.
Type	Boolean
Values	• `YES` : Passes flag `-Warc-repeated-use-of-weak` • `NO` : Passes flag `-Wno-arc-repeated-use-of-weak`
Default Value	`NO`

$CLANG_WARN_OBJC_ROOT_CLASS

Description	Warn about classes that unintentionally do not subclass a root class (such as NSObject).
Type	Enumeration
Values	• `YES` : enabled by default • `NO` : Passes flag `-Wno-objc-root-class` • `YES_ERROR` : Passes flag `-Werror=objc-root-class`
Default Value	`YES_ERROR`

$CLANG_WARN_SUSPICIOUS_IMPLICIT_CONVERSION

Description	Warn about various implicit conversions that can lose information or are otherwise suspicious.
Type	Boolean
Values	• `YES` : Passes flag `-Wconversion` • `NO` : Passes flag `-Wno-conversion`
Default Value	`NO`

$CLANG_WARN_UNREACHABLE_CODE

Description	Warns about potentially unreachable code. If enabled, passes flag `-Wunreachable-code`
Type	Boolean
Default Value	`NO`

$CLANG_WARN__ARC_BRIDGE_CAST_NONARC

Description	If disabled, passes flag `-Wno-arc-bridge-casts-disallowed-in-nonarc`
Type	Boolean
Default Value	YES

$CLANG_WARN__DUPLICATE_METHOD_MATCH

Description	Warn about declaring the same method more than once within the same @interface. If enabled, passes flag `-Wduplicate-method-match`
Type	Boolean
Default Value	YES

$CLANG_WARN__EXIT_TIME_DESTRUCTORS

Description	Warn about destructors for C++ objects that are called when an application is terminating.
Type	Boolean
Values	• `YES` : Passes flag `-Wexit-time-destructors` • `NO` : Passes flag `-Wno-exit-time-destructors`
Default Value	NO

$CLANG_X86_VECTOR_INSTRUCTIONS

Description	Enables the use of extended vector instructions. Only used when targeting Intel architectures. Passes flag `-m$(value)` if not set to `default` .
Type	Enumeration
Values	• `default` : Platform default • `sse3` : SSE 3 • `ssse3` : SSE 3 (with supplemental extensions) • `sse4.1` : SSE 4.1 • `sse4.2` : SSE 4.2 • `avx` : AVX • `avx2` : AVX 2
Default Value	`$(DEFAULT_SSE_LEVEL_4_2_$(GCC_ENABLE_SSE42_EXTENSIONS))`
Example Value	`default`

$CODE_SIGN_ENTITLEMENTS

Description	Specifies the name of the application's entitlements property-list file.
Type	String
Default Value	empty string
Example Value	`Entitlements.plist`

$CODE_SIGN_IDENTITY

Description	Specifies the name of a code signing identity.
Type	String
Default Value	`-`

$CODE_SIGN_RESOURCE_RULES_PATH

Description	Identifies a property-list file containing resource-scanning instructions that override the rules for identifying bundle resources to sign.
Type	Path
Default Value	empty string
Example Value	`ResourceRules.plist`

$COLOR_DIAGNOSTICS

Type	Boolean
Default Value	NO

$COMBINE_HIDPI_IMAGES

Description	Combines image files at different resolutions into one multi-page TIFF file that is HiDPI compliant for Mac OS X 10.7 and later. Only image files in the same directory and with the same base name and extension are combined. The file names must conform to the naming convention used in HiDPI.
Type	Boolean
Default Value	`NO`

$COMPOSITE_SDK_DIRS

Description	Path to directory where Xcode creates a composited SDK of all SDKs used by a target.
Type	PathList
Default Value	`$(CACHE_ROOT)/CompositeSDKs`
Example Value	`/var/folders/m5/j4zdc7f9157659_pd5p0_n980000gn/C/com.apple.DeveloperTools/6.2-6C131e/Xcode/CompositeSDKs`

$COMPRESS_PNG_FILES

Description	Specifies whether to compress PNG files that are resources of the active target as they are copied to the application bundle. This applies only to iOS applications.
Type	Boolean
Values	• `YES` : PNG files (those with the .png suffix) are compressed as they're copied to the application bundle. • `NO` : No PNG compression takes place.
Default Value	`YES`

$CONFIGURATION

Description	Identifies the build configuration (for example, Debug or Release) the target uses to generate the product.
Type	String
Values	• `Debug` • `Release`

$CONFIGURATION_BUILD_DIR

Description	Identifies the directory under which all build-related files for the active build configuration are placed.
Type	Path
Default Value	`$(BUILD_DIR)/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)`
Example Value	`/Users/genica/MyProject/build/Debug`

$CONFIGURATION_TEMP_DIR

Description	Identifies the directory that holds temporary files for the active build configuration.
Type	Path
Default Value	`$(PROJECT_TEMP_DIR)/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug`

$COPYING_PRESERVES_HFS_DATA

Description	Passes flag `-preserve-hfs-data`
Type	Boolean
Default Value	`NO`

$COPY_PHASE_STRIP

Description	Passes flag `-strip-debug-symbols`
Type	Boolean
Default Value	`YES`

$CP

Description	Path to the `cp` tool.
Type	Path
Default Value	`/bin/cp`

$CPP_HEADERMAP_FILE

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME).hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject.hmap`

$CPP_HEADERMAP_FILE_FOR_ALL_NON_FRAMEWORK_TARGET_HEADERS

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME)-all-non-framework-target-headers.hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-all-non-framework-target-headers.hmap`

$CPP_HEADERMAP_FILE_FOR_ALL_TARGET_HEADERS

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME)-all-target-headers.hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-all-target-headers.hmap`

$CPP_HEADERMAP_FILE_FOR_GENERATED_FILES

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME)-generated-files.hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-generated-files.hmap`

$CPP_HEADERMAP_FILE_FOR_OWN_TARGET_HEADERS

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME)-own-target-headers.hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-own-target-headers.hmap`

$CPP_HEADERMAP_FILE_FOR_PROJECT_FILES

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME)-project-headers.hmap`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-project-headers.hmap`

$CPP_HEADERMAP_PRODUCT_HEADERS_VFS_FILE

Type	Path
Default Value	`$(PROJECT_TEMP_DIR)/all-product-headers.yaml`
Example Value	`/Users/genica/MyProject/build/MyProject.build/all-product-headers.yaml`

$CPP_HEADER_SYMLINKS_DIR

Type	Path
Default Value	`$(TEMP_DIR)/$(PRODUCT_NAME).hdrs`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject.hdrs`

$CREATE_INFOPLIST_SECTION_IN_BINARY

Description	If a section should be added to the Mach-O binary header for an embedded Info.plist
Type	Boolean
Default Value	`NO`

$CURRENT_ARCH

Description	Identifies the architecture on which the build is being performed.
Type	String
Example Value	`x86_64`

$CURRENT_PROJECT_VERSION

Description	This setting defines the the current version of the project. The value must be a integer or floating point number like 57 or 365.8.
Type	String
Default Value	empty string

$CURRENT_VARIANT

Description	Identifies the build variant being processed.
Type	String
Values	• When `$(CONFIGURATION)` in `$(BUILD_VARIANTS)` : `$(CONFIGURATION)` • Otherwise: `normal`

$DEAD_CODE_STRIPPING

Description	Activating this setting causes the -dead_strip flag to be passed to ld(1) via cc(1) to turn on dead code stripping. If this option is selected, -gfull (not -gused) must be used to generate debugging symbols in order to have them correctly stripped. Passes flag `-dead_strip`
Type	Boolean
Default Value	`NO`

$DEBUG_INFORMATION_FORMAT

Description	Identifies the format used to store the binary's debug information.
Type	String
Values	• `dwarf` : Use the DWARF format and place the debug information in the binary. • `dwarf-with-dsym` : Use the DWARF format and place the debug information in a dSYM file.
Default Value	`dwarf`

$DEFAULT_COMPILER

Description	This is assigned from the target platform.
Type	String
Default Value	empty string

$DEFAULT_KEXT_INSTALL_PATH

Description	This value is assigned by the target platform.
Type	String
Default Value	`$(SYSTEM_KEXT_INSTALL_PATH)`

$DEFAULT_SSE_LEVEL_3_NO

Type	String
Default Value	`default`

$DEFAULT_SSE_LEVEL_3_YES

Type	String
Default Value	`sse3`

$DEFAULT_SSE_LEVEL_3_SUPPLEMENTAL_NO

Type	String
Default Value	`$(DEFAULT_SSE_LEVEL_3_$(GCC_ENABLE_SSE3_EXTENSIONS))`
Example Value	`default`

$DEFAULT_SSE_LEVEL_3_SUPPLEMENTAL_YES

Type	String
Default Value	`ssse3`

$DEFAULT_SSE_LEVEL_4_1_NO

Type	String
Default Value	`$(DEFAULT_SSE_LEVEL_3_SUPPLEMENTAL_$(GCC_ENABLE_SUPPLEMENTAL_SSE3_INSTRUCTIONS))`

$DEFAULT_SSE_LEVEL_4_1_YES

Type	String
Default Value	`sse4.1`

$DEFAULT_SSE_LEVEL_4_2_NO

Type	String
Default Value	`$(DEFAULT_SSE_LEVEL_4_1_$(GCC_ENABLE_SSE41_EXTENSIONS))`

$DEFAULT_SSE_LEVEL_4_2_YES

Type	String
Default Value	`sse4.2`

$DEFINES_MODULE

Description	If enabled, the product will be treated as defining its own module. This enables automatic production of LLVM module map files when appropriate, and allows the product to be imported as a module.
Type	Boolean
Default Value	`NO`

$DEPLOYMENT_LOCATION

Description	Specifies whether product files are placed in the installation or the build directory.
Type	Boolean
Values	• YES: Product files are placed in `$(DSTROOT)`. • NO: Product files are placed in `$(SYMROOT)`.
Default Value	• When `$(ACTION)=install` : `YES` • Otherwise: `NO`

$DEPLOYMENT_POSTPROCESSING

Description	Specifies whether the binary receives deployment postprocessing. Deployment postprocessing involves stripping the binary, and setting its file mode, owner, and group.
Type	Boolean
Values	• `YES` : Binary receives deployment postprocessing. • `NO` : Binary does not receive deployment postprocessing.
Default Value	• `YES` : When `$(ACTION)=install`. • `NO` : Is the alternative.

$DERIVED_FILE_DIR

Description	Identifies the directory into which derived source files—such as those generated by lex and yacc—are placed.
Type	Path
Default Value	`$(TEMP_DIR)/DerivedSources`

$DERIVED_FILES_DIR

Description	See `$(DERIVED_FILE_DIR)`
Type	Path
Default Value	`$(DERIVED_FILE_DIR)`

$DERIVED_SOURCES_DIR

Description	See `$(DERIVED_FILE_DIR)`
Type	Path
Default Value	`$(DERIVED_FILE_DIR)`

$DEVELOPER_APPLICATIONS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications`

$DEVELOPER_BIN_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/usr/bin`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr/bin`

$DEVELOPER_DIR

Type	Path
Default Value	`xcode-select -p`
Example Value	`/Applications/Xcode.app/Contents/Developer`

$DEVELOPER_FRAMEWORKS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Library/Frameworks`
Example Value	`/Applications/Xcode.app/Contents/Developer/Library/Frameworks`

$DEVELOPER_FRAMEWORKS_DIR_QUOTED

Type	Path
Default Value	`$(DEVELOPER_DIR)/Library/Frameworks`
Example Value	`/Applications/Xcode.app/Contents/Developer/Library/Frameworks`

$DEVELOPER_LIBRARY_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Library`
Example Value	`/Applications/Xcode.app/Contents/Developer/Library`

$DEVELOPER_SDK_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Platforms/MacOSX.platform/Developer/SDKs`
Example Value	`/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs`

$DEVELOPER_TOOLS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Tools`
Example Value	`/Applications/Xcode.app/Contents/Developer/Tools`

$DEVELOPER_USR_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/usr`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr`

$DSTROOT

Description	Identifies the directory into which the product is placed. In this directory, the product is laid out exactly as it would be installed in a user's filesystem.
Type	Path
Default Value	`/tmp/$(PROJECT_NAME).dst`
Example Value	/tmp/MyProject.dst

$DT_TOOLCHAIN_DIR

Description	Path to default toolchain.
Type	Path
Default Value	`/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain`

$DYLIB_COMPATIBILITY_VERSION

Description	Specifies the compatibility version of a dynamic library product. See Dynamic Library Design Guidelines in Dynamic Library Programming Topics for details on assigning version numbers of dynamic libraries.
Type	String
Default Value	empty string

$DYLIB_CURRENT_VERSION

Description	Specifies the current version of a dynamic library product. See "Dynamic Library Design Guidelines" in Dynamic Library Programming Topics for details on assigning version numbers of dynamic libraries.
Type	String
Default Value	empty string

$DYLIB_INSTALL_NAME_BASE

Description	Sets the base value for the internal "install path" `$(LC_ID_DYLIB)` in a dynamic library. This will be combined with the `$(EXECUTABLE_PATH)` to form the full install path. Setting `$(LD_DYLIB_INSTALL_NAME)` directly will override this setting. This setting defaults to the target's `$(INSTALL_PATH)`. It is ignored when building any product other than a dynamic library.
Type	StringList
Default Value	empty string

$EFFECTIVE_PLATFORM_NAME

Description	Name of target platform.
Type	String
Values	• When current platform identifier is `com.apple.platform.macosx` : empty string • Otherwise: `-$(PLATFORM_NAME)`

$EMBEDDED_CONTENT_CONTAINS_SWIFT

Description	Enable this setting to indicate that content embedded in a target's product contains Swift code, so that the standard Swift libraries can be included in the product. See also 1 and 2.
Type	Boolean
Default Value	`NO`

$EMBEDDED_PROFILE_NAME

Description	Name of the embedded provisioning profile file.
Type	String
Default Value	empty string
Example Value	`embedded.provisionprofile`

$ENABLE_APPLE_KEXT_CODE_GENERATION

Description	If enabled, passes flag `-fapple-kext`
Type	Boolean
Default Value	`NO`

$ENABLE_HEADER_DEPENDENCIES

Description	Specifies whether data gathered from header-file scans is used in the build process.
Type	Boolean
Values	• `YES` : The build uses data gathered from header-file scans. • `NO` : The build does not use data gathered from header-file scans.
Default Value	`YES`

$ENABLE_NS_ASSERTIONS

Description	Controls whether assertion logic provided by NSAssert is included in the preprocessed source code or is elided during preprocessing. Disabling assertions can improve code performance. If disabled, passes flag `-DNS_BLOCK_ASSERTIONS=1`
Type	Boolean
Default Value	`YES`

$ENABLE_STRICT_OBJC_MSGSEND

Description	Controls whether objc_msgSend calls must be cast to the appropriate function pointer type before being called. If enabled, passes flag `-DOBJC_OLD_DISPATCH_PROTOTYPES=0`
Type	Boolean
Default Value	`NO`

$EXCLUDED_INSTALLSRC_SUBDIRECTORY_PATTERNS

Description	Space-separated list of subdirectories to exclude.
Type	StringList
Default Value	`.DS_Store .svn .git .hg CVS`

$EXCLUDED_RECURSIVE_SEARCH_PATH_SUBDIRECTORIES

Description	Space-separated list of subdirectories to exclude.
Type	StringList
Default Value	`*.nib *.lproj *.framework *.gch *.xcode* (*) .DS_Store CVS .svn .git .hg`

$EXECUTABLE_EXTENSION

Description	This is the extension used for the executable product generated by the target, which has a default value based on the product type.
Type	String
Default Value	empty string

$EXECUTABLE_PREFIX

Description	This is the prefix used for the executable product generated by the target, which has a default value based on the product type.
Type	String
Default Value	empty string

$EXECUTABLE_SUFFIX

Description	This is the suffix used for the executable product generated by the target, which has a default value based on the product type.
Type	String
Default Value	empty string

$EXECUTABLE_VARIANT_SUFFIX

Description	This is the suffix (based on `$(CURRENT_VARIANT)`) used for the executable product generated by the target, which has a default value based on the product type.
Type	String
Default Value	empty string

$EXPORTED_SYMBOLS_FILE

Description	This is a project-relative path to a file that lists the symbols to export. Passes flag `-exported_symbols_list`
Type	Path
Default Value	empty string

$FILE_LIST

Type	Path
Default Value	`$(OBJECT_FILE_DIR)/LinkFileList`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/Objects/LinkFileList`

$FRAMEWORK_SEARCH_PATHS

Description	Space-separated list of directory paths. Specifies directories in which the compiler searches for frameworks to find included header files. You may specify a recursive path by appending ** to the path. When this build setting is defined, `$(SDKROOT)` is added to the end of the path list that is passed to the compiler.
Type	PathList
Default Value	empty string
Example Value	• `/Users/genica/TestFrameworks/**` • `/Volumes/Auryon/TeamFrameworks/**`

$FRAMEWORK_VERSION

Description	Version identifier of a framework.
Type	String
Default Value	`A`

$GCC3_VERSION

Description	GCC version.
Type	String
Default Value	`3.3`

$GCC_CHAR_IS_UNSIGNED_CHAR

Description	Enabling this setting causes 'char' to be unsigned by default, disabling it causes 'char' to be signed by default. If enabled, passes flag `-funsigned-char`
Type	Boolean
Default Value	`NO`

$GCC_CW_ASM_SYNTAX

Description	Enable the CodeWarrior/Microsoft syntax for inline assembly code in addition to the standard GCC syntax.
Type	Boolean
Values	• `YES` : Use CodeWarrior syntax for inline assembly code. Passes `-fasm-blocks` • `NO` : Do not use CodeWarrior syntax for inline assembly code.
Default Value	`YES`

$GCC_C_LANGUAGE_STANDARD

Description	Choose a standard or non-standard C language dialect.
Type	Enumeration
Values	• `ansi` : `-ansi` ANSI C: Accept ISO C90 and ISO C++, turning off GNU extensions that are incompatible. Incompatible GNU extensions include the `asm` , `inline` , and `typeof` keywords (but not the equivalent `__asm__` , `__inline__` , and `__typeof__` forms), and the `//` syntax for comments. This setting also enables trigraphs. • `c89` : `-std=c89` C89: Accept ISO C90 (1990), but not GNU extensions. • `gnu89` : `-std=gnu89` GNU89: Accept ISO C90 and GNU extensions. • `c99` : `-std=c99` C99: Accept ISO C99 (1999), but not GNU extensions. • `gnu99` : `-std=gnu99` GNU99: Accept ISO C99 and GNU extensions. • `c11` : `-std=c11` C11: Accept ISO C11 (2011), but not GNU extensions. • `gnu11` : `-std=gnu11` GNU11: Accept ISO C11 and GNU extensions. • `compiler-default` : Compiler Default: Tells the compiler to use its default C language dialect. This is normally the best choice unless you have specific needs. (Currently equivalent to GNU99.)
Default Value	`compiler-default`

$GCC_DEBUG_INFORMATION_FORMAT

Description	Debug information file format.
Type	Enumeration
Values	• `dwarf` : Passes `-g` • `dwarf-with-dsym` : Passes `-g` • Otherwise:
Default Value	`$(DEBUG_INFORMATION_FORMAT)`

$GCC_DYNAMIC_NO_PIC

Description	Faster function calls for applications. Not appropriate for shared libraries (which need to be position-independent).
Type	Boolean
Values	• `YES` : Generated code is nonrelocatable when the prerequisite is met. Passes flag `-mdynamic-no-pic` • `NO` : Generated code is relocatable.
Default Value	`NO`

$GCC_ENABLE_ASM_KEYWORD

Description	Controls whether 'asm', 'inline', and 'typeof' are treated as keywords or whether they can be used as identifiers. If disabled, passes flag `-fno-asm`
Type	Boolean
Default Value	`YES`

$GCC_ENABLE_BUILTIN_FUNCTIONS

Description	Controls whether built-in functions that do not begin with `__builtin_` as prefix are recognized. GCC normally generates special code to handle certain built-in functions more efficiently; for instance, calls to "alloca" may become single instructions that adjust the stack directly, and calls to "memcpy" may become inline copy loops. The resulting code is often both smaller and faster, but since the function calls no longer appear as such, you cannot set a breakpoint on those calls, nor can you change the behavior of the functions by linking with a different library. In addition, when a function is recognized as a built-in function, GCC may use information about that function to warn about problems with calls to that function, or to generate more efficient code, even if the resulting code still contains calls to that function. For example, warnings are given with `-Wformat` for bad calls to "printf", when "printf" is built in, and "strlen" is known not to modify global memory. If disabled, passes flag `-fno-builtin`
Type	Boolean
Default Value	`YES`

$GCC_ENABLE_CPP_EXCEPTIONS

Description	Enable C++ exception handling. Generates extra code needed to propagate exceptions. For some targets, this implies GCC will generate frame unwind information for all functions, which can produce significant data size overhead, although it does not affect execution. If you do not specify this option, GCC will enable it by default for languages like C++ which normally require exception handling, and disable it for languages like C that do not normally require it. However, you may need to enable this option when compiling C code that needs to interoperate properly with exception handlers written in C++.
Type	Boolean
Values	• `YES` : Compiler generates code necessary for exception propagation. • `NO` : Compiler does not generate code necessary for exception propagation. Passes flag `-fno-exceptions`
Default Value	`YES`

$GCC_ENABLE_CPP_RTTI

Description	Enable generation of information about every class with virtual functions for use by the C++ runtime type identification features (dynamic_cast and typeid). If you don't use those parts of the language, you can save some space by using this flag. Note that exception handling uses the same information, but it will generate it as needed.
Type	Boolean
Values	• `YES` : Binary includes information about virtual classes. • `NO` : Binary might not include information about virtual classes. Passes flag `-fno-rtti`
Default Value	`YES`

$GCC_ENABLE_EXCEPTIONS

Description	Enable exception handling. Generates extra code needed to propagate exceptions. For some targets, this implies GCC will generate frame unwind information for all functions, which can produce significant data size overhead, although it does not affect execution. If you do not specify this option, GCC will enable it by default for languages like C++ and Objective-C which normally require exception handling, and disable it for languages like C that do not normally require it. However, you may need to enable this option when compiling C code that needs to interoperate properly with exception handlers written in other languages. You may also wish to disable this option if you are compiling older programs that don't use exception handling. Passes flag `-fexceptions`
Type	Boolean
Default Value	`NO`

$GCC_ENABLE_FLOATING_POINT_LIBRARY_CALLS

Description	Generate output containing library calls for floating point. If enabled, passes flag `-msoft-float`
Type	Boolean
Default Value	`NO`

$GCC_ENABLE_KERNEL_DEVELOPMENT

Description	Activating this setting enables kernel development mode. This mode sets `-static`, `-fno-common`, `-fno-cxa-atexit`, `-fno-exceptions`, `-fno-non-call-exceptions`, `-fapple-kext`, `-fno-weak`, and `-fno-rtti` where applicable. This mode also sets `-mno-altivec`, `-msoft-float`, `-fno-builtin`, and `-mlong-branch` for PowerPC targets. Passes flag `-mkernel`
Type	Boolean
Default Value	`NO`

$GCC_ENABLE_OBJC_EXCEPTIONS

Description	This setting enables @try/@catch/@throw syntax for handling exceptions in Objective-C code. Specifies whether the compiler recognizes @try, @catch, and @throw directives.
Type	Boolean
Values	• `YES` : Recognize the Objective-C exception-handling directives. • `NO` : Do not allow the Objective-C exception-handling directives in source code. Passes flag `-fno-objc-exceptions`
Default Value	`YES`

$GCC_ENABLE_OBJC_GC

Description	Specifies the level of garbage-collection support for the generated code.
Type	Enumeration
Values	• `unsupported` : The application cannot load code that requires garbage collection. The loadable bundle cannot be loaded by an application that requires garbage collection. • `supported` : The application can load code that supports or requires garbage collection. The loadable bundle can be loaded by an application with any level of garbage-collection support. Passes flag `-fobjc-gc` • `required` : The application can load only code that supports garbage collection. The loadable bundle can be loaded only by an application that supports garbage collection. Passes flag `-fobjc-gc-only`
Default Value	`unsupported`

$GCC_ENABLE_PASCAL_STRINGS

Description	Recognize and construct Pascal-style string literals. Its use in new code is discouraged. Pascal string literals take the form `\\pstring`. The special escape sequence `\\p` denotes the Pascal length byte for the string, and will be replaced at compile time with the number of characters that follow. The `\p` may only appear at the beginning of a string literal, and may not appear in wide string literals or as an integral constant. If enabled, passes flag `-fpascal-strings`
Type	Boolean
Default Value	`YES`

$GCC_ENABLE_SSE3_EXTENSIONS

Description	Specifies whether the binary uses the built-in functions that provide access to the SSE3 extensions to the IA-32 architecture. Old build setting. Kept around for old project compatibility. Forwards to `$(CLANG_X86_VECTOR_INSTRUCTIONS)`
Type	Boolean
Values	• `YES` : Binary uses SSE3 functions. • `NO` : Binary does not use SSE3 functions.
Default Value	`NO`

$GCC_ENABLE_SSE41_EXTENSIONS

Description	Specifies whether the binary uses the built-in functions that provide access to the SSE4.1 extensions to the IA-32 architecture. Old build setting. Kept around for old project compatibility. Forwards to `$(CLANG_X86_VECTOR_INSTRUCTIONS)`
Type	Boolean
Values	• `YES` : Binary uses SSE4.1 functions. • `NO` : Binary does not use SSE4.1 functions.
Default Value	`NO`

$GCC_ENABLE_SSE42_EXTENSIONS

Description	Specifies whether the binary uses the built-in functions that provide access to the SSE4.2 extensions to the IA-32 architecture. Old build setting. Kept around for old project compatibility. Forwards to `$(CLANG_X86_VECTOR_INSTRUCTIONS)`
Type	Boolean
Values	• `YES` : Binary uses SSE4.2 functions. • `NO` : Binary does not use SSE4.2 functions.
Default Value	`NO`

$GCC_ENABLE_SUPPLEMENTAL_SSE3_INSTRUCTIONS

Description	Old build setting. Kept around for old project compatibility. Forwards to `$(CLANG_X86_VECTOR_INSTRUCTIONS)` .
Type	Boolean
Default Value	`NO`

$GCC_ENABLE_TRIGRAPHS

Description	Controls whether or not trigraphs are permitted in the source code.
Type	Boolean
Values	• `YES` : Passes flag `-trigraphs` • `NO` : Passes flag `-Wno-trigraghs`
Default Value	`NO`

$GCC_FAST_MATH

Description	Enables some floating point optimizations that are not IEEE754-compliant, but which usually work. Programs which require strict IEEE compliance may not work with this option. If enabled, passes flag `-ffast-math`
Type	Boolean
Default Value	`NO`

$GCC_GENERATE_DEBUGGING_SYMBOLS

Description	Enables or disables generation of debug symbols. When debug symbols are enabled, the level of detail can be controlled by the build 'Level of Debug Symbols' setting.
Type	Boolean
Values	• `YES` : Binary includes debugging symbols. • `NO` : Binary does not include debugging symbols.
Default Value	`YES`

$GCC_GENERATE_TEST_COVERAGE_FILES

Description	Activating this setting causes a 'notes' file to be produced that the gcov code-coverage utility can use to show program coverage. If enabled, passes flag `-ftest-coverage`
Type	Boolean
Default Value	`NO`

$GCC_INCREASE_PRECOMPILED_HEADER_SHARING

Description	Enabling this option will enable increased sharing of precompiled headers among targets which share the same prefix header and precompiled header directory. Xcode distinguishes among precompiled header files by generating a hash value based on the command-line options to the compiler used to create the PCH. Enabling this option will exclude certain compiler options from that hash. Presently this option will exclude search path options (`-I` , `-iquote` , `-isystem` , `-F` , `-L`) from the hash. Enabling increased sharing of PCH files carries some risk: If two targets use the same prefix header but have different include paths which cause the prefix header to include different files when they are precompiled, then subtle problems may result because one target will use a PCH which was built using files included by the other target. In this case, this option must be turned off in order to enforce correctness.
Type	Boolean
Default Value	`NO`

$GCC_INLINES_ARE_PRIVATE_EXTERN

Description	When enabled, out-of-line copies of inline methods are declared `private extern` . If enabled, passes flag `-fvisibility-inlines-hidden`
Type	Boolean
Default Value	`NO`

$GCC_INPUT_FILETYPE

Description	Specifies whether to compile each source file according to its file type, or whether to treat all source files in the target as if they are of a specific language.
Type	Enumeration
Values	• `automatic` : According to File Type • `sourcecode.c.c` : C • `sourcecode.c.objc` : Objective-C • `sourcecode.cpp.cpp` : C++ • `sourcecode.cpp.objcpp` : Objective-C++
Default Value	`automatic`

$GCC_INSTRUMENT_PROGRAM_FLOW_ARCS

Description	Activating this setting indicates that code should be added so program flow arcs are instrumented. If enabled, passes flag `-fprofile-arcs`
Type	Boolean
Default Value	`NO`

$GCC_LINK_WITH_DYNAMIC_LIBRARIES

Description	Enabling this option allows linking with the shared libraries. This is the default for most product types. If disabled, passes flag `-static`
Type	Boolean
Default Value	`YES`

$GCC_MACOSX_VERSION_MIN

Description	Sets the minimum deployment version for OS X. Passes flag `-mmacosx-version-min=$(value)`
Type	String
Default Value	`$(MACOSX_DEPLOYMENT_TARGET)`

$GCC_NO_COMMON_BLOCKS

Description	In C, allocate even uninitialized global variables in the data section of the object file, rather than generating them as common blocks. This has the effect that if the same variable is declared (without `extern`) in two different compilations, you will get an error when you link them. The only reason this might be useful is if you wish to verify that the program will work on other systems which always work this way. If enabled, passes flag `-fno-common`
Type	Boolean
Default Value	`NO`

$GCC_OBJC_ABI_VERSION

Description	Used by iPhoneSimulator to specify the ObjC runtime for architecture. Passes flag `-fobjc-abi-version=$(value)`
Type	Enumeration
Values	• `1` : Objective-C version 1 • `2` : Objective-C version 2
Default Value	`$(OBJC_ABI_VERSION)`

$GCC_OBJC_LEGACY_DISPATCH

Description	Used by iPhoneSimulator to specify the ObjC runtime for architecture. Passes flag `-fobjc-legacy-dispatch`
Type	Boolean
Default Value	`NO`

$GCC_OPERATION

Description	Old build setting.
Type	Enumeration
Values	• `compile` • `generate-preprocessed` • `generate-assembler` • `precompile` • `separate-symbols`
Default Value	`compile`

$GCC_OPTIMIZATION_LEVEL

Description	Specifies the degree to which the generated code is optimized for speed and binary size. Passes flag `-O$(value)`
Type	Enumeration
Values	• `0` : None: Do not optimize. * With this setting, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. Statements are independent: if you stop the program with a breakpoint between statements, you can then assign a new value to any variable or change the program counter to any other statement in the function and get exactly the results you would expect from the source code. • `1` : Fast: Optimizing compilation takes somewhat more time, and a lot more memory for a large function. * With this setting, the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time. In Apple's compiler, strict aliasing, block reordering, and inter-block scheduling are disabled by default when optimizing. • `2` : Faster: The compiler performs nearly all supported optimizations that do not involve a space-speed tradeoff. * With this setting, the compiler does not perform loop unrolling or function inlining, or register renaming. As compared to the 'Fast' setting, this setting increases both compilation time and the performance of the generated code. • `3` : Fastest: Turns on all optimizations specified by the 'Faster' setting and also turns on function inlining and register renaming options. This setting may result in a larger binary. • `s` : Fastest, Smallest: Optimize for size. This setting enables all 'Faster' optimizations that do not typically increase code size. It also performs further optimizations designed to reduce code size. • `fast` : Fastest, Aggressive Optimizations: This setting enables 'Fastest' but also enables aggressive optimizations that may break strict standards compliance but should work well on well-behaved code.
Default Value	`s`

$GCC_PFE_FILE_C_DIALECTS

Description	Space-separated list of supported C dialect files.
Type	StringList
Default Value	`c objective-c c++ objective-c++`

$GCC_PRECOMPILE_PREFIX_HEADER

Description	Generates a precompiled header for the prefix header, which should reduce overall build times.
Type	Boolean
Values	• `YES` : Target generates a prefix header when the prerequisite is met. • `NO` : Target does not generate a prefix header.
Default Value	`$(PRECOMPILE_PREFIX_HEADER)`

$GCC_PREFIX_HEADER

Description	Implicitly include the named header. The path given should either be a project relative path or an absolute path.
Type	String
Default Value	`$(PREFIX_HEADER)`
Example Value	`MyProduct_Prefix.pch`

$GCC_PREPROCESSOR_DEFINITIONS

Description	Space-separated list of preprocessor macros of the form "foo" or "foo=bar". Each element is passed with flag `-D`
Type	StringList
Default Value	• When `$(CONFIGURATION)=Debug` : `DEBUG=1` • Otherwise: empty string

$GCC_PREPROCESSOR_DEFINITIONS_NOT_USED_IN_PRECOMPS

Description	Space-separated list of preprocessor macros of the form "foo" or "foo=bar". These macros are not used when precompiling a prefix header file.
Type	StringList
Default Value	empty string

$GCC_PRODUCT_TYPE_PREPROCESSOR_DEFINITIONS

Description	See `$(GCC_PREPROCESSOR_DEFINITIONS)` .
Type	StringList
Default Value	empty string

$GCC_REUSE_STRINGS

Description	Reuse string literals. If disabled, passes flag `-fwritable-strings`
Type	Boolean
Default Value	`YES`

$GCC_SHORT_ENUMS

Description	Make enums only as large as needed for the range of possible values. If enabled, passes flag `-fshort-enums`
Type	Boolean
Default Value	`NO`

$GCC_STRICT_ALIASING

Description	Optimize code by making more aggressive assumptions about whether pointers can point to the same objects as other pointers. Programs which use pointers a lot may benefit from this, but programs that don't strictly follow the ISO C rules about the type with which an object may be accessed may behave unexpectedly.
Type	Boolean
Values	• `YES` : Passes flag `-fstrict-aliasing` • `NO` : Passes flag `-fno-strict-aliasing`
Default Value	`YES`

$GCC_SYMBOLS_PRIVATE_EXTERN

Description	When enabled, all symbols are declared `private extern` unless explicitly marked to be exported using `__attribute__((visibility("default")))` in code. If not enabled, all symbols are exported unless explicitly marked as `private extern`. For more information, see Symbol Visibility.
Type	Boolean
Values	• `YES` : Symbols that do not specify public visibility are not exported. Passes flag: `-fvisibility=hidden` • `NO` : Symbols that do not specify private visibility are exported.
Default Value	`NO`

$GCC_THREADSAFE_STATICS

Description	Emits extra code to use the routines specified in the C++ ABI for thread-safe initialization of local statics. You can disable this option to reduce code size slightly in code that doesn't need to be thread-safe.
Type	Boolean
Values	• `YES` : Binary uses the IA-32 ABI thread-safe initialization functions. • `NO` : Binary does not use the IA-32 ABI thread-safe initialization functions. Passes flag `-fno-threadsafe-statics`
Default Value	`YES`

$GCC_TREAT_IMPLICIT_FUNCTION_DECLARATIONS_AS_ERRORS

Description	Causes warnings about missing function prototypes to be treated as errors. Only applies to C and Objective-C. If enabled, passes flag `-Werror-implicit-function-declaration`
Type	Boolean
Default Value	`NO`

$GCC_TREAT_INCOMPATIBLE_POINTER_TYPE_WARNINGS_AS_ERRORS

Description	Enabling this option causes warnings about incompatible pointer types to be treated as errors. If enabled, passes flag `-Werror=incompatible-pointer-types`
Type	Boolean
Default Value	`NO`

$GCC_TREAT_WARNINGS_AS_ERRORS

Description	Enabling this option causes all warnings to be treated as errors. If enabled, passes flag `"-Werror`
Type	Boolean
Default Value	`NO`

$GCC_UNROLL_LOOPS

Description	Unrolls loops. Unrolling makes the code larger, but may make it faster by reducing the number of branches executed.
Type	Boolean
Values	• `YES` : Compiler generates code with unrolled loops. Passes flag `-funroll-loops` • `NO` : Compiler does not unroll loops.
Default Value	`NO`

$GCC_USE_GCC3_PFE_SUPPORT

Description	Used by GCC 3.1 and later only.
Type	Boolean
Default Value	`YES`

$GCC_USE_STANDARD_INCLUDE_SEARCHING

Description	If disabled, passes flag `-nostdinc` .
Type	Boolean
Default Value	`YES`

$GCC_VERSION

Description	Identifies the GCC version to be used to compile the target's source files. When the target's "System C rule" is set to GCC System Version (instead of a specific version number), this build setting is not available in Run Script build phases.
Type	String
Default Value	empty string

$GCC_WARN_64_TO_32_BIT_CONVERSION

Description	Warn if a value is implicitly converted from a 64 bit type to a 32 bit type. This is a subset of the warnings provided by `-Wconversion` .
Type	Boolean
Values	• `YES` : Passes flag `-Wshorten-64-to-32` • `NO` : Passes flag `-Wno-shorten-64-to-32`
Default Value	`NO`

$GCC_WARN_ABOUT_DEPRECATED_FUNCTIONS

Description	Warn about the use of deprecated functions, variables, and types (as indicated by the 'deprecated' attribute).
Type	Boolean
Values	• `YES` : Passes flag `-Wdeprecated-declarations` • `NO` : Passes flag `-Wno-deprecated-declarations`
Default Value	`YES`

$GCC_WARN_ABOUT_INVALID_OFFSETOF_MACRO

Description	Unchecking this setting will suppress warnings from applying the offsetof macro to a non-POD type. According to the 1998 ISO C++ standard, applying offsetof to a non-POD type is undefined. In existing C++ implementations, however, offsetof typically gives meaningful results even when applied to certain kinds of non-POD types. (Such as a simple struct that fails to be a POD type only by virtue of having a constructor.) This flag is for users who are aware that they are writing non-portable code and who have deliberately chosen to ignore the warning about it. The restrictions on offsetof may be relaxed in a future version of the C++ standard.
Type	Boolean
Values	• `YES` : Passes flag `-Winvalid-offsetof` • `NO` : Passes flag `-Wno-invalid-offsetof`
Default Value	`YES`

$GCC_WARN_ABOUT_MISSING_FIELD_INITIALIZERS

Description	Warn if a structure's initializer has some fields missing. For example, the following code would cause such a warning, because "x.h" is implicitly zero:
	`struct s { int f, g, h; };`
	`struct s x = { 3, 4 };`
	This option does not warn about designated initializers, so the following modification would not trigger a warning:
	`struct s { int f, g, h; };`
	`struct s x = { .f = 3, .g = 4 };`

Type	Boolean
Values	• `YES` : Passes flag `-Wmissing-field-initializers` • `NO` : Passes flag `-Wno-missing-field-initializers`
Default Value	`NO`

$GCC_WARN_ABOUT_MISSING_NEWLINE

Description	Warn when a source file does not end with a newline.
Type	Boolean
Values	• `YES` : Passes flag `-Wnewline-eof` • `NO` : Passes flag `-Wno-newline-eof`
Default Value	`NO`

$GCC_WARN_ABOUT_MISSING_PROTOTYPES

Description	Causes warnings to be emitted about missing prototypes.
Type	Boolean
Values	• `YES` : Passes flag `-Wmissing-prototypes` • `NO` : Passes flag `-Wno-missing-prototypes`
Default Value	`NO`

$GCC_WARN_ABOUT_POINTER_SIGNEDNESS

Description	Warn when pointers passed via arguments or assigned to a variable differ in sign.
Type	Boolean
Values	• `YES` : Passes flag `-Wpointer-sign` • `NO` : Passes flag `-Wno-pointer-sign`
Default Value	`YES`

$GCC_WARN_ABOUT_RETURN_TYPE

Description	Causes warnings to be emitted when a function with a defined return type (not void) contains a return statement without a return-value. Also emits a warning when a function is defined without specifying a return type.
Type	Enumeration
Values	• `YES` : • `YES_ERROR` : Passes flag `-Werror=return-type` • `NO` : Passes flag `-Wno-return-type`
Default Value	`YES_ERROR`

$GCC_WARN_ALLOW_INCOMPLETE_PROTOCOL

Description	Warn if methods required by a protocol are not implemented in the class adopting it. Only applies to Objective-C.
Type	Boolean
Values	• `YES` : Passes flag `-Wprotocol` • `NO` : Passes flag `-Wno-protocol`
Default Value	`YES`

$GCC_WARN_CHECK_SWITCH_STATEMENTS

Description	Warn whenever a switch statement has an index of enumeral type and lacks a case for one or more of the named codes of that enumeration. The presence of a default label prevents this warning. Case labels outside the enumeration range also provoke warnings when this option is used.
Type	Boolean
Values	• `YES` : Passes flag `-Wswitch` • `NO` : Passes flag `-Wno-switch`
Default Value	`YES`

$GCC_WARN_FOUR_CHARACTER_CONSTANTS

Description	Warn about four-char literals (e.g., MacOS-style OSTypes: 'APPL').
Type	Boolean
Values	• `YES` : Passes flag `-Wfour-char-constants` • `NO` : Passes flag `-Wno-four-char-constants`
Default Value	`NO`

$GCC_WARN_HIDDEN_VIRTUAL_FUNCTIONS

Description	Warn when a function declaration hides virtual functions from a base class. For example, in:
	`struct A {`
	`  virtual void f();`
	`};`
	`struct B: public A {`
	`  void f(int);`
	`};`
	the A class version of f() is hidden in B, and code like this:
	`B* b;`
	`b->f();`
	will fail to compile. This setting only applies to C++ and Objective-C++ sources.

Type	Boolean
Values	• `YES` : Passes flag `-Woverloaded-virtual` • `NO` : Passes flag `-Wno-overloaded-virtual`
Default Value	`NO`

$GCC_WARN_INHIBIT_ALL_WARNINGS

Description	Inhibit all warning messages. If enabled, passes flag `-w`
Type	Boolean
Default Value	`NO`

$GCC_WARN_INITIALIZER_NOT_FULLY_BRACKETED

Description	Warn if an aggregate or union initializer is not fully bracketed.
	
	Example, Here initializer for `a` is not fully bracketed, but that for `b` is fully bracketed.
	
	`int a[2][2] = { 0, 1, 2, 3 };`
	`int b[2][2] = { { 0, 1 }, { 2, 3 } };`

Type	Boolean
Values	• `YES` : Passes flag `-Wmissing-braces` • `NO` : Passes flag `-Wno-missing-braces`
Default Value	`NO`

$GCC_WARN_MISSING_PARENTHESES

Description	Warn if parentheses are omitted in certain contexts, such as when there is an assignment in a context where a truth value is expected, or when operators are nested whose precedence people often get confused about.
	
	Also warn about constructions where there may be confusion to which if statement an else branch belongs. Here is an example of such a case:
	
	`{`
	`  if (a)`
	`    if (b)`
	`      foo ();`
	`  else`
	`    bar ();`
	`}`
	
	In C, every else branch belongs to the innermost possible if statement, which in this example is if (b) . This is often not what the programmer expected, as illustrated in the above example by indentation the programmer chose. When there is the potential for this confusion, GCC will issue a warning when this flag is specified. To eliminate the warning, add explicit braces around the innermost if statement so there is no way the else could belong to the enclosing if . The resulting code would look like this:
	
	`{`
	`  if (a)`
	`    {`
	`      if (b)`
	`        foo ();`
	`      else`
	`        bar ();`
	`    }`
	`}`

Type	Boolean
Values	• `YES` : Passes flag `-Wparentheses` • `NO` : Passes flag `-Wno-parentheses`
Default Value	`YES`

$GCC_WARN_MULTIPLE_DEFINITION_TYPES_FOR_SELECTOR

Description	Warn if multiple methods of different types for the same selector are found during compilation. The check is performed on the list of methods in the final stage of compilation. Additionally, a check is performed for each selector appearing in a `@selector(...)` expression, and a corresponding method for that selector has been found during compilation. Because these checks scan the method table only at the end of compilation, these warnings are not produced if the final stage of compilation is not reached, for example because an error is found during compilation, or because the `-fsyntax-only` option is being used.
Type	Boolean
Values	• `YES` : Passes flag `-Wselector` • `NO` : Passes flag `-Wno-selector`
Default Value	`NO`

$GCC_WARN_NON_VIRTUAL_DESTRUCTOR

Description	Warn when a class declares an nonvirtual destructor that should probably be virtual, because it looks like the class will be used polymorphically.
Type	Boolean
Values	• `YES` : Passes flag `-Wnon-virtual-dtor` • `NO` : Passes flag `-Wno-non-virtual-dtor`
Default Value	`NO`

$GCC_WARN_PEDANTIC

Description	Issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any -std option used. If enabled, passes flag `-pedantic`
Type	Boolean
Default Value	`NO`

$GCC_WARN_SHADOW

Description	Warn whenever a local variable shadows another local variable, parameter or global variable or whenever a built-in function is shadowed.
Type	Boolean
Values	• `YES` : Passes flag `-Wshadow` • `NO` : Passes flag `-Wno-shadow`
Default Value	`NO`

$GCC_WARN_SIGN_COMPARE

Description	Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned. If enabled, passes flag `-Wsign-compare`
Type	Boolean
Default Value	`NO`

$GCC_WARN_STRICT_SELECTOR_MATCH

Description	Warn if multiple methods with differing argument and/or return types are found for a given selector when attempting to send a message using this selector to a receiver of type "id" or "Class". When this setting is disabled, the compiler will omit such warnings if any differences found are confined to types which share the same size and alignment.
Type	Boolean
Values	• `YES` : Passes flag `-Wselector` • `NO` : Passes flag `-Wno-selector`
Default Value	`NO`

$GCC_WARN_TYPECHECK_CALLS_TO_PRINTF

Description	Check calls to printf and scanf, etc., to make sure that the arguments supplied have types appropriate to the format string specified, and that the conversions specified in the format string make sense. Enabled by default. If disabled, passes flag `-Wno-format`
Type	Boolean
Default Value	`YES`

$GCC_WARN_UNDECLARED_SELECTOR

Description	Warn if a `@selector(...)` expression referring to an undeclared selector is found. A selector is considered undeclared if no method with that name has been declared before the `@selector(...)` expression, either explicitly in an `@interface` or `@protocol` declaration, or implicitly in an `@implementation` section. This option always performs its checks as soon as a `@selector(...)` expression is found, while `-Wselector` only performs its checks in the final stage of compilation. This also enforces the coding style convention that methods and selectors must be declared before being used.
Type	Boolean
Values	• `YES` : Passes flag `-Wundeclared-selector` • `NO` : Passes flag `-Wno-undeclared-selector`
Default Value	`NO`

$GCC_WARN_UNINITIALIZED_AUTOS

Description	Warn if a variable might be clobbered by a setjmp call or if an automatic variable is used without prior initialization. Note that the compiler may not detect all cases where an automatic variable is initialized or all usage patterns that may lead to use prior to initialization. You can toggle between the normal uninitialized value checking or the more aggressive (conservative) checking which finds more issues but the checking is much stricter.
Type	Enumeration
Values	• `YES` : Passes flag `-Wuninitialized` • `YES_AGGRESSIVE` : Passes flag `-Wconditional-uninitialized` • `NO` : Passes flag `-Wno-uninitialized`
Default Value	`NO`

$GCC_WARN_UNKNOWN_PRAGMAS

Description	Warn when a `#pragma` directive is encountered which is not understood by GCC. If this command line option is used, warnings will even be issued for unknown pragmas in system header files. This is not the case if the warnings were only enabled by the `-Wall` command line option.
Type	Boolean
Values	• `YES` : Passes flag `-Wunknown-pragmas` • `NO` : Passes flag `-Wno-unknown-pragmas`
Default Value	`NO`

$GCC_WARN_UNUSED_FUNCTION

Description	Warn whenever a static function is declared but not defined or a non-inline static function is unused.
Type	Boolean
Values	• `YES` : Passes flag `-Wunused-function` • `NO` : Passes flag `-Wno-unused-function`
Default Value	`NO`

$GCC_WARN_UNUSED_LABEL

Description	Warn whenever a label is declared but not used.
Type	Boolean
Values	• `YES` : Passes flag `-Wunused-label` • `NO` : Passes flag `-Wno-unused-label`
Default Value	`NO`

$GCC_WARN_UNUSED_PARAMETER

Description	Warn whenever a function parameter is unused aside from its declaration.
Type	Boolean
Values	• `YES` : Passes flag `-Wunused-parameter` • `NO` : Passes flag `-Wno-unused-parameter`
Default Value	`NO`

$GCC_WARN_UNUSED_VALUE

Description	Warn whenever a statement computes a result that is explicitly not used.
Type	Boolean
Values	• `YES` : Passes flag `-Wunused-value` • `NO` : Passes flag `-Wno-unused-value`
Default Value	`YES`

$GCC_WARN_UNUSED_VARIABLE

Description	Warn whenever a local variable or non-constant static variable is unused aside from its declaration.
Type	Boolean
Values	• `YES` : Passes flag `-Wunused-variable` • `NO` : Passes flag `-Wno-unused-variable`
Default Value	`NO`

$GENERATE_MASTER_OBJECT_FILE

Description	Activating this setting will cause the object files built by a target to be prelinked using `ld -r` into a single object file, and that object file will then be linked into the final product. This is useful to force the linker to resolve symbols and link the object files into a single module before building a static library. Also, a separate set of link flags can be applied to the prelink allowing additional control over (for instance) exported symbols.
Type	Boolean
Default Value	`NO`

$GENERATE_PKGINFO_FILE

Description	Specifies whether to generate the product's package information file.
Type	Boolean
Values	• `YES` : Generates the product's package information file. • `NO` : Does not generate the product's package information file.
Default Value	• `YES` : In application targets. • `NO` : In other target types.

$GENERATE_PROFILING_CODE

Description	Activating this setting will cause the compiler and linker to generate profiling code. E.g., GCC will generate code suitable for use with gprof(1). Passes flag `-pg`
Type	Boolean
Default Value	`NO`

$GID

Description	Group id of the current user.
Type	String
Default Value	`id -g`

$GROUP

Description	Group name of the current user.
Type	String
Default Value	`id -gn`
Example Value	staff

$HEADERMAP_FILE_FORMAT

Description	Header map file format.
Type	Enumeration
Values	• `traditional` • `plaintext`
Default Value	`traditional`

$HEADERMAP_INCLUDES_FLAT_ENTRIES_FOR_TARGET_BEING_BUILT

Description	Specifies whether the header map contains a name/path entry for every header in the target being built.
Type	Boolean
Values	• `YES` : The header map contains a name/path entry for every header in the target. • `NO` : The header map does not contain name/path entries for the headers that belong to the target.
Default Value	`YES`

$HEADERMAP_INCLUDES_FRAMEWORK_ENTRIES_FOR_ALL_PRODUCT_TYPES

Description	Specifies whether the header map contains a framework-name/path entry for every header in the target being built, including targets that do not build frameworks.
Type	Boolean
Values	• `YES` : The header map contains a framework-name/path entry for every header in the target. • `NO` : The header map does not contain framework-name/path entries for the headers in the target.
Default Value	`YES`

$HEADERMAP_INCLUDES_NONPUBLIC_NONPRIVATE_HEADERS

Description	Specifies if the header map contains non-public and non-private headers.
Type	Boolean
Default Value	`$(HEADERMAP_INCLUDES_PROJECT_HEADERS)`

$HEADERMAP_INCLUDES_PROJECT_HEADERS

Description	Specifies whether the header map contains a name/path entry for every header in the project, regardless of the headers' target membership.
Type	Boolean
Values	• `YES` : The header map contains a name/path entry for every header in the project. • `NO` : The header map does not contain name/path entries for the headers that are part of the project.
Default Value	`YES`

$HEADERMAP_USES_FRAMEWORK_PREFIX_ENTRIES

Type	Boolean
Default Value	`YES`

$HEADERMAP_USES_VFS

Type	Boolean
Default Value	`$(DEFINES_MODULE)`

$HEADER_SEARCH_PATHS

Description	Space-separated list of directory paths. Specifies directories in which to search for header files. (In GCC, this list is passed in the `gcc -I` option.) When this build setting is defined, `$(SDKROOT)` is added to the beginning of each system-header path passed to the compiler.
Type	PathList
Default Value	`$(inherited) $(DT_TOOLCHAIN_DIR)/usr/include`
Example Value	`/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include`

$IBC_COMPILER_AUTO_ACTIVATE_CUSTOM_FONTS

Description	Instructs the XIB compiler to add custom fonts to the application's Info.plist which will cause the fonts to activate upon application launch. Passes flag `--auto-activate-custom-fonts`
Type	Boolean
Default Value	`YES`

$IBC_ERRORS

Description	Show errors encountered during the compilation of XIB files. Passes flag `--errors`
Type	Boolean
Default Value	`YES`

$IBC_FLATTEN_NIBS

Description	Compiles a XIB file into flattened (non-wrapper) NIB file. After flattening, the resulting NIB is more compact but no longer editable by Interface Builder. When this option is disabled, the resulting NIB file remains editable in Interface Builder. If disabled, passes flag `--flatten NO`
Type	Boolean
Default Value	`YES`

$IBC_NOTICES

Description	Show notices encountered during the compilation of XIB files. Passes flag `--notices`
Type	Boolean
Default Value	`YES`

$IBC_OTHER_FLAGS

Description	A list of additional flags to pass to the Interface Builder Compiler. Use this setting if Xcode does not already provide UI for a particular Interface Builder Compiler flag.
Type	StringList
Default Value	empty string

$IBC_WARNINGS

Description	Show warnings encountered during the compilation of XIB files. Passes flag `--warnings`
Type	Boolean
Default Value	`YES`

$ICONV

Description	Path to `iconv` tool.
Type	Path
Default Value	`/usr/bin/iconv`

$INCLUDED_RECURSIVE_SEARCH_PATH_SUBDIRECTORIES

Description	This is a list of fnmatch()-style patterns of file or directory names to include when performing a recursive search. By default this list is empty and is only customized when you want to provide exceptions to the list of filename patterns provided in the "Sub-Directories to Exclude in Recursive Searches".
Type	StringList
Default Value	empty string

$INFOPLIST_EXPAND_BUILD_SETTINGS

Description	Expand build settings in the Info.plist file.
Type	Boolean
Default Value	`YES`

$INFOPLIST_FILE

Description	This is the project-relative path to the plist file that contains the Info.plist information used by bundles.
Type	Path
Default Value	empty string

$INFOPLIST_OTHER_PREPROCESSOR_FLAGS

Description	Other flags to pass to the C preprocessor when preprocessing the Info.plist file.
Type	StringList
Default Value	empty string

$INFOPLIST_OUTPUT_FORMAT

Description	Specifies the output encoding for the output Info.plist (by default, the output encoding will be unchanged from the input). The output endcodings can be 'binary' or 'XML'.
Type	Enumeration
Values	• `same-as-input` • `XML` • `binary`
Default Value	`same-as-input`

$INFOPLIST_PREFIX_HEADER

Description	Implicitly include the given file when preprocessing the Info.plist file. The path given should either be a project relative path or an absolute path.
Type	String
Default Value	empty string

$INFOPLIST_PREPROCESS

Description	Preprocess the Info.plist file using the C Preprocessor.
Type	Boolean
Default Value	`NO`

$INFOPLIST_PREPROCESSOR_DEFINITIONS

Description	Space-separated list of preprocessor macros of the form "foo" or "foo=bar". These macros are used when preprocessing the Info.plist file.
Type	StringList
Default Value	empty string

$INIT_ROUTINE

Description	This is the name of the routine to use for initialization.
Type	String
Default Value	empty string

$INSTALL_DIR

Description	Identifies the directory in the developer's filesystem into which the installed product is placed.
Type	Path
Default Value	`$(DSTROOT)$(INSTALL_PATH)`

$INSTALL_GROUP

Description	The group name or gid for installed products.
Type	String
Default Value	`$(GROUP)`
Example Value	`staff`

$INSTALL_MODE_FLAG

Description	Permissions used for installed product files.
Type	String
Default Value	`u+w,go-w,a+rX`

$INSTALL_OWNER

Description	The owner name or uid for installed products.
Type	String
Default Value	`$(USER)`
Example Value	`genica`

$INSTALL_PATH

Description	Identifies the directory in the user's filesystem into which the installed product is placed.
Type	Path
Values	• Kernel extension project: `$(SYSTEM_LIBRARY_DIR)/Extensions` • Action project: `$(USER_LIBRARY_DIR)/Automator` • Application project: `$(HOME)/Applications` • Audio unit and bundle projects: `$(HOME)/Library/Bundles` • Command-line utility project: `$(HOME)/bin` • Apple plug-in project (complete path depends on specific project template): `$(DSTROOT)` • Dynamic library and static library projects: `/usr/local/lib`
Default Value	empty string

$INSTALL_ROOT

Description	Alias to `$(DSTROOT)`
Type	Path
Default Value	`$(DSTROOT)`

$IPHONEOS_DEPLOYMENT_TARGET

Description	Code will load on this and later versions of iOS. Framework APIs that are unavailable in earlier versions will be weak-linked; your code should check for null function pointers or specific system versions before calling newer APIs. Passes flags `--minimum-deployment-target $(value)`
Type	String
Example Value	`8.2`

$JAVAC_DEFAULT_FLAGS

Description	Default javac flags
Type	String
Default Value	`-J-Xms64m -J-XX:NewSize=4M -J-Dfile.encoding=UTF8`

$JAVA_APP_STUB

Type	Path
Default Value	`$(SYSTEM_LIBRARY_DIR)/Frameworks/JavaVM.framework/Resources/MacOS/JavaApplicationStub`

$JAVA_ARCHIVE_CLASSES

Type	Boolean
Default Value	`YES`

$JAVA_ARCHIVE_TYPE

Type	Enumeration
Values	• `JAR`
Default Value	`JAR`

$JAVA_COMPILER

Description	Path to `javac` tool.
Type	Path
Default Value	`/usr/bin/javac`

$JAVA_FRAMEWORK_RESOURCES_DIRS

Type	PathList
Default Value	`Resources`

$JAVA_JAR_FLAGS

Type	StringList
Default Value	`cv`

$JAVA_SOURCE_SUBDIR

Type	Path
Default Value	`.`

$JAVA_USE_DEPENDENCIES

Type	Boolean
Default Value	`YES`

$JAVA_ZIP_FLAGS

Type	StringList
Default Value	`-urg`

$KEEP_PRIVATE_EXTERNS

Description	Activating this setting will preserve private external symbols rather than turning them into static symbols. This setting is also respected when performing a single-object prelink.
Type	Boolean
Default Value	`NO`

$LD_DEPENDENCY_INFO_FILE

Description	This setting defines the path to which the linker should emit information about what files it used as inputs and generated. Xcode uses this information for its dependency tracking. Setting the value of this setting to empty will disable passing this option to the linker.
Type	Path
Default Value	`$(OBJECT_FILE_DIR_$(CURRENT_VARIANT))/$(CURRENT_ARCH)/$(PRODUCT_NAME)_dependency_info.dat`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/Objects-normal/x86_64/MyProject_dependency_inf`

$LD_DYLIB_INSTALL_NAME

Description	Sets an internal "install path" `$(LC_ID_DYLIB)` in a dynamic library. Any clients linked against the library will record that path as the way dyld should locate this library. If this option is not specified, then the `-o path` will be used. This setting is ignored when building any product other than a dynamic library. Passes flag `-install_name`
Type	Path
Default Value	empty string

$LD_GENERATE_MAP_FILE

Description	Activating this setting will cause the linker to write a map file to disk which details all symbols and their addresses in the output image. The path to the map file is defined by the Path to Link Map File setting. If enabled, passes flags: * `-Xlinker -map` * `-Xlinker $(LD_MAP_FILE_PATH)`
Type	Boolean
Default Value	`YES`

$LD_MAP_FILE_PATH

Description	This setting defines the path to the map file written by the linker when the Write Link Map File setting is activated. By default a separate file will be written for each architecture and build variant, and these will be generated in the Intermediates directory for the target whose produce is being linked.
Type	Path
Default Value	`$(TARGET_TEMP_DIR)/$(PRODUCT_NAME)-LinkMap-$(CURRENT_VARIANT)-$(CURRENT_ARCH).txt`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/MyProject-LinkMap-normal-x86_64.txt`

$LD_NO_PIE

Description	Activating this setting will cause Xcode to not create position independent executables. If enabled, passes flag `-Xlinker -no_pie`
Type	Boolean
Default Value	`NO`

$LD_QUOTE_LINKER_ARGUMENTS_FOR_COMPILER_DRIVER

Description	This setting controls whether arguments to the linker should be 'quoted' using `-Xlinker`. By default Xcode invokes the linker by invoking the driver of the compiler used to build the source files in the target, and passing `-Xlinker` to 'quote' arguments will cause the compiler driver to pass them through to the linker (rather than trying to evaluate them within the driver). By default this setting is enabled. Disabling it will cause Xcode to not use `-Xlinker` to pass arguments to the linker. Disabling this setting is useful if the target has instructed Xcode to use an alternate linker (e.g., by setting the LD setting to the path to another linker) and that alternate linker does not recognize `-Xlinker`.
Type	Boolean
Default Value	`YES`

$LD_RUNPATH_SEARCH_PATHS

Description	This is a list of paths to be added to the runpath search path list for the image being created. At runtime, dyld uses the runpath when searching for dylibs whose load path begins with `@rpath/`. Passes flags: * `-Xlinker -map` * `-Xlinker $(value)`
Type	StringList
Default Value	empty string

$LEGACY_DEVELOPER_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/../PlugIns/Xcode3Core.ideplugin/Contents/SharedSupport/Developer`
Example Value	`/Applications/Xcode.app/Contents/PlugIns/Xcode3Core.ideplugin/Contents/SharedSupport/Developer`

$LEX

Description	Path to `lex` tool.
Type	Path
Default Value	`lex`

$LEXFLAGS

Description	Space-separated list of flags to pass to lex
Type	StringList
Default Value	empty string

$LIBRARY_FLAG_NOSPACE

Description	No space between the library flag and the library name.
Type	Boolean
Default Value	`YES`

$LIBRARY_FLAG_PREFIX

Description	Prefix for passing libraries.
Type	String
Default Value	`-l`

$LIBRARY_KEXT_INSTALL_PATH

Type	Path
Default Value	`$(LOCAL_LIBRARY_DIR)/Extensions`

$LIBRARY_SEARCH_PATHS

Description	This is a list of paths to folders to be searched by the linker for libraries used by the product. Paths are delimited by whitespace, so any paths with spaces in them need to be properly quoted.
Type	PathList
Default Value	empty string

$LINKER_DISPLAYS_MANGLED_NAMES

Description	Activating this setting causes the linker to display mangled names for C++ symbols. Normally, this is not recommended, but turning it on can help to diagnose and solve C++ link errors. If enabled, passes flag `-Xlinker --no-demangle`
Type	Boolean
Default Value	`NO`

$LINK_WITH_STANDARD_LIBRARIES

Description	If this setting activated, then the compiler driver will automatically pass its standard libraries to the linker to use during linking. If desired, this flag can be used to disable linking with the standard libraries, and then individual libraries can be passed as Other Linker Flags. If disabled, passes flag `-nostdlib`
Type	Boolean
Default Value	`YES`

$LLVM_IMPLICIT_AGGRESSIVE_OPTIMIZATIONS

Description	If aggressive optimizations are enabled.
Type	Boolean
Default Value	`$(LLVM_OPTIMIZATION_LEVEL_VAL_$(GCC_OPTIMIZATION_LEVEL))`

$LLVM_LTO

Description	Enabling this setting allows the optimizer to look across object files in your program and optimize across file boundaries during linking. If enabled, passes flag `-flto`
Type	Boolean
Default Value	`NO`

$LLVM_OPTIMIZATION_LEVEL_VAL_0

Type	Boolean
Default Value	`NO`

$LLVM_OPTIMIZATION_LEVEL_VAL_1

Type	Boolean
Default Value	`NO`

$LLVM_OPTIMIZATION_LEVEL_VAL_2

Type	Boolean
Default Value	`NO`

$LLVM_OPTIMIZATION_LEVEL_VAL_3

Type	Boolean
Default Value	`NO`

$LLVM_OPTIMIZATION_LEVEL_VAL_fast

Type	Boolean
Default Value	`YES`

$LLVM_OPTIMIZATION_LEVEL_VAL_s

Type	Boolean
Default Value	`NO`

$LOCAL_ADMIN_APPS_DIR

Type	Path
Default Value	`/Applications/Utilities`

$LOCAL_APPS_DIR

Type	Path
Default Value	`/Applications`

$LOCAL_DEVELOPER_DIR

Type	Path
Default Value	`/Library/Developer`

$LOCAL_LIBRARY_DIR

Type	Path
Default Value	`/Library`

$MACH_O_TYPE

Description	This setting determines the format of the produced binary and how it can be linked when building other binaries.
Type	Enumeration
Values	• `mh_execute` : • `mh_dylib` : Linker flag `-dynamiclib` • `mh_bundle` : Linker flag `-bundle` • `staticlib` : • `mh_object` : Linker flag `-r`
Default Value	empty string

$MACOSX_DEPLOYMENT_TARGET

Description	Code will load on this and later versions of OS X. Framework APIs that are unavailable in earlier versions will be weak-linked; your code should check for null function pointers or specific system versions before calling newer APIs. Passes flag `-mmacosx-version-min=$(value)`
Type	Enumeration
Values	• empty string: Compiler Default - Code will load on any Mac OS system that supports the APIs that are used. • `10.4` : OS X 10.4 - Code will not load on systems earlier than 10.4. • `10.5` : OS X 10.5 - Code will not load on systems earlier than 10.5. • `10.6` : OS X 10.6 - Code will not load on systems earlier than 10.6. • `10.7` : OS X 10.7 - Code will not load on systems earlier than 10.7. • `10.8` : OS X 10.8 - Code will not load on systems earlier than 10.8. • `10.9` : OS X 10.9 - Code will not load on systems earlier than 10.9. • `10.10` : OS X 10.10 - Code will not load on systems earlier than 10.10.
Default Value	empty string
Example Value	`10.9`

$MODULEMAP_FILE

Description	This is the project-relative path to the LLVM module map file that defines the module structure for the compiler. If empty, it will be automatically generated for appropriate products when `$(DEFINES_MODULE)` is enabled.
Type	String
Default Value	empty string

$MODULEMAP_PRIVATE_FILE

Description	This is the project-relative path to the LLVM module map file that defines the module structure for private headers.
Type	String
Default Value	empty string

$MODULE_CACHE_DIR

Description	Absolute path of folder in which compiler stores its cached "cmodules" - this cache is a performance improvement.
Type	Path
Default Value	`$(DERIVED_DATA_DIR)/ModuleCache`

$MODULE_NAME

Description	This is the identifier of the kernel module listed in the generated stub. This is only used when building kernel extensions.
Type	String
Default Value	empty string

$MODULE_START

Description	This defines the name of the kernel module start routine. This is only used when building kernel extensions.
Type	String
Default Value	empty string

$MODULE_STOP

Description	This defines the name of the kernel module stop routine. This is only used when building kernel extensions.
Type	String
Default Value	empty string

$MODULE_VERSION

Description	This is the version of the kernel module listed in the generated stub. This is only used when building kernel extensions.
Type	String
Default Value	empty string

$MTL_ENABLE_DEBUG_INFO

Description	Produce debugging information. This information is required for shader profiling. If enabled, passes flag `-gline-tables-only`
Type	Boolean
Default Value	`NO`

$NATIVE_ARCH

Description	Identifies the architecture on which the build is being performed (same as `$(CURRENT_ARCH)`).
Type	String
Example Value	`i386`

$OBJC_ABI_VERSION

Description	Objective-C ABI version
Type	String

$OBJECT_FILE_DIR

Description	Partially identifies the directory into which variant object files are placed. The complete specification is computed using the variants of this build setting.
Type	Path
Default Value	`$(TARGET_TEMP_DIR)/Objects`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/Objects`

$OBJECT_FILE_DIR_

Description	Fully identifies the directory into which variant object files are placed. For each build variant in `$(BUILD_VARIANTS)`, Xcode generates an `$(OBJECT_FILE_DIR)` build setting with the variant name as a suffix. The generated build setting's value is computed using `$(OBJECT_FILE_DIR)` and the build variant name.
Type	Path
Default Value	`$(OBJECT_FILE_DIR)-$(CURRENT_VARIANT)`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build/Objects-normal`

$OBJROOT

Description	The path where intermediate files will be placed during a build. Intermediate files include generated sources, object files, etc. Shell script build phases can place and access files here, as well. Typically this path is not set per target, but is set per-project or per-user.
Type	Path
Default Value	`$(PROJECT_DIR)/build`

$ONLY_ACTIVE_ARCH

Description	Specifies whether the product includes only object code for the native architecture.
Type	Boolean
Values	• `YES` : The product includes only code for the native architecture `$(NATIVE_ARCH)` . • `NO` : The product includes code for the architectures specified in `$(ARCHS)` .
Default Value	• When `$(CONFIGURATION)=Debug` : `YES` • Otherwise: `NO`

$ORDER_FILE

Description	The path to a file which alters the order in which functions and data are laid out. For each section in the output file, any symbol in that section that are specified in the order file is moved to the start of its section and laid out in the same order as in the order file. Order files are text files with one symbol name per line. Lines starting with a # are comments. A symbol name may be optionally preceded with its object file leafname and a colon (e.g. foo.o:_foo). This is useful for static functions/data that occur in multiple files. A symbol name may also be optionally preceded with the architecture (e.g. `ppc:_foo` or `ppc:foo.o:_foo`). This enables you to have one order file that works for multiple architectures. Literal c-strings may be ordered by quoting the string in the order file (e.g. "Hello, world"). Generally you should not specify an order file in Debug or Development configurations, as this will make the linked binary less readable to the debugger. Use them only in Release or Deployment configurations.
Type	String
Default Value	empty string

$OS

Description	Current OS
Type	String
Default Value	`MACOS`

$OSAC

Description	Path to `osacompile` tool.
Type	Path
Default Value	`/usr/bin/osacompile`

$OSACOMPILE_EXECUTE_ONLY

Description	Saves the output script in execute-only form: the script can be run, but cannot be opened in Script Editor or Xcode. With this option turned off, a user may see the original script source by opening the script.
Type	Boolean
Default Value	NO

$OTHER_CFLAGS

Description	Space-separated list of additional flags to pass to the compiler for C and Objective-C files. Be sure to backslash-escape any arguments that contain spaces or special characters (e.g. path names that may contain spaces). Use this setting if Xcode does not already provide UI for a particular C or Objective-C compiler flag.
Type	StringList
Default Value	empty string

$OTHER_CODE_SIGN_FLAGS

Description	This build setting allows you to pass any additional options you need to codesign(1) as a single space-delimited string.
Type	StringList
Default Value	empty string

$OTHER_CPLUSPLUSFLAGS

Description	Space-separated list of additional flags to pass to the compiler for C++ and Objective-C++ files. Be sure to backslash-escape any arguments that contain spaces or special characters (e.g. path names that may contain spaces). Use this setting if Xcode does not already provide UI for a C++ or Objective-C++ compiler flag.
Type	StringList
Default Value	`$(OTHER_CFLAGS)`

$OTHER_LDFLAGS

Description	Options defined in this setting are passed to invocations of the linker.
Type	StringList
Default Value	empty string

$OTHER_LIBTOOLFLAGS

Description	Options defined in this setting are passed to all invocations of the archive librarian, which is used to generate static libraries.
Type	StringList
Default Value	empty string

$OTHER_OSACOMPILEFLAGS

Description	Space-separated list of additional flags to pass to osacompile. Be sure to backslash-escape any arguments that contain spaces or special characters (e.g. path names that may contain spaces). Use this setting if Xcode does not already provide UI for a particular osacompile flag.
Type	String
Default Value	empty string

$PATH_PREFIXES_EXCLUDED_FROM_HEADER_DEPENDENCIES

Description	Space-separated list of directory paths. Identifies the directories to exclude from header-file scans when the build uses header-file dependencies.
Type	PathList
Default Value	`/usr/include /usr/local/include /System/Library/Frameworks /System/Library/PrivateFrameworks $(SYSTEM_DEVELOPER_DI`

$PLATFORM_DEVELOPER_APPLICATIONS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications`

$PLATFORM_DEVELOPER_BIN_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/usr/bin`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr/bin`

$PLATFORM_DEVELOPER_LIBRARY_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Library`
Example Value	`/Applications/Xcode.app/Contents/Developer/Library`

$PLATFORM_DEVELOPER_SDK_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Platforms/MacOSX.platform/Developer/SDKs`
Example Value	`/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs`

$PLATFORM_DEVELOPER_TOOLS_DIR

Type	String
Default Value	`$(DEVELOPER_DIR)/Tools`
Example Value	`/Applications/Xcode.app/Contents/Developer/Tools`

$PLATFORM_DEVELOPER_USR_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/usr`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr`

$PLATFORM_DIR

Description	Path to the current platform bundle.
Type	Path
Default Value	`xcrun --show-sdk-platform-path --sdk $(SDKROOT)`

$PLATFORM_NAME

Description	Name taken from the platform bundle.
Type	String
Default Value	`macosx`

$PLATFORM_PREFERRED_ARCH

Description	Preferred architectures, value taken from the platform bundle.
Type	String

$PLATFORM_PRODUCT_BUILD_VERSION

Description	Build version taken from the platform bundle.
Type	String

$PLIST_FILE_OUTPUT_FORMAT

Description	Specifies the output encoding for Property List (.plist) files (by default, the output encoding will be unchanged from the input). The output endcodings can be `binary` or `XML`.
Type	Enumeration
Values	• `same-as-input` • `XML` • `binary`
Default Value	`same-as-input`

$PRECOMPILE_PREFIX_HEADER

Description	Old setting, see `$(GCC_PRECOMPILE_PREFIX_HEADER)`
Type	Boolean
Default Value	`NO`

$PRECOMPS_INCLUDE_HEADERS_FROM_BUILT_PRODUCTS_DIR

Description	This setting allows for better control of sharing precompiled prefix header files between projects. By default, Xcode assumes that the prefix header file may include header files from the build directory if the build directory is outside of the project directory. (Xcode cannot determine this ahead of time since other projects may not have been built into the shared build directory at the time the information is needed.) If your prefix file never includes files from the build directory you may set this to `NO` to improve sharing of precompiled headers. If the prefix does use files from a build directory which is inside your project directory, you may set this to `YES` to avoid unintended sharing that may result in build failures.
Type	Boolean
Default Value	`YES`

$PREFIX_HEADER

Description	Old setting, see `$(GCC_PREFIX_HEADER)`
Type	Path
Default Value	empty string

$PRELINK_FLAGS

Description	Additional flags to pass when performing a single-object prelink.
Type	StringList
Default Value	empty string

$PRELINK_LIBS

Description	Additional libraries to pass when performing a single-object prelink.
Type	StringList
Default Value	empty string

$PRESERVE_DEAD_CODE_INITS_AND_TERMS

Description	Activating this setting (in combination with the Dead Code Stripping `-dead_strip option`) causes the `-no_dead_strip_inits_and_terms` flag to be passed to ld(1) via cc(1) to disable dead code stripping for initialization and termination routines. This option should not be used without the aforementioned Dead Code Stripping option.
Type	Boolean
Default Value	`NO`

$PRIVATE_HEADERS_FOLDER_PATH

Description	This is the location to copy the private headers during building, relative to the built products folder.
Type	Path
Default Value	empty string

$PRODUCT_DEFINITION_PLIST

Description	Path to a file specifying additional requirements for a product archive.
Type	String
Default Value	empty string

$PRODUCT_MODULE_NAME

Description	The name to use for the source code module constructed for this target, and which will be used to import the module in implementation source files. Must be a valid identifier.
Type	String
Default Value	`$(PRODUCT_NAME)`
Example Value	`MyProject`

$PRODUCT_NAME

Description	Specifies the name of the product the target builds.
Type	String
Default Value	The name of the target at the time it was created. `$(TARGET_NAME)`
Example Value	`MyProject`

$PROJECT

Description	Name of the project.
Type	String
Example Value	`MyProject`

$PROJECT_DERIVED_FILE_DIR

Description	Path to the derived sources for the current project.
Type	Path
Default Value	`$(OBJROOT)/$(PROJECT_NAME).build/DerivedSources`
Example Value	`/Users/genica/MyProject/build/MyProject.build/DerivedSources`

$PROJECT_DIR

Type	Path
Example Value	`/Users/genica/MyProject`

$PROJECT_FILE_PATH

Description	Path to the project file currently being worked from.
Type	Path
Example Value	`/Users/genica/MyProject/MyProject.xcodeproj`

$PROJECT_NAME

Description	Specifies the name of the project that defines the target.
Type	String
Default Value	The name of the project at the time it was created.
Example Value	`MyProject`

$PROJECT_TEMP_DIR

Description	Identifies the directory in which the project's intermediate build files are placed. This directory is shared between all the targets defined by the project. Run Script build phases should generate intermediate build files in the directory identified by `$(DERIVED_FILE_DIR)`, not the location this build setting specifies.
Type	Path
Default Value	`$(PROJECT_TEMP_ROOT)/$(PROJECT_NAME).build`
Example Value	`/Users/genica/MyProject/build/MyProject.build`

$PROJECT_TEMP_ROOT

Type	Path
Default Value	`$(SRCROOT)/$(SYMROOT)`
Example Value	`/Users/genica/MyProject/build`

$PROVISIONING_PROFILE

Description	The UUID of a valid provisioning profile. A missing or invalid profile will cause a build error.
Type	String
Default Value	empty string

$PUBLIC_HEADERS_FOLDER_PATH

Description	This is the location to copy the public headers during building, relative to the built products folder.
Type	Path
Default Value	empty string

$REMOVE_CVS_FROM_RESOURCES

Type	Boolean
Default Value	`YES`

$REMOVE_GIT_FROM_RESOURCES

Type	Boolean
Default Value	`YES`

$REMOVE_HEADERS_FROM_EMBEDDED_BUNDLES

Type	Boolean
Default Value	`YES`

$REMOVE_HG_FROM_RESOURCES

Type	Boolean
Default Value	`YES`

$REMOVE_SVN_FROM_RESOURCES

Type	Boolean
Default Value	`YES`

$RETAIN_RAW_BINARIES

Description	Specifies whether the binary is stripped.
Type	Boolean
Values	• `YES` : Binary is not stripped. • `NO` : Binary is stripped.
Default Value	`NO`

$REZ_COLLECTOR_DIR

Description	Specifies the directory in which the collected Resource Manager resources generated by ResMerger are stored before they are added to the product.
Type	Path
Default Value	`$(TARGET_TEMP_DIR)/ResourceManagerResources`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/ResourceManagerResources`

$REZ_OBJECTS_DIR

Description	SSpecifies the directory in which compiled Resource Manager resources generated by Rez are stored before they are collected using ResMerger.
Type	Path
Default Value	`$(REZ_COLLECTOR_DIR)/Objects`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/ResourceManagerResources/Objects`

$REZ_SEARCH_PATHS

Description	This is a list of paths to search for files with resource manager resources. Paths are delimited by whitespace, so any paths with spaces in them need to be properly quoted.
Type	String
Default Value	empty string

$RUN_CLANG_STATIC_ANALYZER

Description	Activating this setting will cause Xcode to run the Clang static analysis tool on qualifying source files during every build.
Type	Boolean
Default Value	`NO`

$SCAN_ALL_SOURCE_FILES_FOR_INCLUDES

Description	Activating this setting will cause all source files to be scanned for includes (e.g. of header files) when computing the dependency graph, in which case if an included file is changed then the including file will be rebuilt next time a target containing it is built. Normally only certain types of files - such as C-language source files - are scanned. This setting is useful if your project contains files of unusual type which are compiled using a custom build rule.
Type	Boolean
Default Value	`NO`

$SDKROOT

Description	Specifies the directory of the base SDK to use to build the product.
Type	String
Values	`xcodebuild -showsdks` to display available SDKs.
Default Value	`macosx`

$SDK_DIR

Description	Path to SDK bundle
Type	Path
Default Value	`xcrun --show-sdk-path`

$SDK_NAME

Description	Name of current SDK.
Type	String
Example Value	`macosx10.10`

$SDK_PRODUCT_BUILD_VERSION

Description	Build version from the SDK bundle.
Type	String

$SECTORDER_FLAGS

Description	These flags are typically used to specify options for ordering symbols within segments, for example the `-sectorder` option to ld. Generally you should not specify symbol ordering options in Debug or Development configurations, as this will make the linked binary less readable to the debugger. Use them only in Release or Deployment configurations.
Type	StringList
Default Value	empty string

$SED

Description	Path to `sed` tool.
Type	Path
Default Value	`/usr/bin/sed`

$SEPARATE_STRIP

Description	Activating this setting when the linked product is to be stripped will cause stripping to occur via a separate invocation of strip(1). Otherwise stripping will occur during linking, if possible.
Type	Boolean
Default Value	`NO`

$SEPARATE_SYMBOL_EDIT

Description	Activating this setting when the linked product's symbols are to be edited will cause editing to occur via a separate invocation of nmedit(1). Otherwise editing will occur during linking, if possible.
Type	Boolean
Default Value	`NO`

$SHARED_PRECOMPS_DIR

Description	Specifies the directory in which to place precompiled headers. Targets can share precompiled headers by specifying the same value for this build setting.
Type	Path
Default Value	`$(CACHE_ROOT)/SharedPrecompiledHeaders`

$SKIP_INSTALL

Description	Specifies whether to place the product at the location indicated by `$(DSTROOT)` or the uninstalled products directory inside the directory indicated by `$(TARGET_TEMP_DIR)`.
Type	Boolean
Values	• `YES` : When `$(DEPLOYMENT_LOCATION)=YES` , the product is placed in `$(TARGET_TEMP_DIR)/UninstalledProducts` . • `NO` : The product is placed in `$(DSTROOT)` .
Default Value	`NO`

$SOURCE_ROOT

Type	Path
Default Value	`$(SRCROOT)`
Example Value	`/Users/genica/MyProject`

$SRCROOT

Description	Identifies the directory containing the target's source files.
Type	Path
Default Value	`.`
Example Value	`/Users/genica/MyProject`

$STRINGS_FILE_OUTPUT_ENCODING

Description	Specify the encoding to be used for the output files (by default, the output encoding will be 16-bit Unicode). The value can be either an NSStringEncoding, i.e. one of the numeric values recognized by NSString, or it can be an IANA character set name as understood by CFString. The operation will fail if the file cannot be converted to the specified encoding.
Type	String
Values	• `UTF-16` • `UTF-8` • `binary`
Default Value	`UTF-16`

$STRIPFLAGS

Description	Additional flags to be passed when stripping the linked product of the build.
Type	Boolean
Default Value	Default value will be computed from `$(UNSTRIPPED_PRODUCT)`

$STRIP_INSTALLED_PRODUCT

Description	Activating this setting causes the linked product of the build to be stripped of symbols as part of deployment postprocessing.
Type	Boolean
Default Value	`YES`

$STRIP_STYLE

Description	Defines the level of symbol stripping to be performed on the linked product of the build. The default value is defined by the target's product type.
Type	Enumeration
Values	• `all` : All Symbols - Completely strips the binary, removing the symbol table and relocation information. Passes flag `-s` • `non-global` : Non-Global Symbols - Strips non-global symbols, but saves external symbols. Passes flag `-x` • `debugging` : Debugging Symbols - Strips debugging symbols, but saves local and global symbols. Passes flag `-S`
Default Value	`all`

$SUPPORTED_PLATFORMS

Description	The list of supported platforms from which a base SDK can be used. This setting is used if the product can be built for multiple platforms using different SDKs.
Type	StringList
Example Value	`macosx`

$SWIFT_OPTIMIZATION_LEVEL

Description	Swift optimization level.
Type	Enumeration
Values	• `-Onone` : None • `-O` : Fastest • `-Ounchecked` : Fastest, Unchecked
Default Value	`-Onone`

$SYMROOT

Description	The path at which all products will be placed when performing a build. Typically this path is not set per target, but is set per-project or per-user.
Type	Path
Default Value	`$(PROJECT_DIR)/build`

$SYSTEM_ADMIN_APPS_DIR

Type	Path
Default Value	`/Applications/Utilities`

$SYSTEM_APPS_DIR

Type	Path
Default Value	`/Applications`

$SYSTEM_CORE_SERVICES_DIR

Type	Path
Default Value	`/System/Library/CoreServices`

$SYSTEM_DEMOS_DIR

Type	Path
Default Value	`/Applications/Extras`

$SYSTEM_DEVELOPER_APPS_DIR

Type	Path
Default Value	`$(DEVELOPER_APPLICATIONS_DIR)`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications`

$SYSTEM_DEVELOPER_BIN_DIR

Type	Path
Default Value	`$(DEVELOPER_BIN_DIR)`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr/bin`

$SYSTEM_DEVELOPER_DEMOS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications/Utilities/Built Examples`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications/Utilities/Built Examples`

$SYSTEM_DEVELOPER_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)`
Example Value	`/Applications/Xcode.app/Contents/Developer`

$SYSTEM_DEVELOPER_DOC_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/ADC Reference Library`
Example Value	`/Applications/Xcode.app/Contents/Developer/ADC Reference Library`

$SYSTEM_DEVELOPER_GRAPHICS_TOOLS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications/Graphics Tools`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications/Graphics Tools`

$SYSTEM_DEVELOPER_JAVA_TOOLS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications/Java Tools`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications/Java Tools`

$SYSTEM_DEVELOPER_PERFORMANCE_TOOLS_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications/Performance Tools`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications/Performance Tools`

$SYSTEM_DEVELOPER_RELEASENOTES_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/ADC Reference Library/releasenotes`
Example Value	`/Applications/Xcode.app/Contents/Developer/ADC Reference Library/releasenotes`

$SYSTEM_DEVELOPER_TOOLS

Type	Path
Default Value	`$(DEVELOPER_TOOLS_DIR)`
Example Value	`/Applications/Xcode.app/Contents/Developer/Tools`

$SYSTEM_DEVELOPER_TOOLS_DOC_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/ADC Reference Library/documentation/DeveloperTools`
Example Value	`/Applications/Xcode.app/Contents/Developer/ADC Reference Library/documentation/DeveloperTools`

$SYSTEM_DEVELOPER_TOOLS_RELEASENOTES_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/ADC Reference Library/releasenotes/DeveloperTools`
Example Value	`/Applications/Xcode.app/Contents/Developer/ADC Reference Library/releasenotes/DeveloperTools`

$SYSTEM_DEVELOPER_USR_DIR

Type	Path
Default Value	`$(DEVELOPER_USR_DIR)`
Example Value	`/Applications/Xcode.app/Contents/Developer/usr`

$SYSTEM_DEVELOPER_UTILITIES_DIR

Type	Path
Default Value	`$(DEVELOPER_DIR)/Applications/Utilities`
Example Value	`/Applications/Xcode.app/Contents/Developer/Applications/Utilities`

$SYSTEM_DOCUMENTATION_DIR

Type	Path
Default Value	`/Library/Documentation`

$SYSTEM_KEXT_INSTALL_PATH

Type	Path
Default Value	`/System/Library/Extensions`

$SYSTEM_LIBRARY_DIR

Type	Path
Default Value	`/System/Library`

$TARGETNAME

Description	Alias of `$(TARGET_NAME)`
Type	String
Example Value	`MyProject`

$TARGET_BUILD_DIR

Description	Identifies the root of the directory hierarchy that contains the product's files (no intermediate build files). Run Script build phases that operate on product files of the target that defines them should use the value of this build setting. But Run Script build phases that operate on product files of other targets should use `$(BUILT_PRODUCTS_DIR)` instead.
Type	Path
Default Value	`$(CONFIGURATION_BUILD_DIR)`
Example Value	`/Users/genica/MyProject/build/build/Debug`

$TARGET_NAME

Description	Name of current build target.
Type	String
Example Value	`MyProject`

$TARGET_TEMP_DIR

Description	Identifies the directory containing the target's intermediate build files. Run Script build phases should place intermediate files at the location indicated by `$(DERIVED_FILE_DIR)`, not the directory identified by this build setting.
Type	Path
Default Value	`$(CONFIGURATION_TEMP_DIR)/$(TARGET_NAME).build`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build`

$TARGETED_DEVICE_FAMILY

Description	Comma-separated list of numeric identifiers. Specifies the device families on which the product must be capable of running.
Type	String
Values	• `1` : iPhone/iPod touch. • `2` : iPad.
Default Value	`1`
Example Value	`1,2`

$TEMP_DIR

Type	Path
Default Value	`$(TARGET_TEMP_DIR)`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build`

$TEMP_FILES_DIR

Type	Path
Default Value	`$(TEMP_DIR)`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build`

$TEMP_FILE_DIR

Type	Path
Default Value	`$(TEMP_DIR)`
Example Value	`/Users/genica/MyProject/build/MyProject.build/Debug/MyProject.build`

$TEMP_ROOT

Type	Path
Default Value	`$(OBJROOT)`
Example Value	`/Users/genica/MyProject/build`

$TEST_HOST

Description	Path to the executable into which a bundle of tests is injected. Only specify this setting if testing an application or other executable.
Type	String
Default Value	empty string

$TREAT_MISSING_BASELINES_AS_TEST_FAILURES

Description	When running tests that measure performance via XCTestCase, report missing baselines as test failures.
Type	Boolean
Default Value	`NO`

$UID

Description	Current user id.
Type	String
Default Value	`id -u`
Example Value	`501`

$UNEXPORTED_SYMBOLS_FILE

Description	This is a project-relative path to a file that lists the symbols not to export. Passes flag `-unexported_symbols_list`
Type	String
Default Value	empty string

$UNSTRIPPED_PRODUCT

Type	Boolean
Default Value	`NO`

$USER

Description	Current user name.
Type	String
Default Value	`$(USER)`
Example Value	`genica`

$USER_APPS_DIR

Type	Path
Default Value	`$(HOME)/Applications`
Example Value	`/Users/genica/Applications`

$USER_HEADER_SEARCH_PATHS

Description	This is a list of paths to folders to be searched by the compiler for included or imported user header files (those headers listed in quotes) when compiling C, Objective-C, C++, or Objective-C++. Paths are delimited by whitespace, so any paths with spaces in them need to be properly quoted. See the description of the `$(ALWAYS_SEARCH_USER_PATHS)` build setting for more details on how this setting is used. If the compiler doesn't support the concept of user headers, then the search paths are prepended to the any existing header search paths defined in `$(HEADER_SEARCH_PATHS)` .
Type	PathList
Default Value	empty string

$USER_LIBRARY_DIR

Type	Path
Default Value	`$(HOME)/Library`
Example Value	`/Users/genica/Library`

$USE_HEADERMAP

Description	Enable use of headermap files.
Type	Boolean
Default Value	`YES`

$USE_HEADER_SYMLINKS

Description	Part of headermap settings.
Type	Boolean
Default Value	`NO`

$VALIDATE_PRODUCT

Description	Specifies whether to run product-validation tests.
Type	Boolean
Values	• `YES` : The build runs validation tests on the generated product. • `NO` : The build does not run validation tests on the generated product.
Default Value	`NO`

$VALID_ARCHS

Description	Space-separated list of identifiers. Specifies the architectures for which the binary may be built. During the build, this list is intersected with the value of `$(ARCHS)` build setting; the resulting list specifies the architectures the binary can run on. If the resulting architecture list is empty, the target generates no binary.
Type	StringList
Default Value	Taken from platform
Example Value	`i386 x86_64`

$VERSIONING_SYSTEM

Description	Selects the process used for version-stamping generated files.
Type	String
Values	• empty string: None - Use no versioning system. • `apple-generic` : Apple Generic - Use the current project version setting.
Default Value	empty string

$VERSION_INFO_BUILDER

Description	This defines a reference to the user performing a build to be included in the generated Apple Generic Versioning stub.
Type	String
Default Value	$(USER)

$VERSION_INFO_EXPORT_DECL

Description	This defines a prefix string for the version info symbol declaration in the generated Apple Generic Versioning stub. This can be used, for example, to add an optional 'export' keyword to the version symbol declaration. This should rarely be changed.
Type	String
Default Value	empty string

$VERSION_INFO_FILE

Description	Used to specify a name for the source file that will be generated by Apple Generic Versioning and compiled into your product.
Type	String
Default Value	`$(PRODUCT_NAME)_vers.c`

$VERSION_INFO_PREFIX

Description	Used as a prefix for the name of the version info symbol in the generated versioning source file. If you prefix your exported symbols you will probably want to set this to the same prefix.
Type	String
Default Value	empty string

$VERSION_INFO_SUFFIX

Description	Used as a suffix for the name of the version info symbol in the generated versioning source file. This is rarely used.
Type	String
Default Value	empty string

$WARNING_CFLAGS

Description	Space-separated list of additional warning flags to pass to the compiler. Use this setting if Xcode does not already provide UI for a particular compiler warning flag.
Type	StringList
Default Value	empty string

$WARNING_LDFLAGS

Description	These flags are passed with linker invocations, and by default give the `-no_arch_warnings` flag to the linker to avoid many warnings being generated during multi-architecture builds.
Type	StringList
Default Value	empty string

$WRAPPER_EXTENSION

Description	This is the extension used for product wrappers, which has a default value based on the product type.
Type	String
Default Value	empty string

$XCODE_APP_SUPPORT_DIR

Type	Path
Default Value	`$(DEVELOPER_LIBRARY_DIR)/Xcode`
Example Value	`/Applications/Xcode.app/Contents/Developer/Library/Xcode`

$XCODE_PRODUCT_BUILD_VERSION

Description	This value is hard-coded into Xcode's frameworks.
Type	String
Example Value	`6C131e`

$XCODE_VERSION_ACTUAL

Description	This value is hard-coded into Xcode's frameworks.
Type	String
Example Value	`0620`

$XCODE_VERSION_MAJOR

Description	This value is hard-coded into Xcode's frameworks.
Type	String
Example Value	`0600`

$XCODE_VERSION_MINOR

Description	This value is hard-coded into Xcode's frameworks.
Type	String
Example Value	`0020`

$YACC

Description	Path to `yacc` tool
Type	Path
Default Value	`yacc`

$YACCFLAGS

Description	Space-separated list of flags to pass to yacc
Type	StringList
Default Value	empty string

[home | parent | top]