

A COMPILER LEGACY SUPPORT

Overview

The VisualDSP++ environment and tools provide several types of support for legacy code that was developed with previous releases of the development tools. For more information on legacy code support see the *VisualDSP++ Linker and Utilities Manual for ADSP-21xx DSPs*.

Tools Differences

VisualDSP++ 2.0 includes a new C/C++ Compiler, Linker, and Debugger, and a binary file format, ELF. It also introduces an integrated development and debugging environment (IDDE) for easy management of your ADSP-219x projects.

There are a number of enhancements to the other tools as well. As a result, VisualDSP++, Release 2.0 has some differences from Release 6.1 that you will need to be aware of. In some cases you will need to modify your sources to use the new tools.

Of the new features and enhancements, the following have the most impact on your existing projects:

- Some tools switches have changed. If you use any of the modified or obsolete switches, you must revise your command line scripts or batch files in order to rebuild your project.

Tools Differences

- The code generation tools no longer support AEXE-format DSP executables (.exe). They now generate ELF-format DSP executables (.dxe), and the debugger requires DSP executables to be in the ELF/DWARF-2 format. As a result, AEXE-formatted files must be recompiled or reassembled in order to be debugged under VisualDSP++ 2.0. An ELF/DWARF-to-AEXE conversion utility is available in VisualDSP++ 2.0 that will perform back-conversion. An AEXE-to-ELF conversion utility performs forward conversion.
- Some assembly instructions and directives have changed from the VisualDSP 6.1 syntax, but a `-legacy` assembler switch has been provided to assemble files in the old syntax. You may need to review diagnostic messages and revise your source code in order to reassemble your source. Legacy syntax and the new syntax under VisualDSP++ 2.0 cannot be used together in the same source file. They can be mixed together within the same project, as long as they are assembled in different source files.
- Some C compiler extensions to the ISO/ANSI standard have changed. If you use any of the modified or removed extensions, you must revise your code in order to rebuild your project.
- The run-time model has changed. If you call a VisualDSP Release 6.1 assembly language subroutine from your C/C++ program, you must revise the assembly code to comply with the new rules for the C/C++ run-time environment.
- The Architecture File (.ACH) is no longer supported. If you re-link using your Release 6.1 object files or object libraries, you must create a Linker Description File for each object or object library before using the new Linker.

The remainder of this section describes these and other known differences between VisualDSP Releases 6.1 and VisualDSP++ 2.0. It also provides assistance when possible for making these required changes.

C/C++ Compiler and C/C++ Run-time Library

The new cc219x compiler provided in VisualDSP++ 2.0 does not support some switches and extensions that were available in the g21 compiler. As a result, the compiler supports a set of new rules for the run-time environment. This section lists the extensions and switches that have been removed, replaced, or whose function works differently than in Release 6.1. For further details about the cc219x compiler, see [Chapter 2 Compiler](#).

Segment Placement Support Keyword Changed to Section

The `segment()` placement keyword has changed to `section()`. The `section()` construct now precedes the variable declaration, and its argument is a string. For example:

```
section("my_sec") int myvar;
```

For more information about the `section()` construct, see [“Placement Support Keyword \(section\)” on page 2-69](#).

G21 Compatibility Call

The cc219x compiler provides a special G21 compatibility call that enables use of existing libraries with the new compiler. The `extern OldAsmCall` declaration can be added to the prototype(s) of the functions developed under Release 6.1. Your programs will be faster, smaller, and more reliable after the C code is upgraded to use the new compiler.



This convention is similar to the C++/C linkage specification.

Support for G21-Based Options And Extensions

The cc219x compiler supports most of the switches and extensions of the previous GNU-based compiler release. For a list of absolute or modified options, see [“Compiler Switch Modifications” on page A-5](#).

ANSI C Extensions

The following extensions are no longer supported, or their functions have been modified:

- `typedef` — This extension was used to define the type of an expression.
- **Complex types:** `complex`, `creal`, `cimag`, and `conj` — These extensions were used to define complex numbers. Although you cannot write complex number literals, you can have a complex type defined with real and imaginary components. These types need to be managed by the programmer. Support for complex types using such an approach is used in the `libdsp` definitions and use of various complex types.
- **Compound statements within expressions** — This extension was used to declare variables within an expression. You can achieve these results using `inline` functions.
- **Iterator types:** `iter` and `sum` — These extensions created loop expressions that were used as a shorthand for working with arrays.
- **Assigning variables to specific registers:** `asm` — This extension was used to declare a variable and specify a machine register in which to store it.

If you use any of these extensions in your C source code, revise that source.

Compiler Switch Modifications

The switches listed in [Table A-1](#) have been removed or their actions have been modified. If you use any of these switches to compile your C/C++ code, remove or replace the switch before recompiling the code with the new cc219x compiler.

Table A-1.C/C++ Compiler: Obsolete and Replaced Switches

Release 6.1 Switch	Operation under Release 6.1	Change for VisualDSP++ 2.0
-a	Specify Architecture File.	Replaced with -T <Linker Description File> in linker command line. Subsumed in IDDE's build process.
-dD, -dM, and -dN	Output the results of preprocessing and/or list of #defines.	Removed.
-deps	Instruct driver to rebuild only components of a target that have changed.	Removed.
-Fno-<high med low>	Disable specified optimization level.	Removed. Optimization controlled with -O switches.
-fcond-mismatch	Allow conditional expression mismatch.	Removed.
-finline-functions	Force function inlining.	Removed.
-fkeep-inline-functions	Force keeping of inlined functions.	Removed.
-fno-asm	Don't recognize asm as a keyword.	Replaced with -no-extra-keywords.
-fno-builtin	Don't recognize builtin functions.	Replaced with -no-builtin.

Tools Differences

Table A-1.C/C++ Compiler: Obsolete and Replaced Switches (Cont'd)

Release 6.1 Switch	Operation under Release 6.1	Change for VisualDSP++ 2.0
-fsigned-bitfields -funsigned-bitfields	Control whether bit field is signed or unsigned.	Removed. Bit field is signed or unsigned based on the sign of the type definition declaring the bit-field.
-fno-signed-bitfields -fno-unsigned-bitfields	Negative form of the -fsigned-bitfield and -funsigned-bitfield.	Removed. Bit field is signed or unsigned based on the sign of the type definition declaring the bit-field.
-fsigned-char -funsigned-char	Specify whether to default to signed or unsigned char type.	Replaced with -signed-char and -unsigned-char.
-fsyntax-only	Check syntax only; no output.	Replaced with -syntax-only.
-fwritable-strings	Store string constants in the writable data segment.	Removed. String constants are placed in the seg_data1 section. Definition in the .ldf can place this section in RAM or ROM.
-imacros	Process macro file.	Removed.
-MD and -MMD	Output rules for the make utility; used with -E.	Removed.
-mboot-page=	Specify boot page.	Removed. The ADSP-219x processors do not support paging.
-mdmdata= -mpmdata= -mdcode=	Specify target architecture file segments.	Removed. You can control placement of object file segments using the SECTIONS{} command in the Linker Description File.

Table A-1.C/C++ Compiler: Obsolete and Replaced Switches (Cont'd)

Release 6.1Switch	Operation under Release 6.1	Change for VisualDSP++ 2.0
-mlistm	Merge C code with assembler-generated code.	Removed.
-mno-doloops	Don't generate loop structures in assembled code.	Removed. The compiler only generates do loop control structures when it is safe to do so and the compiler is generating optimized code (-O, -Os). Interrupt handling routines should save and restore the loop stack if they are to use the same construct to avoid overflowing the loop stacks.
-mno-inits	Don't initialize variables in assembled code.	Removed.
-mpjump	Place the jump table in pm memory.	Removed.
-mreserved=	Instructs the compiler not to use specified registers.	Removed. Replaced with -reserve
-mrom	Make the module a ROM module.	Removed.
-msmall-code	Optimize for size, not for speed.	Removed. Replaced with -Os.
-mstatic-spill	Use dm memory when all register are used.	Removed.
-nostdinc	Do not search standard system directories for header files.	Replaced with -no-std-inc.
-nostdlib	Do not use standard system libraries and startup files when linking.	Replaced with -no-std-lib.

Tools Differences

Table A-1.C/C++ Compiler: Obsolete and Replaced Switches (Cont'd)

Release 6.1Switch	Operation under Release 6.1	Change for VisualDSP++ 2.0
-runhdr	Specify a particular runtime header.	Removed.
-traditional-cpp	Support some preprocessing features.	Removed.

New and Obsolete Warnings

The VisualDSP++ 2.0 compiler includes several new warning switches. These switches control the number and type of messages reported during a given compilation. They are described in [Table A-2](#).

Table A-2.C/C++ Compiler: New Warning Switches

VisualDSP ++ 2.0 Warning Switch	Description
-warn-protos	Produce a warning when a function is called without a full prototype.
-Wdriver-limit <i>number</i>	Set a maximum <i>number</i> of driver errors.
-Werror-limit <i>number</i>	Set a maximum <i>number</i> of compiler errors.
-Wremarks	Indicates that the compiler may issue remarks, which are diagnostic messages even milder than warnings.
-Wterse	Enable terse warnings.
-pedantic	Causes the compiler to generate warnings for any constructs in a C or C++ source file that does not conform to the ANSI standard.
-W<error remark suppress warn> <num> [, num ...]	Overrides the severity of specific compilation diagnostic messages, where <i>num</i> is the number representing the message to override.

The Release 6.1 warning switches that are no longer supported are listed in [Table A-3](#):

Table A-3.C Compiler: Obsolete Warning Switches

-Wall	-Wformat	-Wswitch
-Wchar-subscripts	-Wimplicitit	-Wtrigraphs
-Wcomment	-Wparentheses	-Wuninitialized
-Werror	-Wreturn-type	-Wunused

See “[Compiler Command Line Switches](#)” in [Chapter 2](#) for further information on the compiler’s switch set.

Run-Time Model

The cc219x Compiler in VisualDSP++ 2.0 produces code that is not fully compatible with the Release 6.1 run-time model. VisualDSP++ 2.0 has significant changes in the registers and stack usage. These changes are especially important if you call an assembly language subroutine from a C/C++ program, or a C/C++ function from an assembly language program. For more information about the VisualDSP++ 2.0 run-time model, see “[C/C++ Run-Time Model](#)” in [Chapter 2](#).

C/C++ Run-Time Library

This release includes a set of documented ANSI standard routines that you can call from your C/C++ programs. Many of these routines have been modified to provide better support for the improved performance and code compilation of cc219x.

Tools Differences

[Table A-4](#) lists the C/C++ run-time libraries, that have been modified for VisualDSP++ 2.0.

Table A-4.C/C++ Compiler: Modified C/C++ Run-Time Library Functions

Header File	Purpose	Change for VisualDSP++ 2.0
math.h	Mathematics	Added functions: acos, asin, atan, atan2, ceil, cos, cosh, exp, fabs, floor, fmod, frexp, ldexp, log, log10, modf, pow, sin, sinh, sqrt, tan, tanh
signal.h	Signal Handling	Improved functions: clear_interrupt, interrupt, raise, signal
stdio.h	Input/Output	Added function: perror
stdlib.h	Standard Library	Removed functions: isascii, toascii
cctype.h	Character Handling	Added function: strtod

The cc219x compiler release now includes a complete documented DSP library (libdsp.dlb) and associated include files, providing efficient standard functions required by DSP application designers. For details, refer to [Chapter 4, “DSP Run-Time Library.”](#)



Future cc219x compiler releases may include additional library functions. For complete information on the library contents, see [Chapter 3, “C/C++ Run-Time Library.”](#)