2 FEATURES AND TOOLS

In This Chapter

This chapter contains the following topics:

- "VisualDSP++ Features" on page 2-2
- "Code Development Tools" on page 2-5

VisualDSP++ Features

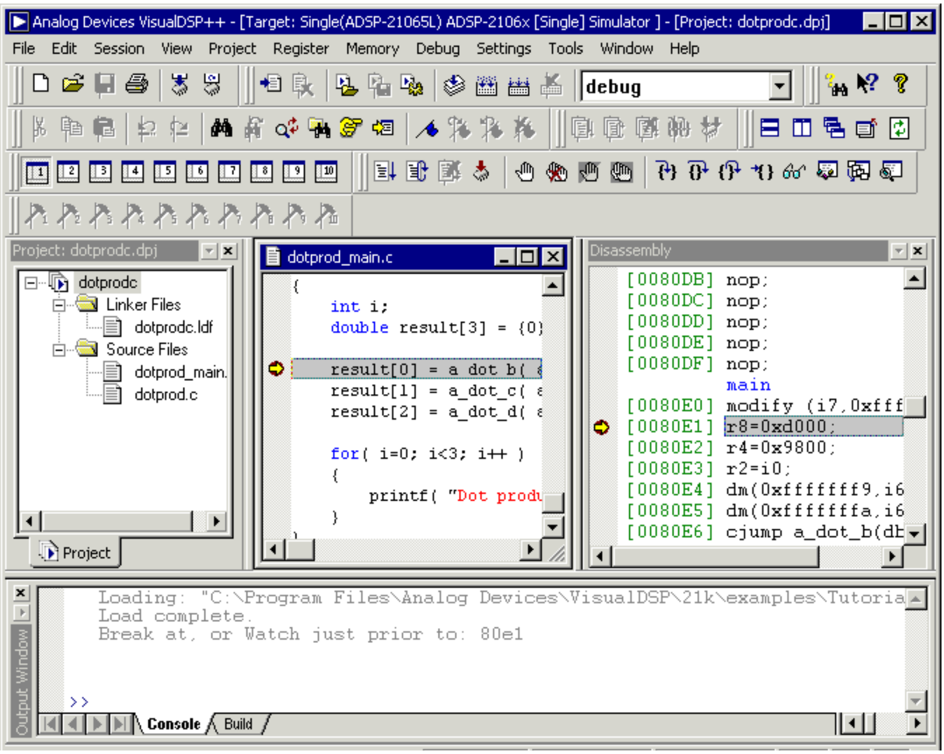
VisualDSP++ provides the following features:

- **Extensive editing capabilities**. You create and modify source files by using multiple language syntax highlighting, drag-and-drop, bookmarks, and other standard editing operations. You can view files generated by the code development tools.
- **Flexible project management**. You specify a project definition that identifies the files, dependencies, and tools that you will use to build projects. You can create this project definition once or modify it to meet changing development needs.
- **Easy access to code development tools**. Analog Devices provides the following code development tools: C/C++ compiler, assembler, linker, splitter, and loader. You specify options for these tools by using dialog boxes instead of complicated command-line scripts. Options that control how the tools process inputs and generate outputs have a one-to-one correspondence to command-line switches. You can define options for a single file or an entire project. You can define these options once or modify them as necessary.
- **Flexible project build options**. You control builds at the file or project level. VisualDSP++ enables you to build files or projects selectively, update project dependencies, or incrementally build only the files that have changed since the previous build. You view the status of your project build in progress. If your build reports an error, double-click on the file name in the error message to open that source file. You can then correct the error, rebuild the file or project, and start a debug session.
- **VisualDSP++ Kernel (VDK) Support**. You can add VDK support to a project to structure and scale application development. The **Kernel** tab page of the Project window enables you to manipulate events, event bits, priorities, semaphores, and thread types.

- **Flexible workspace management.** You can create up to ten workspaces and quickly switch between them. Assigning a different project to each workspace enables you to build and debug multiple projects in a single session.

- **Easy movement between debug and build activities.** You start the debug session and move freely between editing, build, and debug activities.

Figure 2-1 shows the Integrated Development and Debugging Environment.



```
{
    int i;
    double result[3] = {0}

    result[0] = a_dot_b( 
    result[1] = a_dot_c( 
    result[2] = a_dot_d( 

    for( i=0; i<3; i++ )
    {
        printf( "Dot produ
    }
```

```
[0080DB] nop;
[0080DC] nop;
[0080DD] nop;
[0080DE] nop;
[0080DF] nop;
         main
[0080E0] modify (i7,0xfff
[0080E1] r8=0xd000;
[0080E2] r4=0x9800;
[0080E3] r2=i0;
[0080E4] dm(0xfffffff9,i6
[0080E5] dm(0xfffffffa,i6
[0080E6] cjump a_dot_b(db
```

```
Loading: "C:\Program Files\Analog Devices\VisualDSP\21k\examples\Tutoria
Load complete.
Break at, or Watch just prior to: 80e1

>>
```

Figure 2-1. The VisualDSP++ IDDE

VisualDSP++ reduces your debugging time by providing these key features:

- **Easy-to-use debugging activities**. You debug with one common, easy-to-use interface for all processor simulators and emulators, or hardware evaluation and development boards. You can easily switch between these targets.
- **Multiple language support**. You debug programs written in C, C++, or assembly, and view your program in machine code. For programs written in C/C++, you view the source in C/C++ or mixed C/C++ and assembly, and display the values of local variables or evaluate expressions (global and local) based on the current context.
- **Effective debug control**. You set breakpoints on symbols and addresses, then step through your program's execution to find problems in coding logic. You set watchpoints (conditional breakpoints) on registers, stacks, and memory locations to identify when they are accessed. You use trace to see the sequence of operations performed prior to the breakpoint. When you use conditional breakpoints with trace, you quickly find problems that are caused by system interactions.
- **Tools for improving performance**. You use the trace, profile, and linear and statistical profiles to identify bottlenecks in your DSP application and to identify program optimization needs. You use plotting to view data arrays graphically. You generate interrupts, outputs, and inputs to simulate real-world application conditions.
- **Multiprocessor debugging**. You can easily manage and debug any number of processors from the same debug session. Fully synchronous multiprocessor operations such as step, run, and halt allow cycle-accurate debugging of complex systems. You can arrange processors into logical groups for better control of a system's behavior.

Code Development Tools

The ADSP-21*xxx* code development tools include:

- C/C++ compiler
- Run-time library with over 100 math, DSP, and C run-time library routines
- Assembler
- Linker
- Splitter
- Loader
- Simulator
- Emulator

These tools support the ADSP-21*xxx* family of 32-bit DSPs, and enable you to develop applications that take full advantage of the ADSP-21*xxx* architecture.

VisualDSP++ includes a linker that supports multiprocessing, shared memory, and memory overlays.

The code development tools provide the following key features:

- **Easy-to-program C, C++, and assembly languages**. You can program in C/C++, assembly, or mix C/C++ and assembly in one source. The assembly language is based on an algebraic syntax that is easy to learn, program, and debug.
- **Flexible system definition**. You define multiple types of executables for a single type of DSP in one Linker Description File (`.LDF`). You specify input files, including objects, libraries, shared memory files, overlay files, and executables.
- **Support for overlays, multiprocessors, and shared memory executables**. The linker places code and resolves symbols in multiprocessor memory space for use by multiprocessor systems. The loader enables you to configure multiprocessors with less code and faster boot time. You can create host, link port, and PROM boot images.

Software and hardware tool kits include context-sensitive Help and manuals in `.PDF` format.

For details about assembly syntax, refer to the *VisualDSP++ 2.0 Assembler and Preprocessor Manual for ADSP-21xxx DSPs*.