VisualDSP++™ 3.1
Getting Started Guide for Blackfin Processors

Revision 2.1, April 2003

Part Number
82-000410-15

Analog Devices, Inc.
One Technology Way
Norwood, Mass. 02062-9106

ANALOG DEVICES

Copyright Information

Printed in the USA.

Disclaimer

Trademark and Service Mark Notice

Revision 2.1

Contents

TUTORIAL

PREFACE

Thank you for purchasing VisualDSP++™, the development software for Analog Devices processors.

Purpose of This Manual

The *VisualDSP++ 3.1 Getting Started Guide for Blackfin Processors* provides a step-by-step tutorial that highlights many of the VisualDSP++ features. By completing this tutorial you will become familiar with the VisualDSP++ environment, and you will learn how to use these features in your own digital signal processing (DSP) development projects.

Intended Audience

This manual is intended for DSP programmers who are familiar with Analog Devices processors. The manual assumes that the audience has a working knowledge of Analog Devices processor architecture and instruction set.

DSP programmers who are unfamiliar with Analog Devices processors should refer to their processor's Hardware Reference and Instruction Set Reference, which describe the processor architecture and instruction set. Note that the *ADSP-BF533 Blackfin Processor Hardware Reference* includes information about the ADSP-BF531 and ADSP-BF532 processors.

Manual Contents

This guide contains the following chapters.

- Chapter 1, “Features and Tools”

  Provides an overview of VisualDSP++ features and code development tools

- Chapter 2, “Tutorial”

  Provides step-by-step instructions for creating sessions, and for building and debugging projects by using examples of C/C++ and assembly sources

  The tutorial is organized to follow the steps that you take in developing a typical programming project. Before you begin actual programming, you should be familiar with the architecture of your particular processor and the other software development tools.

What’s New in This Manual

The older Blackfin® processor names, ADSP-21532 and ADSP-21535, have been deprecated and replaced with ADSP-BF532 and ADSP-BF535, respectively.

Technical or Customer Support

You can reach DSP Tools Support in the following ways.

- Visit the DSP Development Tools website at
  `http://www.analog.com/technology/dsp/developmentTools/index.html`
- Submit a DSP Tools Technical Support Form:

  `http://forms.analog.com/Form_Pages/DSP/tools/contactDSP.asp`
- Email questions to `dsptools.support@analog.com`
- Phone questions to **1-800-ANALOGD**
- Contact your ADI local sales office or authorized distributor
- Send questions by mail to

  ```
  Analog Devices, Inc.
  DSP Division
  One Technology Way
  P.O. Box 9106
  Norwood, MA 02062-9106
  USA
  ```

Supported Processors

The name *Blackfin* refers to the family of Analog Devices 16-bit, fixed-point processors. VisualDSP++ currently supports the following Blackfin processors.

- ADSP-BF531 processor
- ADSP-BF532 processor (formerly ADSP-21532)
- ADSP-BF533 processor

- ADSP-BF535 processor (formerly ADSP-21535)
- ADSP-DM102 processor
- AD6532 processor

Product Information

You can obtain product information from the Analog Devices website, from the product CD-ROM, or from the printed publications (manuals).

Analog Devices is online at `www.analog.com`. Our website provides information about a broad range of products—analog integrated circuits, amplifiers, converters, and digital signal processors.

MyAnalog.com

MyAnalog.com is a free feature of the Analog Devices website that allows customization of a webpage to display only the latest information on products you are interested in. You can also choose to receive weekly email notification containing updates to the webpages that meet your interests. MyAnalog.com provides access to books, application notes, data sheets, code examples, and more.

Registration:

Visit `www.myanalog.com` to sign up. Click **Register** to use MyAnalog.com. Registration takes about five minutes and serves as means for you to select the information you want to receive.

If you are already a registered user, just log on. Your user name is your email address.

DSP Product Information

For information on digital signal processors, visit our website at `www.analog.com/dsp`, which provides access to technical publications, data sheets, application notes, product overviews, and product announcements.

You may also obtain additional information about Analog Devices and its products in any of the following ways.

- Email questions or requests for information to `dsp.support@analog.com`
- Fax questions or requests for information to
  1-781-461-3010 (North America)
  089/76 903-557 (Europe)
- Access the Digital Signal Processing Division's FTP website at
  `ftp ftp.analog.com` or **ftp 137.71.23.21**
  `ftp://ftp.analog.com`

Related Documents

For information on product related development software, see the following publications.

VisualDSP++ 3.1 User's Guide for Blackfin Processors

VisualDSP++ 3.1 C/C++ Compiler and Library Manual for Blackfin Processors

VisualDSP++ 3.1 C/C++ Assembler and Preprocessor Manual for Blackfin Processors

VisualDSP++ 3.1 Linker and Utilities Manual for Blackfin Processors

VisualDSP++ 3.1 Product Bulletin

VisualDSP++ Kernel (VDK) User's Guide

VisualDSP++ Component Software Engineering User's Guide

Quick Installation Reference Card

Online Technical Documentation

Online documentation comprises VisualDSP++ Help system and tools manuals, Dinkum Abridged C++ library and FlexLM network license manager software documentation. You can easily search across the entire VisualDSP++ documentation set for any topic of interest. For easy printing, supplementary `.PDF` files for the tools manuals are also provided.

A description of each documentation file type is as follows.

File	Description
`.CHM`	Help system files and VisualDSP++ tools manuals.
`.HTML`	Dinkum Abridged C++ library and FlexLM network license manager software documentation. Viewing and printing the `.HTML` files require a browser, such as Internet Explorer 4.0 (or higher).
`.PDF`	VisualDSP++ tools manuals in Portable Documentation Format, one `.PDF` file for each manual. Viewing and printing the `.PDF` files require a PDF reader, such as Adobe Acrobat Reader (4.0 or higher).

If documentation is not installed on your system as part of the software installation, you can add it from the VisualDSP++ CD-ROM at any time by rerunning the Tools installation.

Access the online documentation from the VisualDSP++ environment, Windows Explorer, or Analog Devices website.

From VisualDSP++

- Access VisualDSP++ online Help from the Help menu's **Contents**, **Search**, and **Index** commands.
- Open online Help from context-sensitive user interface items (toolbar buttons, menu commands, and windows).

From Windows

In addition to any shortcuts you may have constructed, there are many ways to open VisualDSP++ online Help or the supplementary documentation from Windows.

Help system files (.CHM files) are located in the Help folder, and .PDF files are located in the Docs folder of your VisualDSP++ installation. The Docs folder also contains the Dinkum Abridged C++ library and FlexLM network license manager software documentation.

Using Windows Explorer

- Double-click any file that is part of the VisualDSP++ documentation set.
- Double-click the vdsp-help.chm file, which is the master Help system, to access all the other .CHM files.

Using the Windows Start Button

- Access VisualDSP++ online Help by clicking the **Start** button and choosing **Programs**, **VisualDSP**, and **VisualDSP++ Documentation**.
- Access the .PDF files by clicking the **Start** button and choosing **Programs**, **VisualDSP**, **Documentation for Printing**, and the name of the book.

From the Web

To download the tools manuals, point your browser at http://www.analog.com/technology/dsp/developmentTools/gen_purpose.html.

Select a DSP family and book title. Download archive (.ZIP) files, one for each manual. Use any archive management software, such as WinZip, to decompress downloaded files.

Printed Manuals

For general questions regarding literature ordering, call the Literature Center at **1-800-ANALOGD** (**1-800-262-5643**) and follow the prompts.

VisualDSP++ Documentation Set

VisualDSP++ manuals may be purchased through Analog Devices Customer Service at **1-781-329-4700**; ask for a Customer Service representative. The manuals can be purchased only as a kit. For additional information, call **1-603-883-2430**.

If you do not have an account with Analog Devices, you will be referred to Analog Devices distributors. To get information on our distributors, log onto `http://www.analog.com/salesdir/continent.asp`.

Hardware Manuals

Hardware reference and instruction set reference manuals can be ordered through the Literature Center or downloaded from the Analog Devices website. The phone number is **1-800-ANALOGD** (**1-800-262-5643**). The manuals can be ordered by a title or by product number located on the back cover of each manual.

Data Sheets

All data sheets can be downloaded from the Analog Devices website. As a general rule, any data sheet with a letter suffix (L, M, N) can be obtained from the Literature Center at **1-800-ANALOGD** (**1-800-262-5643**) or downloaded from the website. Data sheets without the suffix can be downloaded from the website only—no hard copies are available. You can ask for the data sheet by a part name or by product number.

If you want to have a data sheet faxed to you, the phone number for that service is **1-800-446-6212**. Follow the prompts and a list of data sheet code numbers will be faxed to you. Call the Literature Center first to find out if requested data sheets are available.

Contacting DSP Publications

Please send your comments and recommendation on how to improve our manuals and online Help. You can contact us by:

- Emailing `dsp.techpubs@analog.com`
- Filling in and returning the attached Reader's Comments Card found in our manuals

Notation Conventions

The following table identifies and describes text conventions used in this manual.

Additional conventions, which apply only to specific chapters, may appear throughout this document.

Example	Description
Close command (**File** menu) or **OK**	Text in **bold** style indicates the location of an item within the VisualDSP++ environment's menu system. For example, the **Close** command appears on the **File** menu.
`{this | that}`	Alternative required items in syntax descriptions appear within curly brackets and separated by vertical bars; read the example as `this` or `that`.
`[this | that]`	Optional items in syntax descriptions appear within brackets and separated by vertical bars; read the example as an optional `this` or `that`.
`[this,…]`	Optional item lists in syntax descriptions appear within brackets delimited by commas and terminated with an ellipsis; read the example as an optional comma-separated list of `this`.
`.SECTION`	Registers, commands, directives, keywords, code examples, and feature names are in text with `letter gothic` font.
`filename`	Non-keyword placeholders appear in text with italic style format.
Note:	A note, providing information of special interest or identifying a related topic. In the online version of this book, the word **Note** appears instead of this symbol.
Caution:	A caution, providing information about critical design or programming issues that influence operation of a product. In the online version of this book, the word **Caution** appears instead of this symbol.

1 FEATURES AND TOOLS

This chapter contains the following topics.

VisualDSP++ Features

VisualDSP++ provides the following features.

- **Extensive editing capabilities.** Create and modify source files by using multiple language syntax highlighting, drag-and-drop, bookmarks, and other standard editing operations. View files generated by the *code development* tools.
- **Flexible project management.** Specify a project definition that identifies the files, dependencies, and tools that you will use to build projects. Create this project definition once or modify it to meet changing development needs.

- **Easy access to code development tools.** Analog Devices provides these code development tools: C/C++ compiler, assembler, linker, splitter, and loader. 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. Define options for a single file or for an entire project. Define these options once or modify them as necessary.
- **Flexible project build options.** 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. View the status of your project build in progress. If the build reports an error, double-click on the file name in the error message to open that source file. Then correct the error, rebuild the file or project, and start a debug session.
- **VisualDSP++ Kernel (VDK) Support**. 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.** 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 1-1 shows the Integrated Development and Debugging Environment.

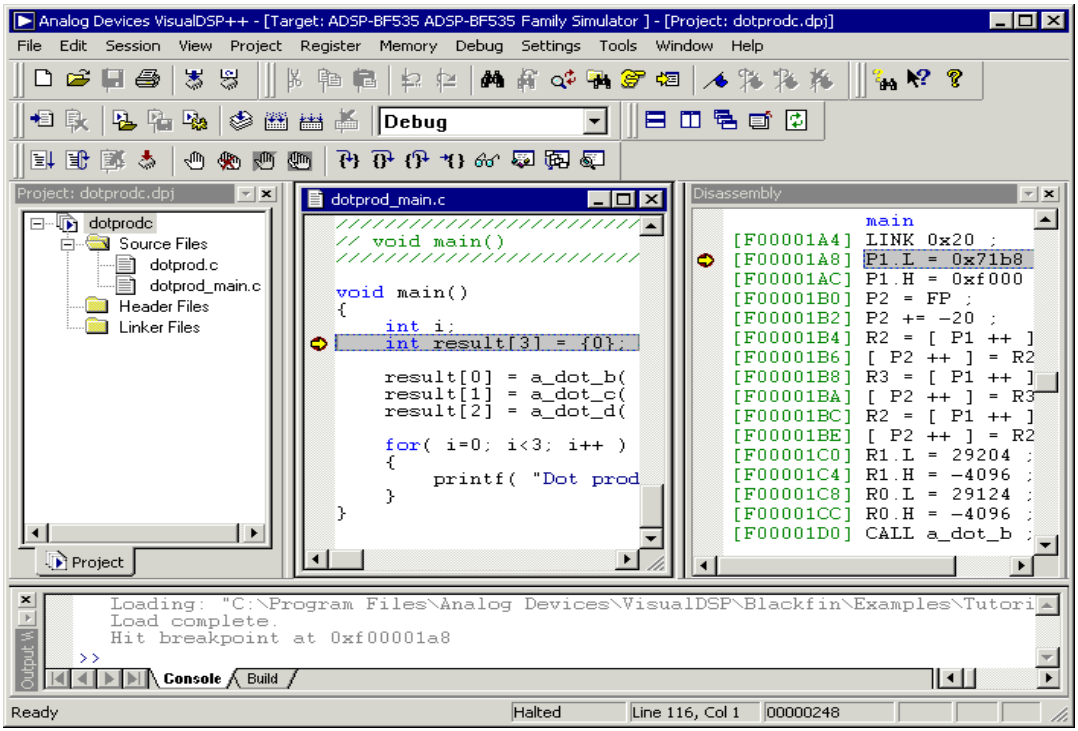

Figure 1-1. The VisualDSP++ IDDE

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

- **Easy-to-use debugging activities.** Debug with one common, easy-to-use interface for all processor simulators and emulators, or hardware evaluation and development boards. Switch easily between these targets.
- **Multiple language support.** Debug programs written in C, C++, or assembly, and view your program in machine code. For programs written in C/C++, you can 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.** Set breakpoints on symbols and addresses and then step through the program's execution to find problems in coding logic. Set watchpoints (conditional breakpoints) on registers, stacks, and memory locations to identify when they are accessed.
- **Tools for improving performance.** Use the trace, profile, and linear and statistical profiles to identify bottlenecks in your DSP application and to identify program optimization needs. Use plotting to view data arrays graphically. Generate interrupts, outputs, and inputs to simulate real-world application conditions.

New Features in Release 3.1

The following features are new in Release 3.1.

- **Compiled simulation**. A traditional simulator decodes and interprets one instruction at a time. Each executed instruction often requires repeated decoding. Compiled simulation removes the overhead of having to decode each instruction repeatedly.

  Compiled simulation is process whereby the .DXE that may be loaded into a traditional simulator is converted into an .EXE file that will execute directly on the system hosting VisualDSP++. The execution speed of a compiled simulation program is greater than that of a standard .DXE program.

  Compiled simulation employs a simulation compiler that preprocesses instructions in the .DXE file and generates an intermediate C++ source program. This program is compiled and linked with a standard set of libraries to produce an .EXE file that effects the simulation of the original .DXE file.

  In the **Project Options** dialog box, a new tabbed page, **Compiled Simulation**, lets you enable optimization and specify additional options. For details about using compiled simulation, refer to the appendix "Simulation of Blackfin Processors" in the *VisualDSP++ 3.1 User's Guide for Blackfin Processors* or to the online Help.

- **New processor support**. Release 3.1 supports these new Blackfin processors: ADSP-BF531, ADSP-BF533, ADSP-DM102, and AD6532.

- **ROM Splitter**. A ROM splitter is available from the **Loader** page in the **Project Options** dialog box. Selecting **ROM splitter options** from the **Category** pull-down menu lets you enable the splitter as well as specify a **Mask address** and **Additional options**. For more information about using the splitter, refer to the online Help.

- **New support for handling global uninitialized data**. You can create a smaller executable file (`.DXE`) by initializng the *uninitialized data* section to zero at runtime.

  A new check box, **Global un-init data into bsz**, was added to the **Compile** page in the **Project Options** dialog box. Selecting this option tells the compiler to capture all global uninitialized data and put it in its own `INPUT_SECTION`. On the **Link** page a new pull-down menu, **Runtime initialization**, lets you specify **None** (do not perform runtime initialization) or **bsz** (perform runtime initialization).

  If you do not select **bsz**, the data is collected but is still in the `.DXE` file. If you select **bsz**, the data is placed in a compressed format in the `.DXE` file, and the original data is removed. At runtime the data is expanded back in its original location.

  For information about using the memory intializer tool with the compiler and linker, see the *VisualDSP++ 3.1 C/C++ Compiler and Library Manual for Blackfin Processors* and the *VisualDSP++ 3.1 Linker and Utilities Manual for Blackfin Processors*.

Code Development Tools

Code development tools for the Blackfin processor include:

- C/C++ compiler
- Runtime library with over 100 math, DSP, and C runtime library routines
- Assembler
- Linker
- Splitter

- Loader
- Simulator
- Emulator (must be purchased separately from VisualDSP++)

These tools support the family of 16-bit Blackfin processors, and enable you to develop applications that take full advantage of the Blackfin processor 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.** 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.** Define multiple types of executables for a single type of processor in one Linker Description File (`.LDF`). 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. 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++ 3.1 Assembler and Preprocessor Manual for Blackfin Processors*.

VisualDSP++ Help System

The VisualDSP++ Help system is designed to help you obtain information quickly. You can use the Help system's table of contents, index, full-text search function, bookmark function, and extensive hyperlinks to jump to topics.

VisualDSP++ Help is a merge of several Help systems (`.CHM` files). Each is identified with a book icon  in your product installation's **Help** folder.

Most of the Help system comprises VisualDSP++ tools *manuals*, such as the Assembler and Preprocessor manuals. These manuals are also provided in PDF format (on installation disk) for printing and are available from Analog Devices as printed books.

Some `.CHM` files support *pop-up* messages for dialog box controls (buttons, fields, and so on). These messages, which appear in little yellow boxes, compose part of the context-sensitive Help in VisualDSP++.

The Help system describes the VisualDSP++ *user interface*. Help files include concepts, procedures, and reference information. Each toolbar button, menu-bar command, and debugging window in VisualDSP++ is linked to a topic in one of these files.

Using the Help Window

The Help window comprises three parts:

- The *Navigation pane* provides tabbed pages (**Contents**, **Index**, **Search**, and **Favorites**) that show different navigational views.
- The *Viewing pane* displays the selected object (topic, Web page, video, `.PDF` file, application).
- *Toolbar buttons* enable you to navigate or specify options.

Figure 1-2 shows the parts of the **VisualDSP++ Help** window.

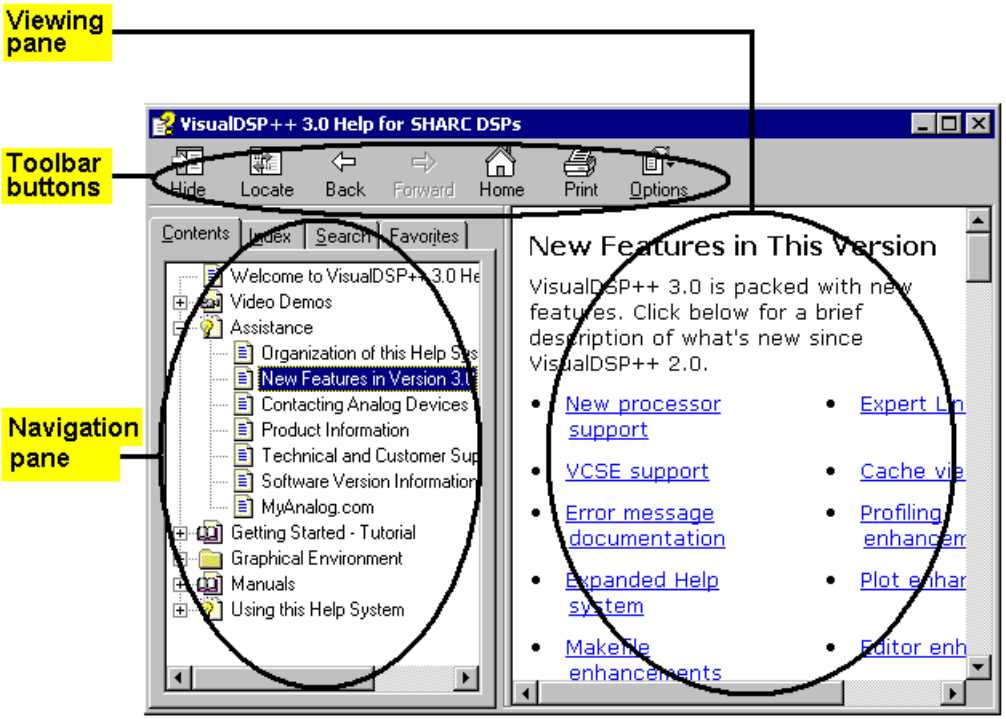

Figure 1-2. Parts of the VisualDSP++ Help Window

Invoking Online Help

You can invoke online Help from VisualDSP++ or from the Windows **Start** button. You can also access Help manually via Windows Explorer.

To access online Help from VisualDSP++, choose **Contents**, **Search**, or **Index** from the **Help** menu.

To access online Help from the Windows **Start** button, click the **Start** button and choose **Programs**, **VisualDSP**, and **VisualDSP Documentation**.

The Help function is programmed to look for the Help system in the VisualDSP++ `Help` folder.

By default, the VisualDSP++ software installation procedure places the complete set of Help files (except the *Getting Started Guide*) in the folder `VisualDSP\Help`.

If you receive an error message after invoking Help, the Help system:

- May not have been loaded onto your PC
- May have been deleted
- May reside in a directory other than the default directory

To locate the help (`.CHM`) files manually, use the Windows **Search** function as follows.

1. Record the Help file (`.CHM`) named in the error message.
2. From the Windows **Start** button, choose **Search** and **For Files or Folders**. Enter the name of the `.CHM` file from step 1.
3. After locating the file, launch it manually by clicking the file name from the **Search Results** window or from Windows Explorer.

Viewing Context-Sensitive Help

You can view context-sensitive Help (help pertinent to your current activity) for various items in VisualDSP++.

VisualDSP++'s context-sensitive Help is linked to toolbar buttons, menu commands, windows, and dialog box items.

Viewing Menu, Toolbar, or Window Help

1. Click the toolbar's Help button  or press **Shift+F1**.

   The mouse pointer becomes a Help pointer  .

2. Move the Help pointer over a menu command, toolbar button, or window.

3. Click the mouse to open the Help window. A description of the object appears in the right panel.

Viewing Dialog Box Button or Field Help

Perform one of these actions:

- Select a field or button in a dialog box and press **F1** or **Shift+F1**.
- Click the Question-Mark button  in the top-right corner of the dialog box.

  The mouse pointer becomes a Help pointer  .

  Move the Help pointer over a dialog box control (button or field) and click the mouse. A description of the object appears in a yellow pop-up window.

- Position the mouse pointer over a label or control (button or filed) in a dialog box and right-click.

  A **What's This** button 

  appears. Move the mouse pointer over the **What's This** button and click.

"What's This" Help is not configured for all items.

Viewing Window Help

1. Click the window to make it active.
2. Press the **F1** key to open the Help window.

   A description of the window appears in the right panel.

Using Help Window Navigation Buttons

You can move through the Help system and view Help topics by using the Help window's navigational aids, as shown in Figure 1-3.

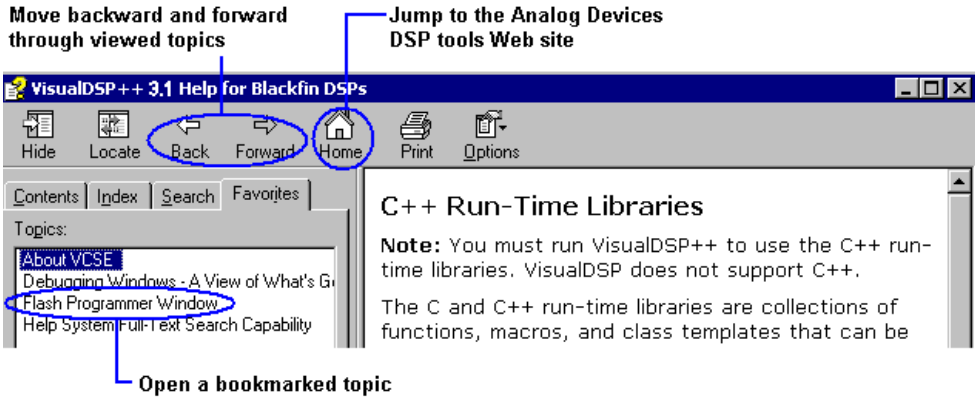

Figure 1-3. Help Window's Navigational Aids

Other standard Microsoft HTML Help buttons are described in Table 1-1.

Table 1-1. Standard Microsoft HTML Help Buttons

Button	Purpose
	Hides the Help window's left pane. This button narrows the Help window.
	Displays the Help window's left pane. This button restores a full view after you click **Hide**.
	Highlights the name of the current topic on the **Contents** page (left pane). After you jump around the Help system, this button shows the current topic's relation to other topics.

Copying Example Code from Help

You can copy code from the Help system and then paste it into your application. Be aware that the copied text may carry unwanted control codes. For example, if you copy a hyphen with a parameter, the actual code of the copied hyphen may be an ASCII `0x96` instead of an ASCII `0x2D`. The hyphen may look OK, but it will cause an error when the command runs.

Printing Help

You can print a specific Help topic, or you can print multiple Help topics (an entire section of online Help).

Table 1-2. How to Print Help Topics

To print	Do this
Current topic	Right-click within the help topic and choose **Print**.
Selected topic	On the **Contents** page: Right-click the topic  and choose **Print**.
Entire section of Help	On the **Contents** page: Right-click a book icon  or  and choose **Print**. Then choose **Print the selected heading and all subtopics**.

Tip: From the Help window's **Contents** page, click 

, located at the top of the window.

Bookmarking Frequently Used Help Topics

You can bookmark a topic in online Help just like you might bookmark a page in a book. This feature is also called setting up favorite places.

Each time you bookmark a Microsoft HTML Help topic, a record is recorded in the file, `HH.DAT`. This file not only records VisualDSP++ Help bookmarks, but also the bookmarks you place in other application Help systems that use `.CHM` files.

Once you have placed a bookmark onto a topic, you can view a list of bookmarked topics and quickly open one.

Placing a Bookmark at a Topic

1. Display the topic.
2. On the left side of the Help window, click the **Favorites** tab.
3. Click **Add**.

   The Help system adds the topic and displays it in the alphabetized list.

   You can remove a bookmark by selecting the name and clicking **Remove**.

Opening a Bookmarked Topic

1. On the left side of the Help window, click the **Favorites** tab.
2. Perform one of these actions:
   - Double-click the topic.
   - Select the topic and click **Display**.

Navigating in Online Help

To move around in the Help system, you can click the following.

- **A hyperlink within text.** The text is underlined and displayed in a color that is different from the regular black text.
- **A topic listed under a See Also heading.** The text is underlined and displayed in a color that is different from the regular black text.
- **A mini button or its associated text.** The button is a small gray square and the underlined text is in a different color.

- **A topic name on the Contents page** (Figure 1-4)

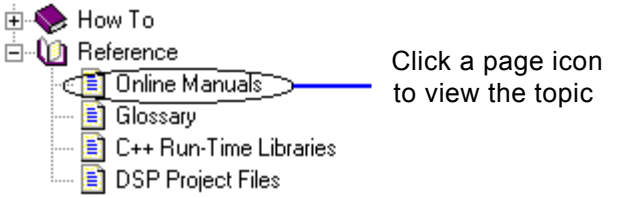

Figure 1-4. Contents Page – Online Manuals Topic

- **An index entry on the Index page** (Figure 1-5)

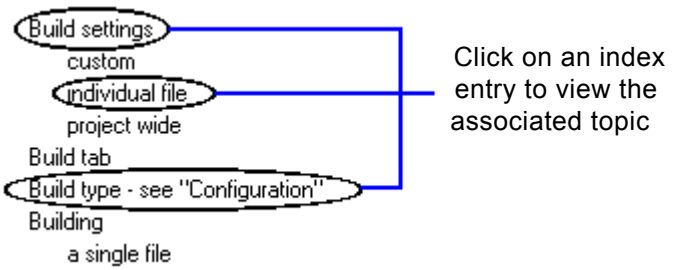

Figure 1-5. Index Entries on the Index Page

- **A topic name on the Search page.** The bottom portion of the **Search** page displays the located topics (hits) that include your search string.

Using the Search Features

VisualDSP++ Help provides both full-text and advanced search capabilities to help you find information.

Help System Search Rules

Different rules apply for each type of search.

Rules for Full-Text Searches

Observe these rules when formulating queries:

- Searches are not case-sensitive. You can type your search in uppercase or lowercase characters.

  You can search for any combination of letters (a–z) and numbers (0–9).

- Searches ignore punctuation marks such as the period, colon, semicolon, comma, and hyphen.

- Group the elements of your search by using double quotes or parentheses to set apart each element.

- You cannot search for quotation marks.

Note that if you are searching for a file name with an extension, group the entire string in double quotes, ("`filename.ext`"). Otherwise, the period breaks the file name into two separate terms. The default operation between terms is `AND`, which creates the logical equivalent to `filename AND ext`.

Rules for Advanced Searches

These rules apply to advanced searches:

- Expressions in parentheses are evaluated before the rest of the query.
- If a query does not contain a nested expression, it is evaluated from left to right. For example, "`folder NOT file OR project`" finds topics containing the word "folder" without the word "file," or topics containing the word "project." The expression "`folder NOT (file OR project)`", however, finds topics containing the word "folder" without either of the words "file" or "project."
- You cannot nest expressions deeper than five levels.

Full-Text Searches

The full-text search capability enables you to locate every occurrence of a text string within the Help system. You specify a particular word or phrase, and the search function finds only the topics that contain that word or phrase.

You can search previous results, match similar words, and search through the topic titles only.

A basic search consists of the word or phrase that you want to locate. You can use similar word matches, a previous results list, or topic titles to further define your search.

You can run an advanced search, which uses Boolean operators and wildcard expressions to further narrow the search criteria. Figure 1-6 on page 1-19 shows an example of a Boolean search for "`new AND plot`".

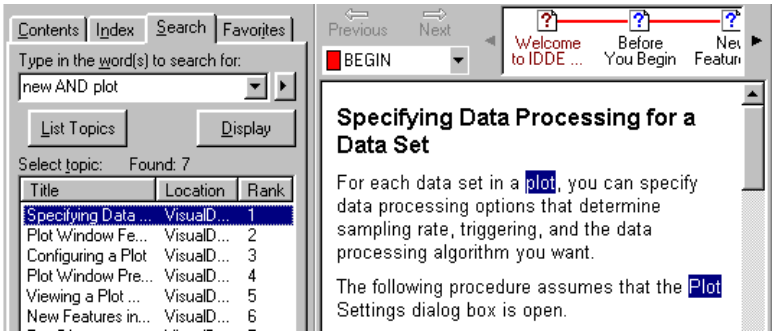

Figure 1-6. Boolean Search for “new AND plot”

To find information with full-text search:

1. Click the Help viewer’s **Search** tab.
2. In **Type in the word(s) to search for**, type the word or phrase you want to find.
3. Select **Search previous results** to narrow your search.
4. Select **Match similar words** to find words that are similar to the search string.
5. Select **Search titles only** to search only the topic titles.
6. Click the **Options** button 

at the top of the Help Viewer window to highlight all instances of search terms found in topic files. Then choose **Search Highlight On**.
7. Click **List Topics**, select the topic you want, and then click **Display**.

   Note that you can sort the topic list by clicking the **Title**, **Location**, or **Rank** column heading.

Advanced Search Techniques

You can use the following search techniques to narrow your searches for more precise results.

- Wildcard expressions
- Boolean operators
- Nested expressions

Using Wildcard Expressions

Wildcard expressions enable you to search for one or more characters by using a question mark or asterisk. Table 1-3 describes the results of these different kinds of searches.

Table 1-3. How to Use Wildcard Expressions to Define a Search

To find	Example	Results
A single word	project	Locates topics that contain the word "project." Other grammatical variations, such as "projects" are located.
A phrase	"project window" (note the quotation characters) project window	Locates topics that contain the literal phrase "project window" and all its grammatical variations. Without the quotation characters, the query is equivalent to specifying "project AND window," which finds topics containing both of the individual words, instead of the phrase.
Wildcard expressions	link* -or- .C??	Locates topics that contain the terms "linker," "linking," "links," and so on. The asterisk cannot be the only character in the term. Locates topics that contain the terms ".CPP" or ".CXX." The question mark cannot be the only character in the term.

Using Boolean Operators

Use the Boolean AND, OR, NOT, and NEAR operators to precisely define your search by creating a relationship between search terms.

Insert a Boolean operator by typing the operator (AND, OR, NEAR, or NOT) or by clicking the arrow button.

Note that if you do not specify an operator, AND is used. For example, the query `call stack` is equivalent to `call AND stack`.

Table 1-4 describes the results of using Boolean operators to define a search.

Table 1-4. How to Use Boolean Operators to Define a Search

To find	Example	Results
Both terms in the same topic	new AND plot	Locates topics that contain both the words “new” and “plot”
Either term in a topic	new OR plot	Locates topics that contain either the word “new” or the word “plot” or both
The first term without the second term	new NOT plot	Locates topics that contain the word “new”, but not the word “plot”
Both terms in the same topic, close together	new NEAR plot	Locates topics that contain the word “new” within eight words of the word “plot”

You cannot use the |, &, or ! characters as Boolean operators. You must use OR, AND, or NOT.

Using Nested Expressions

Use nested expressions to create complex searches for information. For example, `new AND ((plot OR waterfall) NEAR window)` finds topics containing the word “new” along with the words “plot” and “window” close together, or topics containing “new” along with the words “waterfall” and “window” close together.

Viewing Online Manuals

VisualDSP++ includes three types of user documentation.

Table 1-5. Types of User Documentation

Files	Purpose
`.CHM`	VisualDSP++ online Help system files and VisualDSP++ manuals are provided in Microsoft HTML Help format. Installing VisualDSP++ automatically copies these files to the `VisualDSP\Help` folder. Online Help is ideal for searching the entire tools manual set. Invoke Help from the VisualDSP++ **Help** menu or via the Windows **Start** button. The `.CHM` files require Internet Explorer 4.0 (or higher) or the installation of a component that provides a `.CHM` file viewer.
`.PDF`	Manuals and data sheets in Portable Documentation Format are located in the installation CD’s `Docs` folder. Viewing and printing a `.PDF` file requires a PDF reader, such as Adobe Acrobat Reader (4.0 or higher). Running `setup.exe` on the installation CD provides easy access to these documents. You can also copy PDF files from the installation CD onto another disk.
`.HTM` or `.HTML`	Dinkum Abridged C++ library and FlexLM network license manager software documentation is located on the installation CD in the `Docs\Reference` folder. Viewing or printing these files requires a browser, such as Internet Explorer 4.0 (or higher). You can copy these files from the installation CD onto another disk.

The VisualDSP++ software installation procedure does not copy PDF versions of books and data sheets or supplemental reference documentation to the VisualDSP directory.

Printing Online Documents

You can print documents from the VisualDSP++ Tools Installation CD-ROM.

To print online documents:

1. Insert the VisualDSP++ Tools Installation CD-ROM in your CD-ROM drive.
2. Open the **Docs** folder by using one of these options:
   - From the **VisualDSP++ Tools Installation** main menu, click **View Documentation**. (If the main menu does not appear, run `setup.exe`.)
   - In Windows Explorer, select your CD-ROM drive (for example, **d:**) and open the **Docs** folder.
3. Open the folder where the document is located.

   The `Data Sheets` folder contains copies of DSP data sheets.

   The `Hardware Manuals` folder contains copies of hardware manuals.

   The `Reference` folder includes the HTML files that comprise the Dinkum Abridged C++ library and the FlexLM network license documentation.

   The `Tools Manuals` folder contains copies of VisualDSP++ tools manuals.

4. Double-click the document that you want to print. Selecting a PDF file opens Adobe Acrobat Reader and displays the document. Selecting an HTML file opens a browser and displays the document.
5. From the **File** menu, choose **Print** and specify the pages that you want to print (and other print options).

2 TUTORIAL

This chapter contains the following topics.

Overview

This tutorial demonstrates key features and capabilities of the VisualDSP++ Integrated Development and Debugging Environment (IDDE). The exercises use sample programs written in C, C++, and assembly for Blackfin processors. The ADSP-BF535 Family Simulator is used for all exercises.

You can use different Blackfin processors with only minor changes to the Linker Description Files (`.LDFs`) included with each project.

VisualDSP++ includes basic Linker Description Files for each processor type in the `ldf folder`. The default installation path is:

```
Analog Devices\VisualDSP\Program Files
```

The source files for these exercises are installed during the VisualDSP++ software installation.

The tutorial contains five exercises:

- In **Exercise One**, you will start up VisualDSP++, build a project containing C source code, and profile the performance of a C function.
- In **Exercise Two**, you will create a new project, create a Linker Description File to link with the assembly routine, rebuild the project, and profile the performance of the assembly language routine.
- In **Exercise Three**, you will plot the various waveforms produced by a Finite Impulse Response (FIR) algorithm.
- In **Exercise Four**, you will use linear profiling to examine the efficiency of the FIR algorithm used in Exercise Three. Using the collected linear profile data, you will pinpoint the most time-consuming areas of the algorithm, which are likely to require hand tuning in the assembly language.
- In **Exercise Five**, you will install a VCSE component on your system and add the component to the project. Then you will build and run the program with the component.

Become familiar with the VisualDSP++ toolbar buttons, shown in Figure 2-1. They are shortcuts for menu commands such as **Open** a file and **Run** a program. Toolbar buttons and menu commands that are not available for tasks that you want to perform are disabled and displayed in gray.

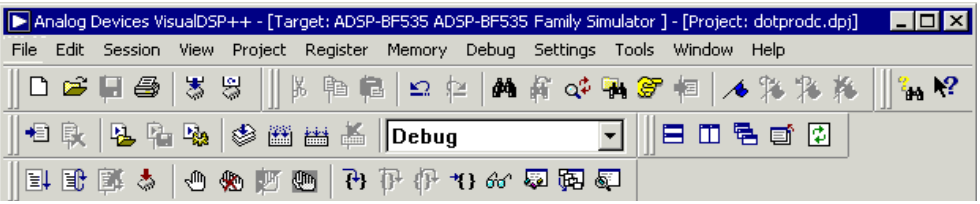

Figure 2-1. VisualDSP++ Toolbar Buttons

Exercise One: Building and Running a C Program

In this exercise, you will:

- Start up the VisualDSP++ environment
- Open and build an existing project
- Examine windows and dialog boxes
- Run the program

The sources for this exercise are in the `dot_product_c` folder. The default installation path is:

```
Program Files\Analog Devices\VisualDSP\Blackfin\Examples\
Tutorial\dot_product_c
```

Step 1: Start VisualDSP++ and Open a Project

To start VisualDSP++ and open a project:

1. Click the Windows **Start** button and select **Programs**, **VisualDSP**, and **VisualDSP++ Environment**.

   If you are running VisualDSP++ for the first time, the **New Session** dialog box (Figure 2-6 on page 2-11) opens to enable you to set up a session.

   a. Select the values shown in Table 2-1.

Table 2-1. Session Specification

Box	Value
Debug Target	ADSP-BF535 Family Simulator
Platform	ADSP-BF535 Family Simulator
Session Name	ADSP-BF535 ADSP-BF535 Family Simulator
Processor	ADSP-BF535

   b. Click **OK**. The VisualDSP++ main window appears.

   If you have already run VisualDSP++ and the **Reload last project at startup** option is selected on the **Project** page under **Settings** and **Preferences**, VisualDSP++ opens the last project that you worked on. To close this project, choose **Close** from the **Project** menu, and then click **No** when prompted to save the project. Since you have made no changes to the project, you do not have to save it.

2. From the **Project** menu, choose **Open**.

   VisualDSP++ displays the **Open Project** dialog box.

3. In the **Look in** box, open the `Program Files\Analog Devices` folder and double-click the following subfolders in succession.

   `VisualDSP\Blackfin\Examples\Tutorial\dot_product_c`

This path is based on the default installation.

4. Double-click the `dotprodc` project (`.dpj`) file.

   VisualDSP++ loads the project in the **Project** window, as shown in Figure 2-2. The environment displays messages in the **Output** window as it processes the project settings and file dependencies.

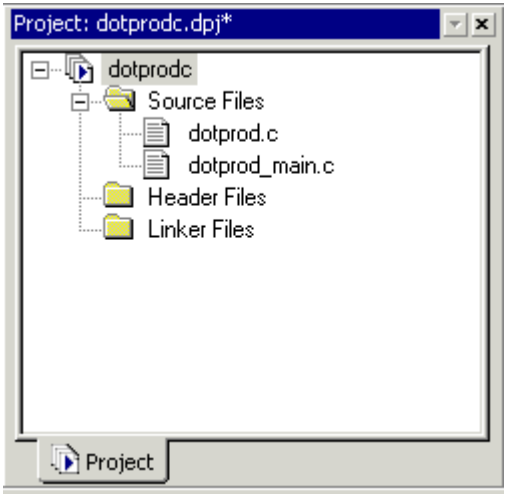

Figure 2-2. Project Loaded in the Project Window

The first time that you open projects installed from the software kit, VisualDSP++ may detect that files, folders, or both have moved. If you receive a "`Project has been moved`" message, click **OK** to continue.

The `dotprodc` project comprises two C language source files, `dotprod.c` and `dotprod_main.c`, which define the arrays and calculate their dot products.

5. From the **Settings** menu, choose **Preferences** to open the **Preferences** dialog box, shown in Figure 2-3.

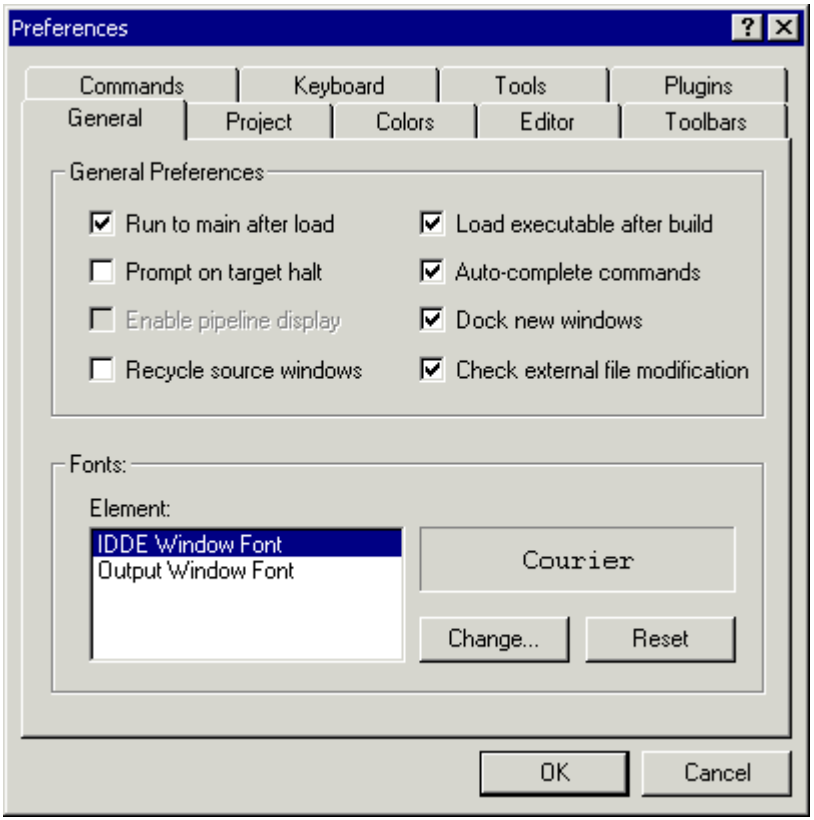

Figure 2-3. Preferences Dialog Box

6. On the **General** page, under **General Preferences**, make sure that the following options are selected.
   - **Run to main after load**
   - **Load executable after build**
7. Click **OK** to close the **Preferences** dialog box.

   The VisualDSP++ main window appears. You are now ready to build the project.

Step 2: Build the dotprodc Project

To build the `dotprodc` project:

1. From the **Project** menu, choose **Build Project**.

   VisualDSP++ first checks and updates the project dependencies and then builds the project by using the project source files.

   As the build progresses, the **Output** window displays status messages (error and informational) from the tools. For example, when a tool detects invalid syntax or a missing reference, the tool reports the error in the **Output** window.

   If you double-click the file name in the error message, VisualDSP++ opens the source file in an editor window. You can then edit the source to correct the error, rebuild, and launch the debug session. If the project build is up-to-date (the files, dependencies, and options have not changed since the last project build), no build is performed unless you run the **Rebuild All** command. Instead, you see the message "`Project is up to date`." If the build has no errors, a message reports "`Build completed successfully`."

In this example (Figure 2-4) notice that the compiler detects an undefined identifier and issues the following error in the **Output** window.

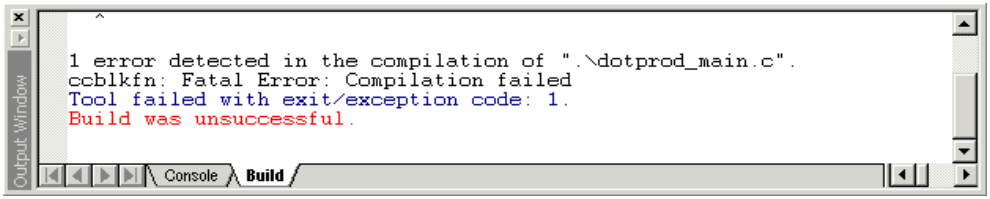

Figure 2-4. Example of Error Message

2. Double-click the error message text in the **Output** window.

   VisualDSP++ opens the C source file `dotprod_main.c` in an editor window and places the cursor on the line that contains the error (see Figure 2-5 on page 2-9).

   The editor window in Figure 2-5 shows that the integer variable declaration `int` has been misspelled as `itn`.

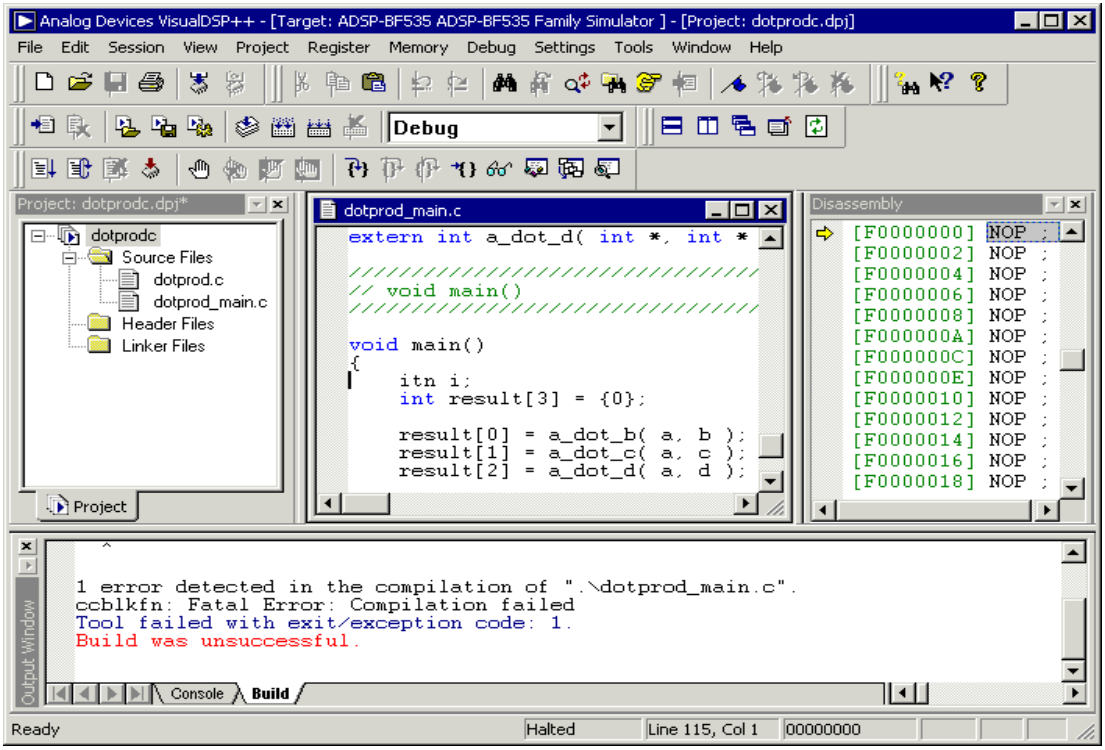

Figure 2-5. Output Window and Editor Window

3. In the editor window, click on `itn` and change it to `int`. Notice that `int` is now color coded to signify that it is a valid C keyword.
4. Save the source file by choosing **Save** from the **File** menu.
5. Build the project again by choosing **Build Project** from the **Project** menu. The project is now built without any errors, as reported on the **Build** page in the **Output** window.

Now that you have built your project successfully, you can run the example program.

Step 3: Run the Program

In this procedure, you will:

- Set up the debug session before running the program
- View debugger windows and dialog boxes

Since you enabled **Load executable after build** on the **General** page in the **Preferences** dialog box, the executable file `dotprodc.dxe` is automatically downloaded to the target. If the debug session's processor does not match the project's build target, VisualDSP++ reports this discrepancy and asks if you want to select another session before downloading the executable to the target. If VisualDSP++ does not open the **Session List** dialog box, skip steps 1–4.

To set up the debug session:

1. **In the Session List** dialog box, click **New Session** to open the **New Session** dialog box, shown in Figure 2-6.

   For subsequent debugging sessions, use the **New Session** command on the **Sessions** menu to open the **New Session dialog box**.

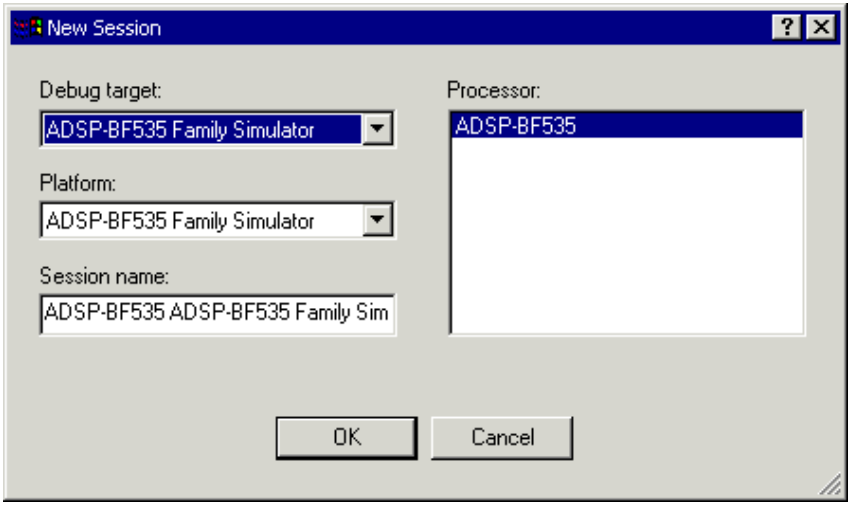

Figure 2-6. New Session Dialog Box

2. Specify the target and processor information listed in the Table 2-2.

Table 2-2. Session Specification

Box	Value
Debug Target	ADSP-BF535 Family Simulator
Platform	ADSP-BF535 Family Simulator
Session Name	ADSP-BF535 ADSP-BF535 Family Simulator
Processor	ADSP-BF535

3. Click **OK** to close the **New Session** dialog box and return to the **Session List** dialog box.
4. With the new session name highlighted, click **Activate**.

If you do not click **Activate**, the session mismatch message appears again.

VisualDSP++ closes the **Session List** dialog box, automatically loads your project's executable file (`dotprodc.dxe`), and advances to the main function of your code (see Figure 2-7).

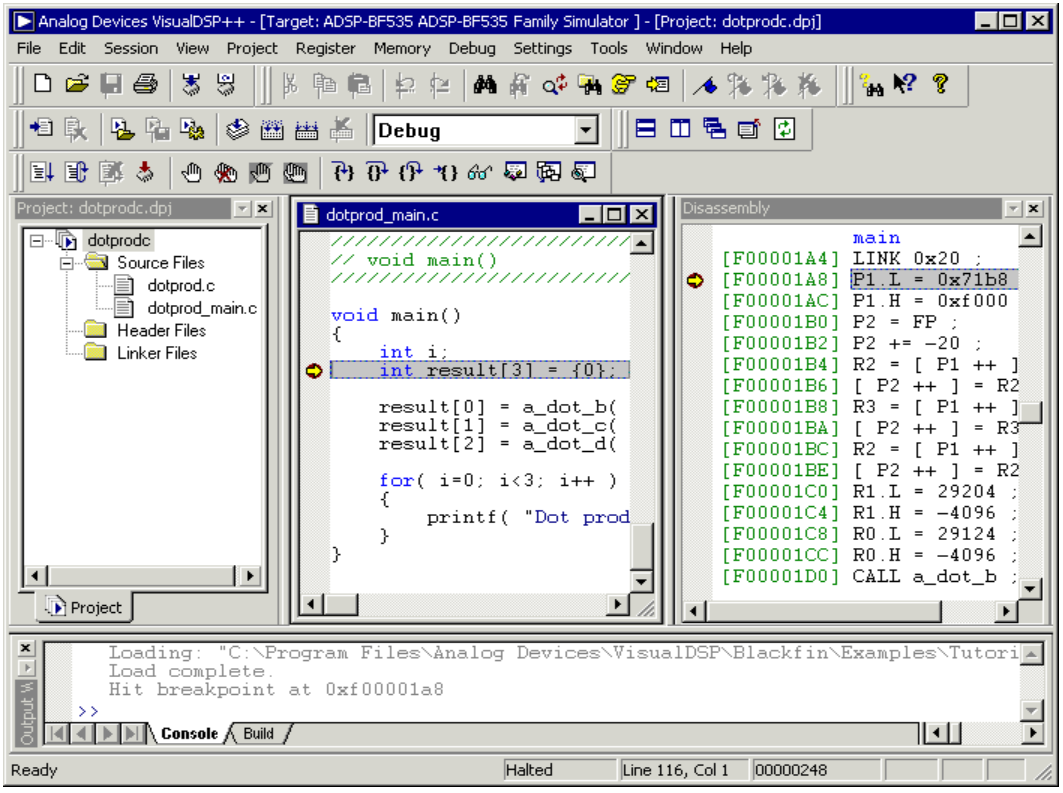

Figure 2-7. Loading dotprodc.dxe

5. Look at the information in the open windows.

   The **Output** window's Console page contains messages about the status of the debug session. In this case, VisualDSP++ reports that the `dotprodc.dxe` load is complete.

   The **Disassembly** window displays the machine code for the executable. Use the scroll bars to move around the **Disassembly** window.

   Note that a solid red circle and a yellow arrow appear at the start of the program labeled "`main`". The solid red circle ● indicates that a breakpoint is set on that instruction, and the yellow arrow ➪ indicates that the processor is currently halted at that instruction. When VisualDSP++ loads your C program, it automatically sets two breakpoints, one at the beginning and one at the end of code execution.

6. From the **Settings** menu, choose **Breakpoints** to view the breakpoints set in your program. VisualDSP++ displays the **Breakpoints** dialog box, shown in Figure 2-8 on page 2-14.

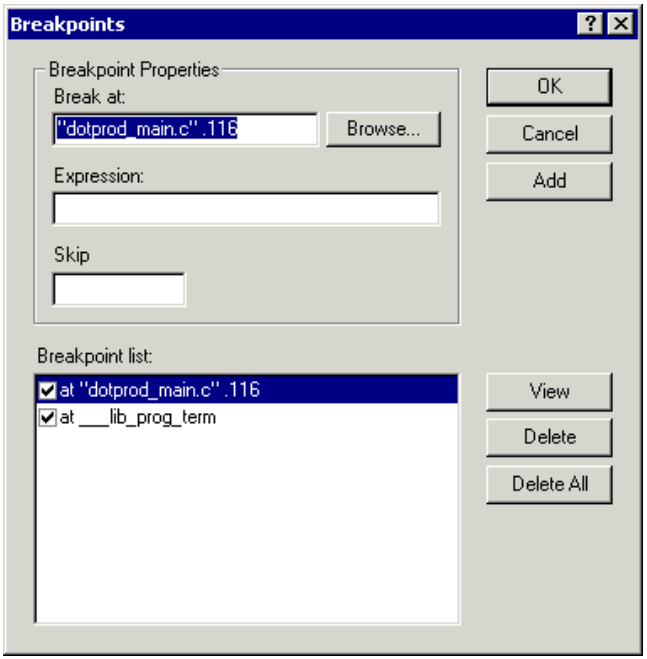

Figure 2-8. Breakpoints Dialog Box

The breakpoints are set at these C program labels:

- `at “dotprod_main.c” 116`
- `at __lib_prog_term`

The **Breakpoints** dialog box enables you to view, add, and delete breakpoints and to browse for symbols.In the **Disassembly** and editor windows, double-clicking on a line of code toggles (adds or deletes) breakpoints. In the editor window, however, you must place the cursor in the gutter before double-clicking.

Use these tool buttons to set or clear breakpoints:

 Toggles a breakpoint for the current line

 Clears all breakpoints

7. Click **OK** or **Cancel** to exit the **Breakpoints** dialog box.

Step 4: Run dotprodc

To run `dotprodc`, click the **Run** button  or choose **Run** from the **Debug** menu.

VisualDSP++ computes the dot products and displays the following results on the **Console** page (Figure 2-9) in the **Output** window.

```
Dot product [0] = 13273595
Dot product [1] = -49956078
Dot product [2] = 35872518
```

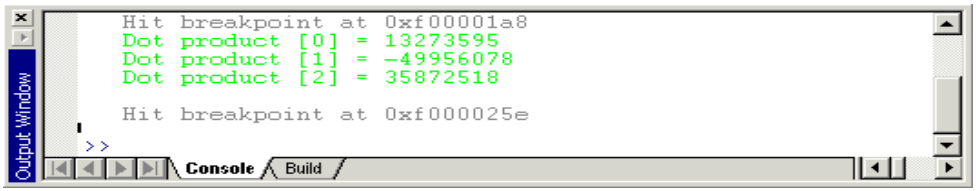

Figure 2-9. Results of the dotprodc Program

You are now ready to begin Exercise Two.

Exercise Two: Modifying a C Program to Call an Assembly Routine

In Exercise One, you built and ran a C program. In this exercise, you will modify this program to call an assembly language routine, create a Linker Description File to link with the assembly routine, and rebuild the project. The project files are largely identical to those of Exercise One. Minor modifications illustrate the changes needed to call an assembly language routine from C source code.

Step 1: Create a New Project

To create a new project:

1. From the **Project** menu, choose **Close** to close the dotprodc project. Click **Yes** when prompted to close all open editor windows. If you have modified your project during this session, you are prompted to save the project. Click **No**.
2. From the **Project** menu, choose **New** to open the **Save New Project As** dialog box, shown in Figure 2-10.

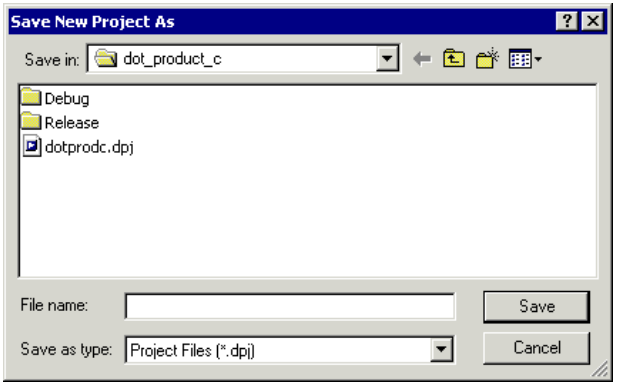

Figure 2-10. Save New Project As Dialog Box

3. Click the up-one-level button  until you locate the `dot_product_asm` folder, and then double-click this folder.
4. In the **File name** box, type `dot_product_asm`, and click **Save**.

   The **Project Options** dialog box (Figure 2-11) appears.

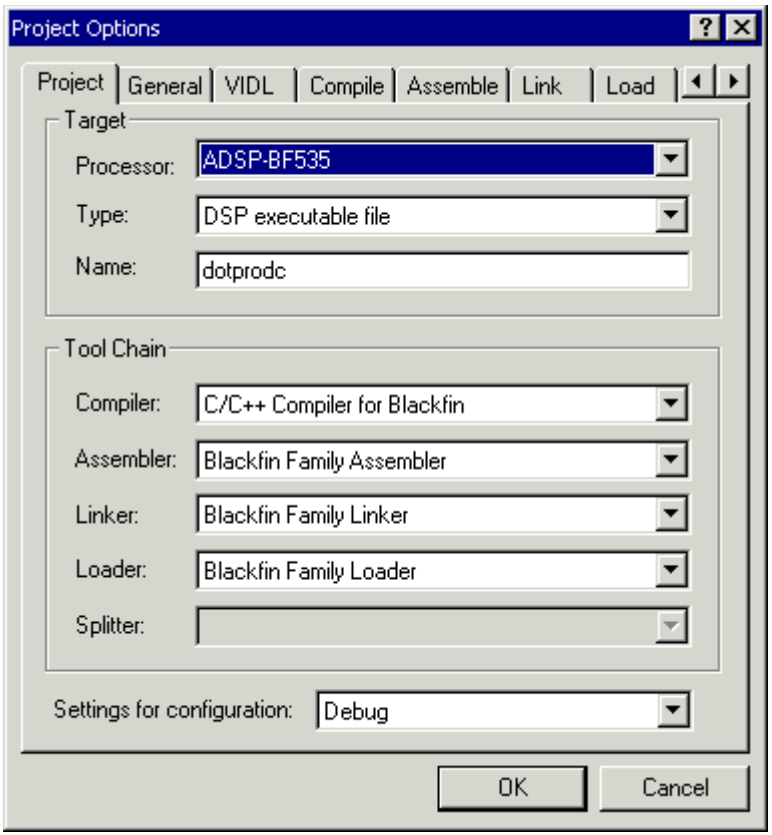

Figure 2-11. Project Options Dialog Box: Project Page

This dialog box enables you to specify project build information.

5. Take a moment to view the tabbed pages in the **Project Options** window: **Project**, **General**, **Compile**, **Assemble**, **Link**, **Load**, **Compiled Simulation**, and **Post Build**. On each page, you specify the tool options used to build the project.

6. On the **Project** page (Figure 2-11 on page 2-17), specify the values shown in Table 2-3.

Table 2-3. Completing the Project Page

Box	Value
Processor	ADSP-BF535
Type	DSP executable file
Name	dot_product_asm
Settings for configuration	Debug

These settings specify information for building an executable file for the ADSP-BF535 processor. The executable contains debug information, so you can examine program execution.

7. Click the **Compile** tab to display the **Compile** page, shown in Figure 2-12 on page 2-19.

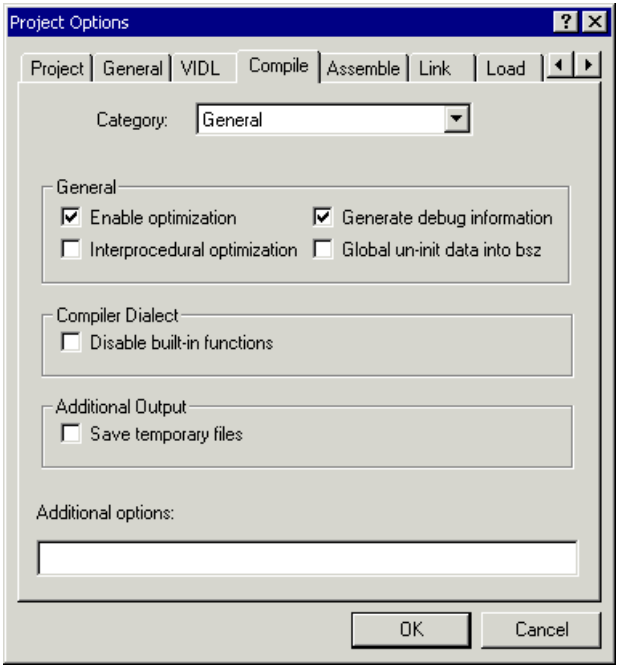

Figure 2-12. Project Options Dialog Box: Compile Page

8. Complete the **General** group box as follows.

    a. Select the **Enable optimization** check box to enable optimization.

    b. Select the **Generate debug information** check box, if it is not already selected, to enable debug information for the C source.

    These settings direct the C compiler to optimize code for the ADSP-BF535 processor. Because the optimization takes advantage of DSP architecture and assembly language features, some of the C debug information is not saved. Debugging is, therefore, performed through debug information at the assembly language level.

9. Click **OK** to apply changes to the project options and to close the **Project Options** dialog box. When prompted to add support for the VisualDSP++ kernel, click **No**.

You are now ready to add the source files to the project.

Step 2: Add Source Files to dot_product_asm

To add the source files to the new project:

1. Click the **Add File** button  , or from the **Project** menu, choose **Add to Project**, and then choose **File(s)**.

   The **Add Files** dialog box (Figure 2-13) appears.

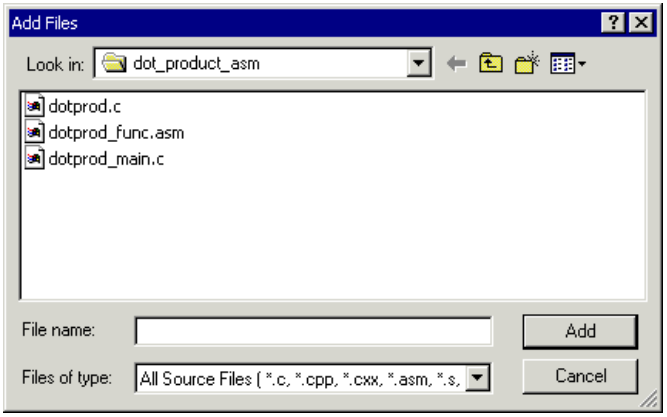

Figure 2-13. Add Files Dialog Box: Adding Source Files to the Project

2. In the **Look in** box, locate the project folder, `dot_product_asm`.
3. In the **Files of type** box, select **All Source Files** from the drop-down list.
4. Hold down the **Ctrl** key and click `dotprod.c` and `dotprod_main.c`. Then click **Add**.

To display the files that you added in step 4, open the `Source Files` folder in the **Project** window.

You are now ready to create a Linker Description File for the project.

Step 3: Create a Linker Description File for the Project

In this procedure, you will use the Expert Linker to create a Linker Description File for the project.

To create a Linker Description File:

1. From the **Tools** menu, choose **Expert Linker** and then choose **Create LDF** to open the **Create LDF Wizard**, shown in Figure 2-14 on page 2-22.

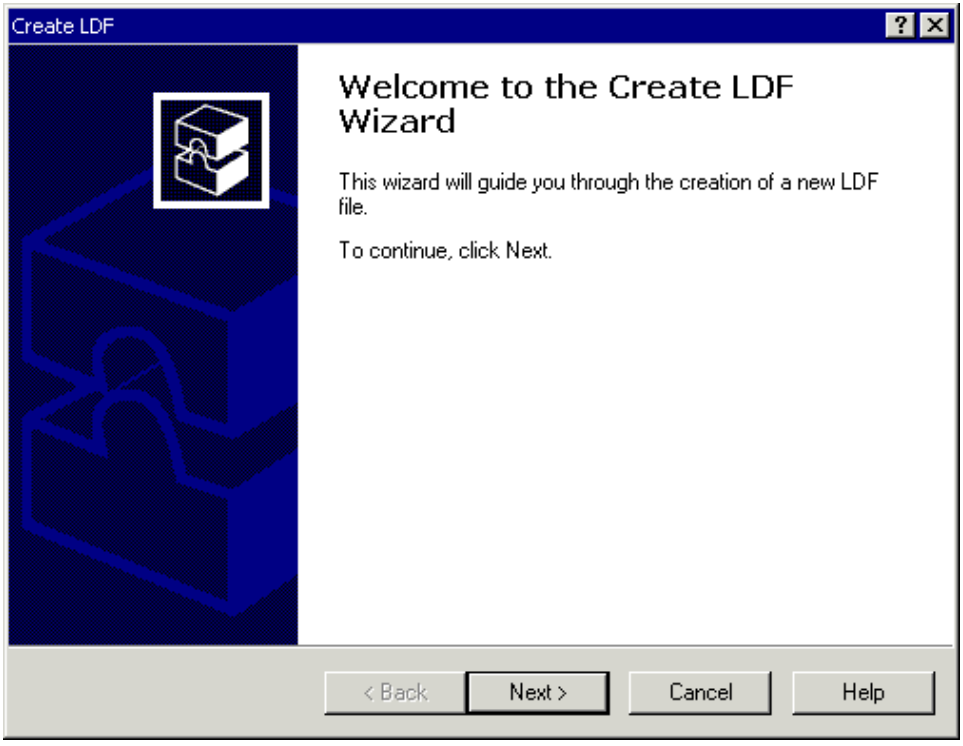

Figure 2-14. Create LDF Wizard

2. Click **Next** to display the **Create LDF – Step 1 of 3** page, shown in Figure 2-15.

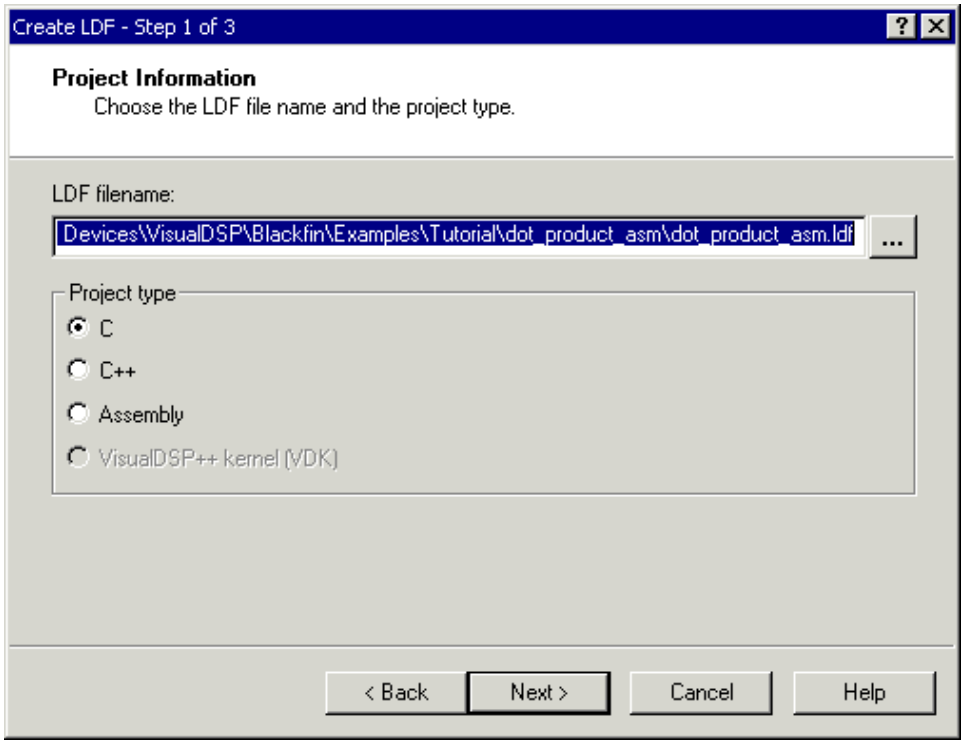

Figure 2-15. Create LDF – Step 1 of 3 Page

This page enables you to assign the **LDF file name** (based on the project name) and to select the **Project type**.

3. Accept the values selected for your project and click **Next** to display the **Create LDF – Step 2 of 3** page, shown in Figure 2-16.

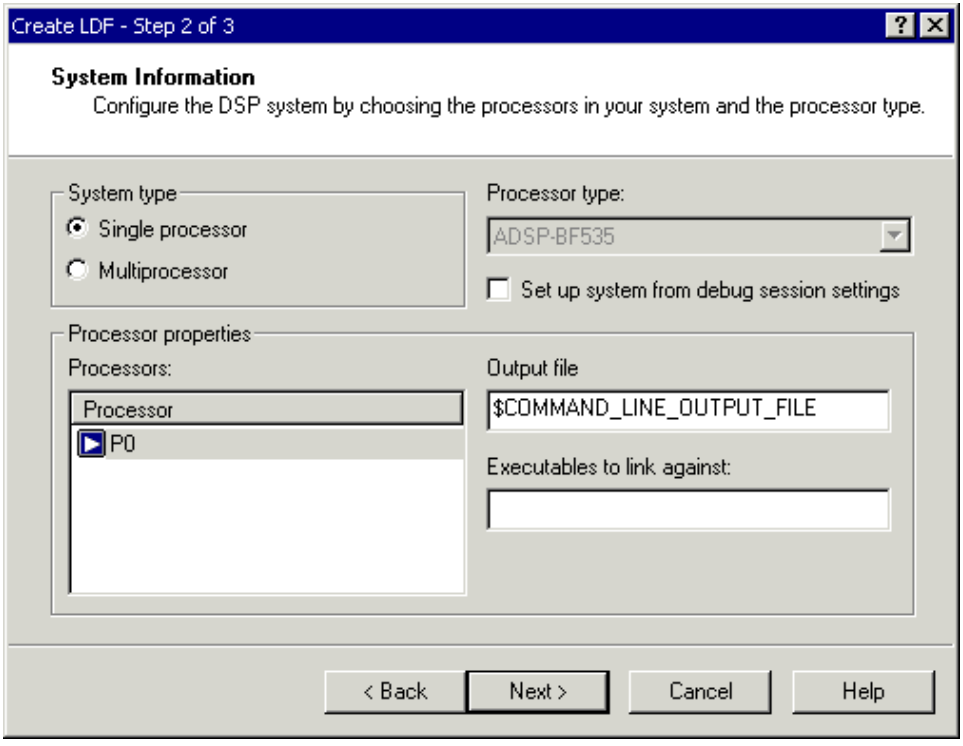

Figure 2-16. Create LDF – Step 2 of 3 Page

This page enables you to set the **System type** (defaulted to **Single processor**), the **Processor type** (defaulted to **ADSP-BF535** to match the project), and the name of the linker **Output file** (defaulted to the name selected by the project).

4. Accept the default values and click **Next** to display the **Create LDF – Step 3 of 3** page, shown in Figure 2-17.

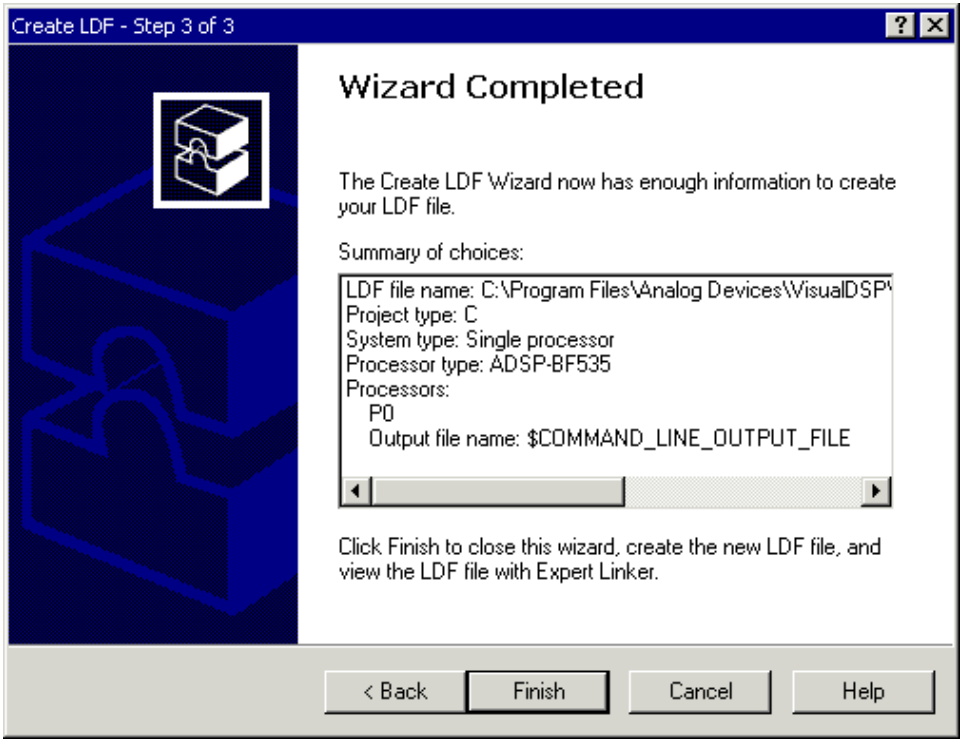

Figure 2-17. Create LDF – Step 3 of 3 Page

5. Review the **Summary of choices** and click **Finish** to create the `.LDF` file.

   You now have a new `.LDF` file in your project. The new file is in the **Linker Files** folder in the **Project** window.

   The **Expert Linker** window opens with a representation of the `.LDF` file that you created. This file is complete for this project. Close the **Expert Linker** window.

6. Click the **Build Project** button  to build the project. The C source file opens in an editor window, and execution halts.

The C version of the project is now complete. You are now ready to modify the sources to call the assembly function.

Step 4: Modify the Project Source Files

In this procedure, you will:

- Modify `dotprod_main.c` to call `a_dot_c_asm` instead of `a_dot_c`
- Save the modified file

To modify `dotprod_main.c` to call the assembly function:

1. Resize or maximize the editor window for better viewing.
2. From the **Edit** menu, choose **Find** to open the **Find** dialog box, shown in Figure 2-18.

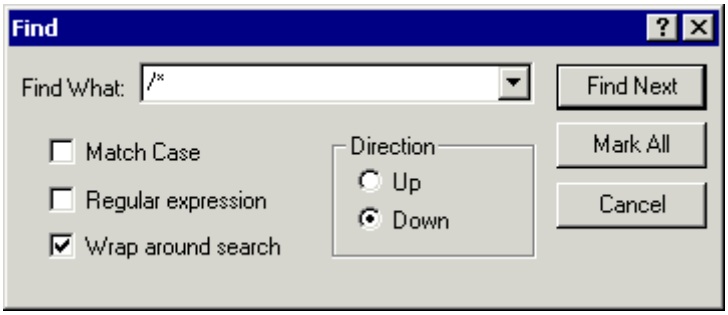

Figure 2-18. Find Dialog Box: Locating Occurrences of /*

3. In the **Find What** box, type `/*`, and then click **Mark All**.

   The editor bookmarks all lines containing `/*` and positions the cursor at the first instance of `/*` in the `extern int a_dot_c_asm` declaration.

4. Select the comment characters `/*` and use the **Ctrl+X** key combination to cut the comment characters from the beginning of the `a_dot_c_asm` declaration. Then move the cursor up one line and use the **Ctrl+V** key combination to paste the comment characters at the beginning of the `a_dot_c` declaration. Because syntax coloring is turned on, the code will change color as you cut and paste the comment characters.

   Repeat this step for the end-of-comment characters `*/` at the end of the `a_dot_c_asm` declaration. The `a_dot_c` declaration is now fully commented out, and the `a_dot_c_asm` declaration is no longer commented.

5. Press **F3** to move to the next bookmark.

   The editor positions the cursor on the `/*` in the function call to `a_dot_c_asm`, which is currently commented out. Note that the previous line is the function call to the `a_dot_c` routine.

6. Press **Ctrl+X** to cut the comment characters from the beginning of the function call to `a_dot_c_asm`. Then move the cursor up one line and press **Ctrl+V** to paste the comment characters at the beginning of the call to `a_dot_c`.

   Repeat this step for the end-of-comment characters `*/`. The `main()` function should now be calling the `a_dot_c_asm` routine instead of the `a_dot_c` function, previously called in Exercise One.

   Figure 2-19 on page 2-29 shows the changes made in step 6.

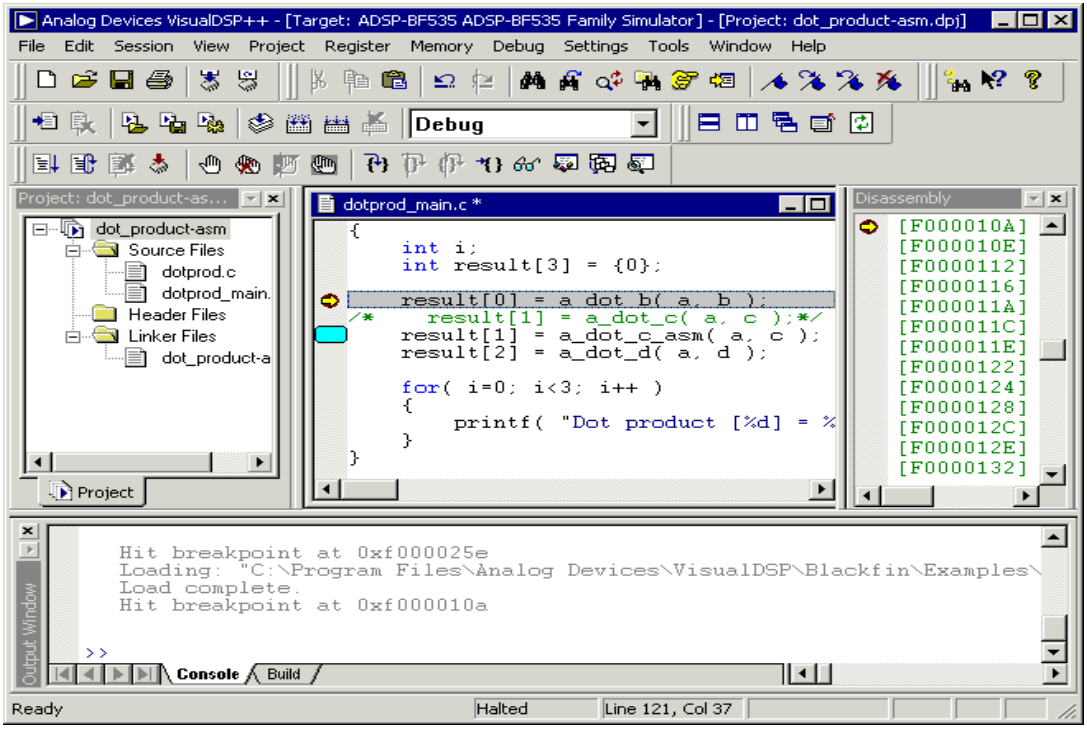

Figure 2-19. Editor Window: Modifying dotprod_main.c to Call a_dot_c_asm

7. From the **File** menu, choose **Save** to save the changes to the file.
8. Place the cursor in the editor window. Then, from the **File** menu, choose **Close** to close the `dotprod_main.c` file.

You are now ready to modify `dotprodasm.ldf`.

Step 5: Use the Expert Linker to modify dot_prod_asm.ldf

In this procedure you will:

- View the Expert Linker representation of the `.LDF` file that you created
- Modify the `.LDF` file to map in the section for the `a_dot_c_asm` assembly routine

To examine and then modify `dot_prod_asm.ldf` to link with the assembly function:

1. Click the **Add File** button  .
2. Select `dotprod_func.asm` and click **Add**.
3. Try to build the project by performing one of these actions:
   - Click the **Build Project** button  .
   - From the **Project** menu, choose **Build Project**.

     Notice the linker error in the **Output** window, shown in Figure 2-20.

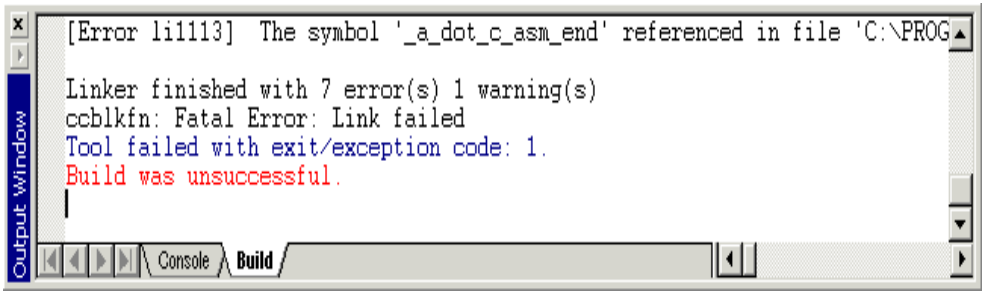

Figure 2-20. Output Window: Linker Error

4. In the **Project** window, double-click in the `dot_prod_asm.ldf` file. The **Expert Linker** window (Figure 2-21) opens with a representation of your file.

   The left pane contains a list of the **Input Sections** that are in your project or are mapped in the `.LDF` file. A red `X` is over the icon in front of the section named "`my_asm_section`" because the Expert Linker has determined that the section is not mapped by the `.LDF` file.

   The right pane contains a graphical representation of the memory segments that the Expert Linker defined when it created the `.LDF` file. Change the view mode by right-clicking in the right pane and choosing **View Mode**. Then choose **Memory Map Tree** to display the tree view shown in Figure 2-21.

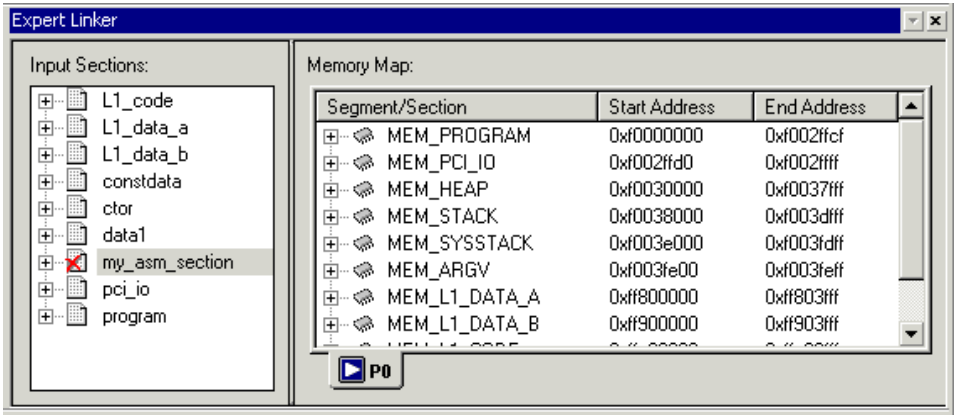

Figure 2-21. Expert Linker Window

5. Map the section `my_asm_section` into the memory segment named `MEM_PROGRAM` as follows.

   Open the my_asm_section input section by clicking on the plus sign in front of it. The input section expands to show that the linker macro `$OBJECTS` and the object file `dotprod_func.doj` both have a section that has not been mapped. Drag the icon in front of `$OBJECTS` to the memory map pane and onto the `MEM_PROGRAM`.

   As shown in Figure 2-22, the red `X` should no longer appear because the section `my_asm_section` has been mapped.

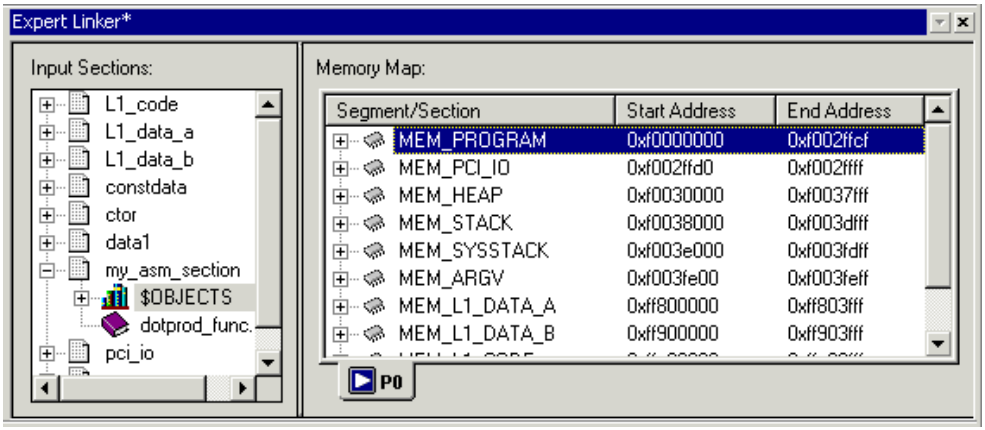

Figure 2-22. Dragging $OBJECTS onto MEM_PROGRAM

6. From the **Tools** menu, choose **Expert Linker** and **Save** to save the modified file. Then close the **Expert Linker** window.

   If you forget to save the file and then rebuild the project, VisualDSP++ will see that you modified the file and will automatically save it.

You are now ready to rebuild and run the modified project.

Step 6: Rebuild and Run dot_product_asm

To run `dot_product`:

1. Build the project by performing one of these actions:
   - Click the **Build Project** button  .
   - From the **Project** menu, choose **Build Project**.

   At the end of the build, the **Output** window displays the following message on the **Build** page.

   "`Build completed successfully.`"

   VisualDSP++ loads the program, runs to main, and displays the **Output**, **Disassembly**, and editor windows (shown in Figure 2-23 on page 2-34).

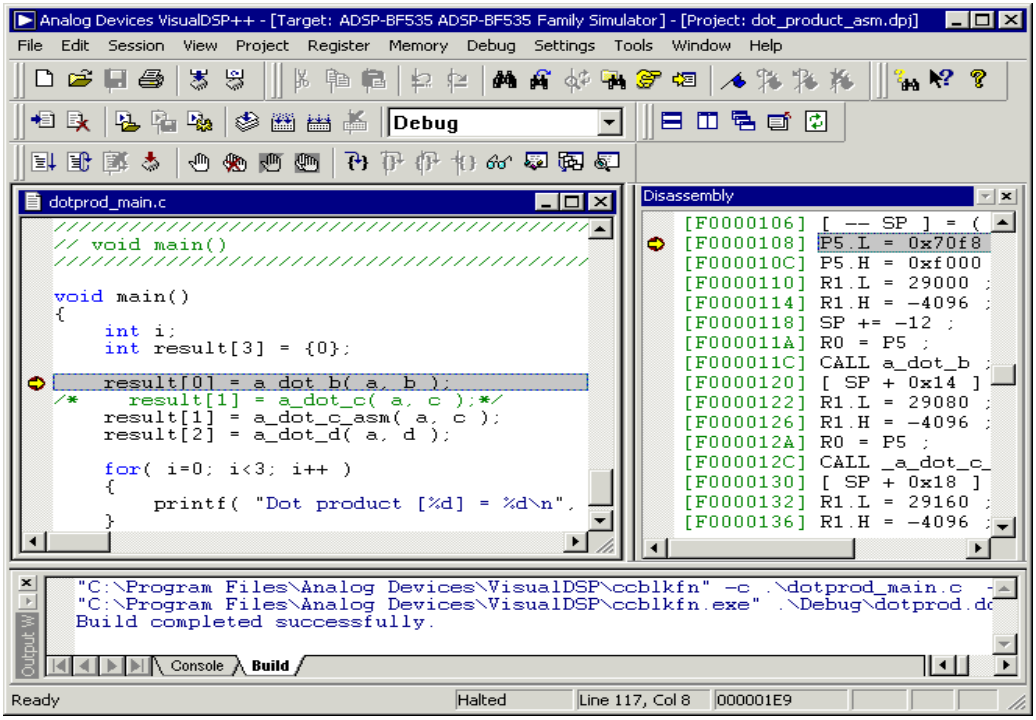

Figure 2-23. Windows Left Open from the Previous Debugger Session

2. Click the **Run** button  to run `dot_product_asm`.

   The program calculates the three dot products and displays the results on the **Console** page in the **Output** window. When the program stops running, the message "`Halted`" appears in the status bar at the bottom of the window. The results, shown below, are identical to the results obtained in Exercise One.

```
Dot product [0] = 13273595
Dot product [1] = -49956078
Dot product [2] = 35872518
```

You are now ready to begin Exercise Three.

Exercise Three: Plotting Data

In this exercise, you will load and debug a pre-built program that applies a simple Finite Impulse Response (FIR) filter to a buffer of data. You will use VisualDSP++'s plotting engine to view the different data arrays graphically, both before and after running the program.

Step 1: Load the FIR Program

To load the FIR program:

1. Keep the **Disassembly** window and **Console** page (in the **Output** window) open, but close all other windows.
2. From the **File** menu, choose **Load Program** or click  . The **Open a Processor Program** dialog box appears.
3. Select the FIR program to load as follows.
    a. Open your `Analog Devices` folder and double-click the `VisualDSP\Blackfin\Examples\Tutorial\fir\Debug` subfolder.
    b. Double-click `FIR.DXE` to load the program in an editor window (Figure 2-24 on page 2-36).

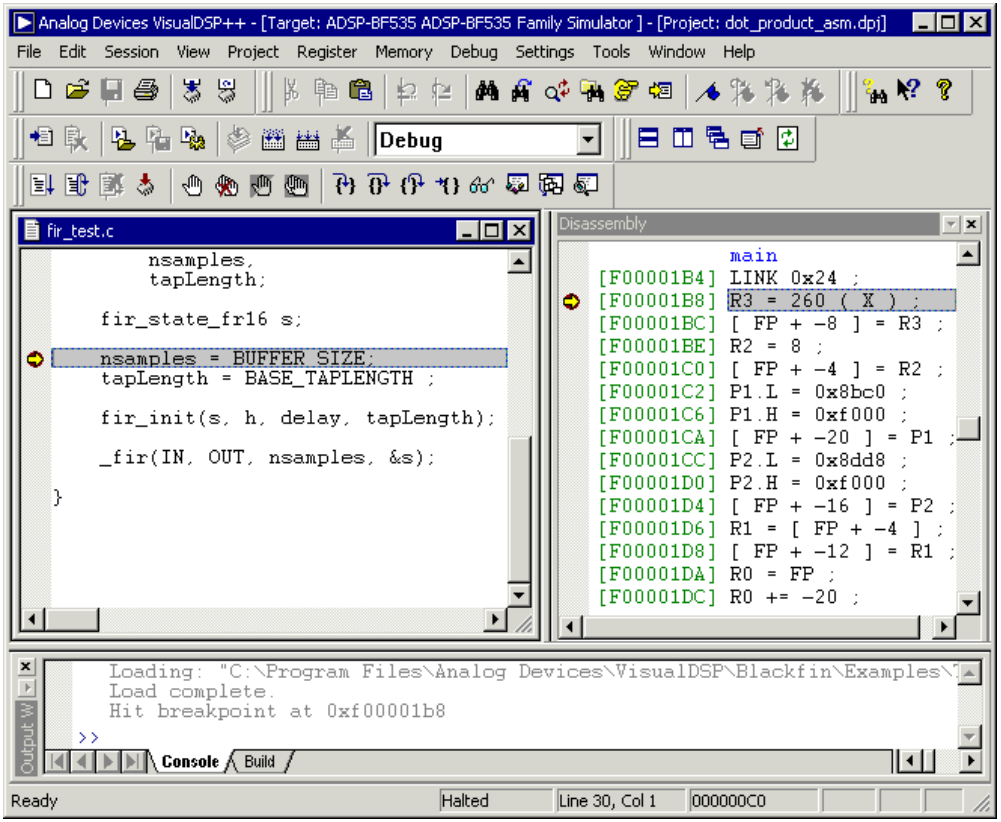

Figure 2-24. Loading the FIR Program

4. Look at the source code of the FIR program.

   You can see two global data arrays:

   - `IN`
   - `OUT`

   You can also see one function, `fir`, that operates on these arrays.

You are now ready to open a plot window.

Step 2: Open a Plot Window

To open a plot window:

1. From the **View** menu, choose **Debug Windows** and **Plot**. Then choose **New** to open the **Plot Configuration** dialog box, shown in Figure 2-25.

   Here you will add the data sets that you want to view in a plot window.

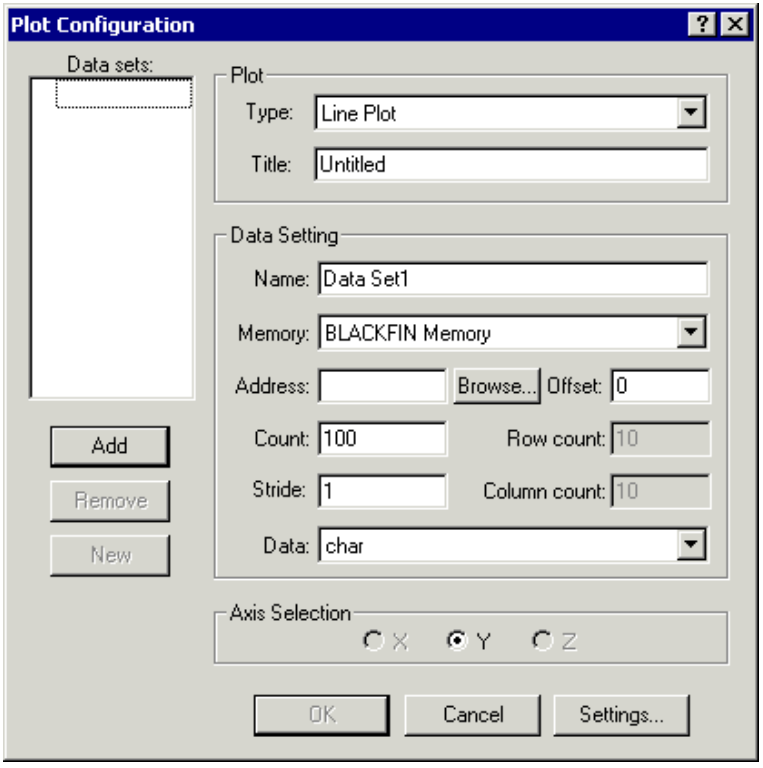

Figure 2-25. Plot Configuration Dialog Box

2. In the **Plot** group box, specify the following values.
   - In the **Type** box, select **Line Plot** from the drop-down list.
   - In the **Title** box, type `fir`.
3. Enter two data sets to plot by using the values in Table 2-4.

Table 2-4. Two Data Sets: Input and Output

Box	Input Data Set	Output Data Set	Description
Name	Input	Output	Data set
Memory	BLACKFIN Memory	BLACKFIN Memory	Data memory
Address	IN	OUT	The address of this data set is that of the Input or Output array. Click **Browse** to select the value from the list of loaded symbols.
Count	128	128	The array is 260 elements long, but you are plotting the first 128 elements.
Stride	1	1	The data is contiguous in memory.
Data	short	short	Input and Output are arrays of int values.

After entering each data set, click **Add** to add the data set to the **Data Sets** list, as shown in Figure 2-26 on page 2-39. The **Plot Configuration** dialog box should now look like the one in Figure 2-26.

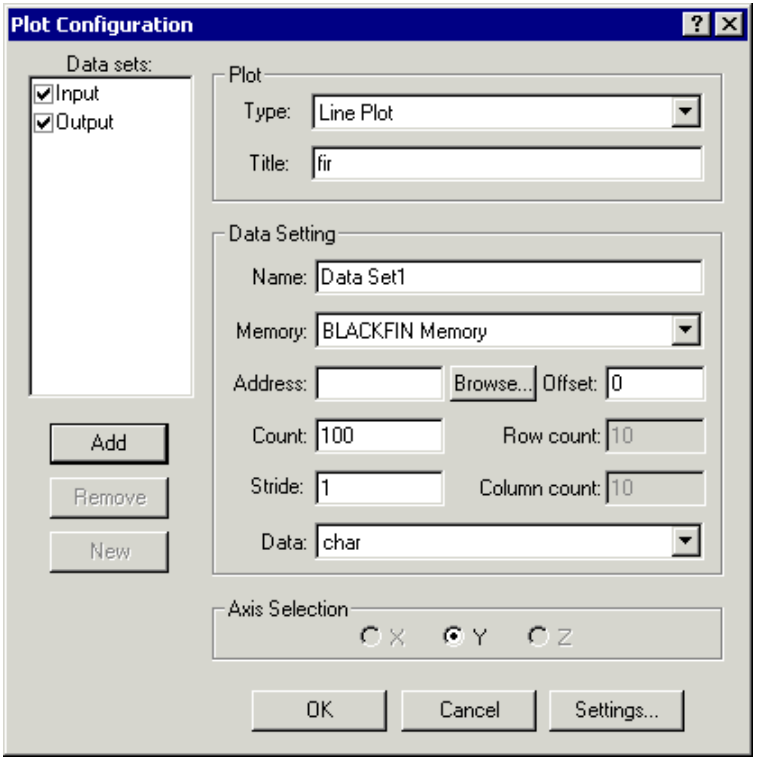

Figure 2-26. Plot Configuration Dialog Box with Input/Output Data Sets

4. Click **OK** to apply the changes and to open a plot window with these data sets.

   The plot window now displays the two arrays. Since, by default, the simulator initializes memory to zero, the Output data set appears as one horizontal line, shown in Figure 2-27 on page 2-40.

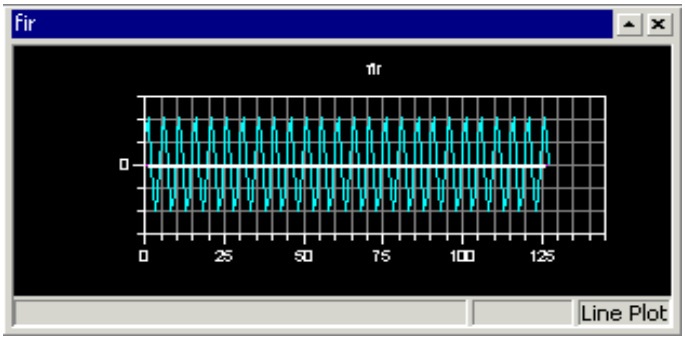

Figure 2-27. Plot Window: Before Running the FIR Program

5. Right-click in the plot window and choose **Modify Settings**. On the **General** page, in the **Options** group box, select **Legend** to display the legend box (shown in Figure 2-28 on page 2-41).

Step 3: Run the FIR Program and View the Data

To run the FIR program and view the data:

1. Press **F5** or click the **Run** button  to run to the end of the program.

   When the program halts, you see the results of the FIR filter in the Output array. The two data sets are visible in the plot window, as shown in Figure 2-28 on page 2-41.

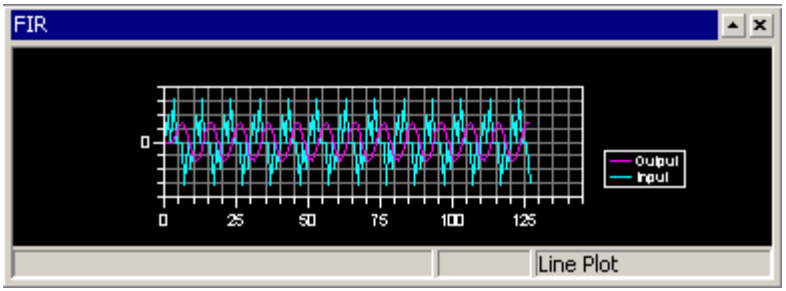

Figure 2-28. Plot Window After Running the FIR Program to Completion

Next you will zoom in on a particular region of interest in the plot window to focus in on the data.

2. Click the left mouse button inside the plot window and drag the mouse to create a rectangle to zoom into. Then release the mouse button to magnify the selected region.

   Figure 2-29 shows the selected region, and Figure 2-30 on page 2-42 shows the magnified result.

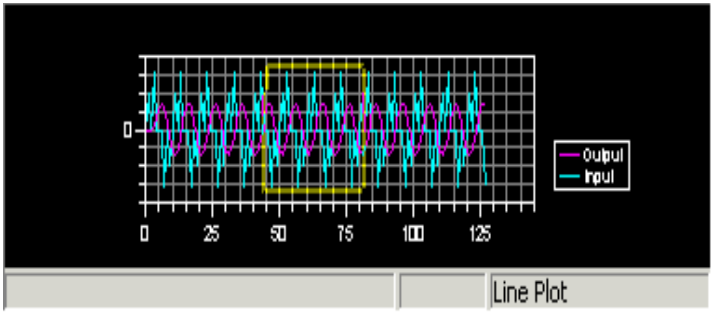

Figure 2-29. Plot Window: Selecting a Region to Magnify

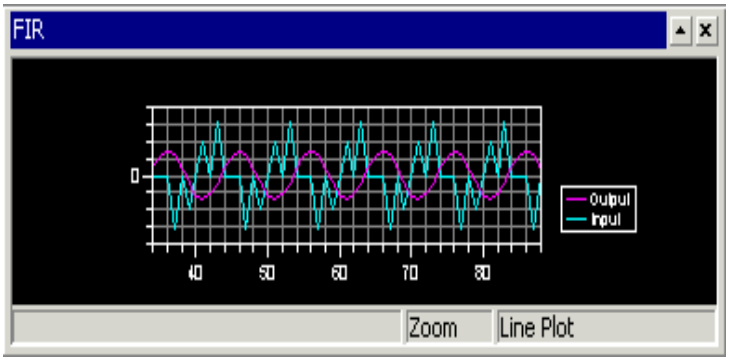

Figure 2-30. Plot Window: Magnified Result

To return to the previous view (before magnification), right-click in the plot window and choose **Reset Zoom** from the pop-up menu. You can view individual data points in the plot window by enabling the data cursor, as explained in the next step.

3. Right-click inside the plot window and choose **Data Cursor** from the popup menu. Move to each data point in the current data set by pressing and holding the Left (←) or Right (→) arrow key on the keyboard. To switch data sets, press the Up (↑) or Down (↓) arrow key. The value of the current data point appears in the lower-left corner of the plot window, as shown in Figure 2-31.

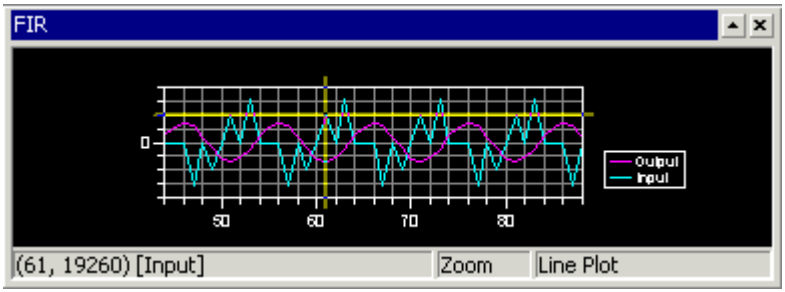

Figure 2-31. Plot Window: Using the Data Cursor Feature

4. Right-click in the plot window and choose **Data Cursor** from the pop-up menu. In the next step, you will look at data sets in the frequency domain.
5. Right-click in the plot window and choose **Modify Settings** to open the **Plot Settings** dialog box.
6. Complete these steps:
    a. Click the **Data Processing** tab to display the **Data Processing** page, shown in Figure 2-32.

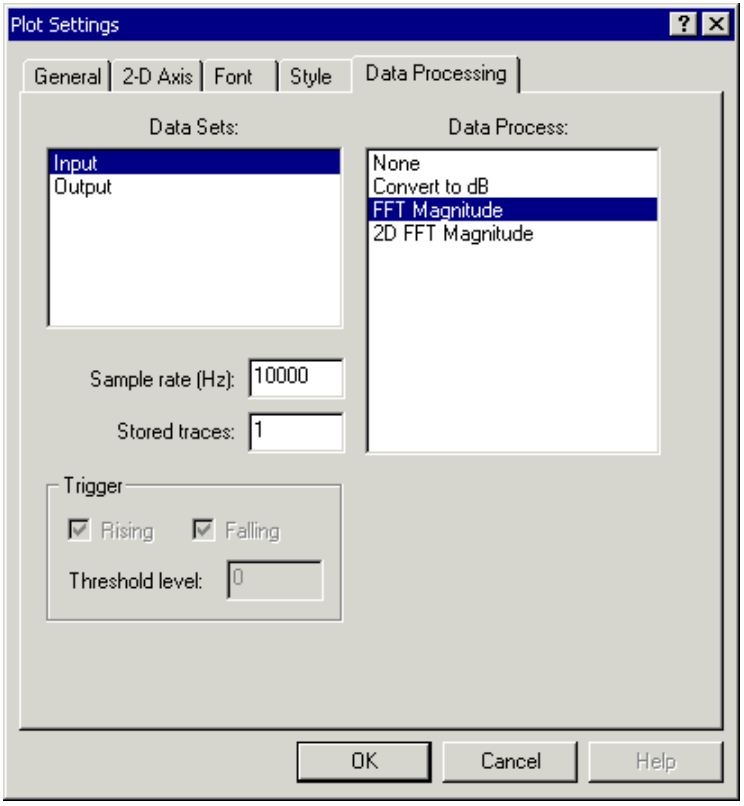

Figure 2-32. Data Processing Page

b. In the **Data Sets** box, make sure that **Input** (the default) is selected. In the **Data Process** box, choose **FFT Magnitude**.
c. In the **Sample rate (Hz)** box, type `10000`.
d. In the **Data Sets** box, select **Output**. In the **Data Process** box, choose **FFT Magnitude**
e. Click **OK** to exit the **Data Processing** page.

VisualDSP++ performs a Fast Fourier Transform (FFT) on the selected data set before it is plotted. FFT enables you to view the signal in the frequency domain, as shown in Figure 2-33.

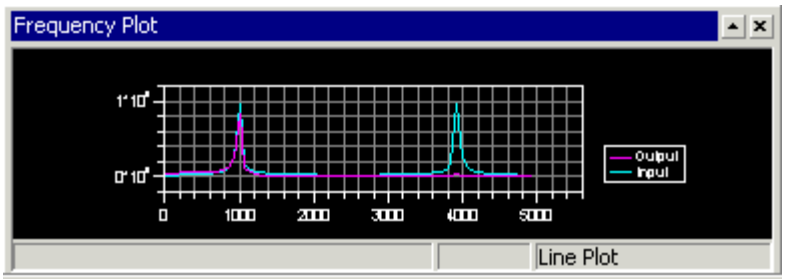

Figure 2-33. FFT Performed on a Selected Data Set

Now, complete the following steps to look at the FIR filter's response in the frequency domain.

1. From the **View** menu, choose **Debug Windows** and **Plot**. Then choose **New** to open the **Plot Configuration** dialog box.
2. Set up the Filter Frequency Response plot by completing the **Plot** and **Data Setting** group boxes as shown in Figure 2-34 on page 2-45.

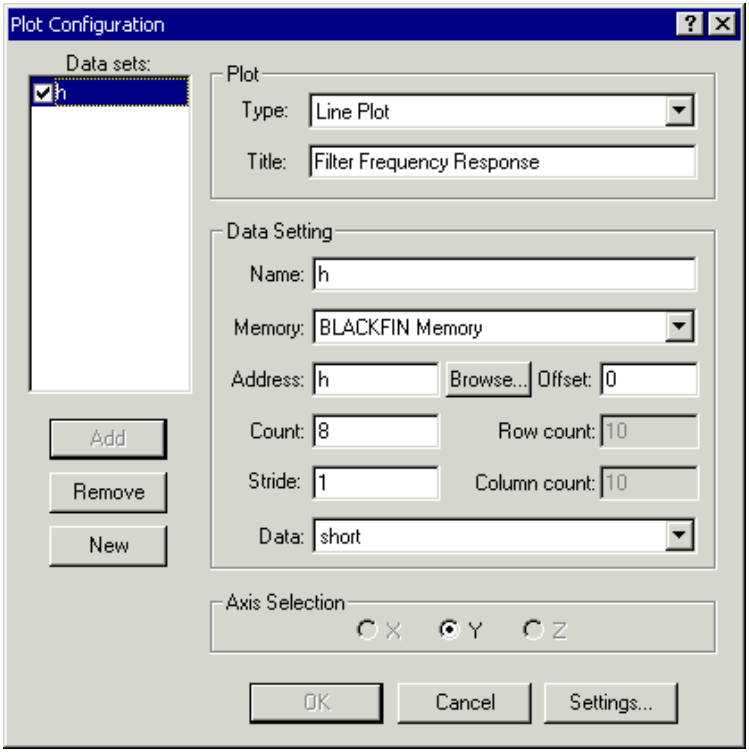

Figure 2-34. Filter Frequency Response Data Set

3. Click **Add** to add the data set to the **Data sets** box.
4. Click **OK** to apply the changes and to open the plot window with this data set.
5. Right-click in the plot window and choose **Modify Settings** to open the **Plot Settings** dialog box.

6. Click the **Data Processing** tab to display the **Data Processing** page, shown in Figure 2-32 on page 2-43. Complete this page as follows.

    a. In the **Data Sets** box, select **h**.

    b. In the **Data Process** box, choose **FFT Magnitude**.

    c. In the **Sample rate (Hz)** box, type **10000**.

    d. Click **OK** to exit the **Data Processing** page.

    VisualDSP++ performs a Fast Fourier Transform (FFT) on the selected data set, and enables you to view the filter response plot in the *frequency* domain, as shown in Figure 2-35.

    This plot shows that the FIR filter is a low pass filter that cuts off all frequency components above 4,000 Hz. When you apply a low pass filter to the input signal, the resulting signal has no output above 4,000 Hz.

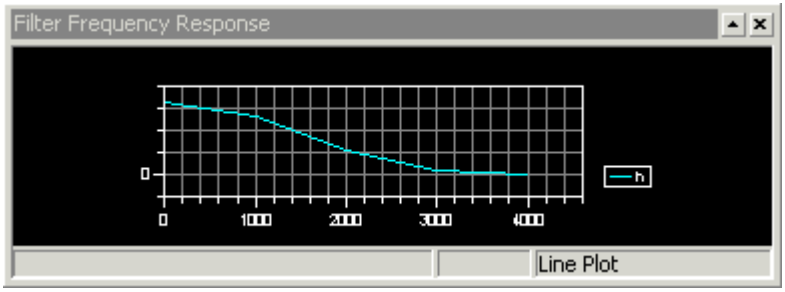

Figure 2-35. Filter Frequency Response Plot

You are now ready to begin Exercise Four.

Exercise Four: Linear Profiling

In this exercise, you will load and debug the FIR program from the previous exercise. You will then use linear profiling to evaluate the program's efficiency and to determine where the application is spending the majority of its execution time in the code.

VisualDSP++ supports two types of profiling: linear and statistical.

- You use linear profiling with a simulator. The count in the **Linear Profiling Results** window is incremented every time a line of code is executed.
- You use statistical profiling with a JTAG emulator connected to a processor target. The count in the **Statistical Profiling Results** window is based on random sampling.

Step 1: Load the FIR Program

To load the FIR program:

1. Close all open windows except for the **Disassembly** window and the **Output** window.
2. From the **File** menu, choose **Load Program**, or click  . The **Open a Processor Program** dialog box appears.
3. Select the program to load as follows.
    a. Open the `Analog Devices` folder and double-click the `VisualDSP\Blackfin\Examples\Tutorial\fir\Debug` subfolder.
    b. Double-click `FIR.DXE` to load and run the FIR program. VisualDSP++ opens an editor window.

You are now ready to set up linear profiling.

Step 2: Open the Profiling Window

To open the linear profiling window:

1. From the **Tools** menu, choose **Linear Profiling** and then choose **New Profile**.

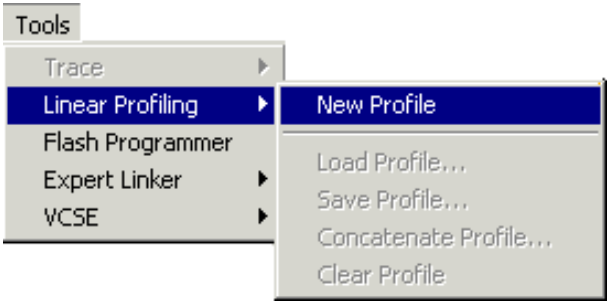

Figure 2-36. Setting Up Linear Profiling for the FIR Program

The **Linear Profiling Results** window opens.

2. For a better view of the data, use the window's title bar to drag and dock the window to the top of the VisualDSP++ main window, as shown in Figure 2-37.

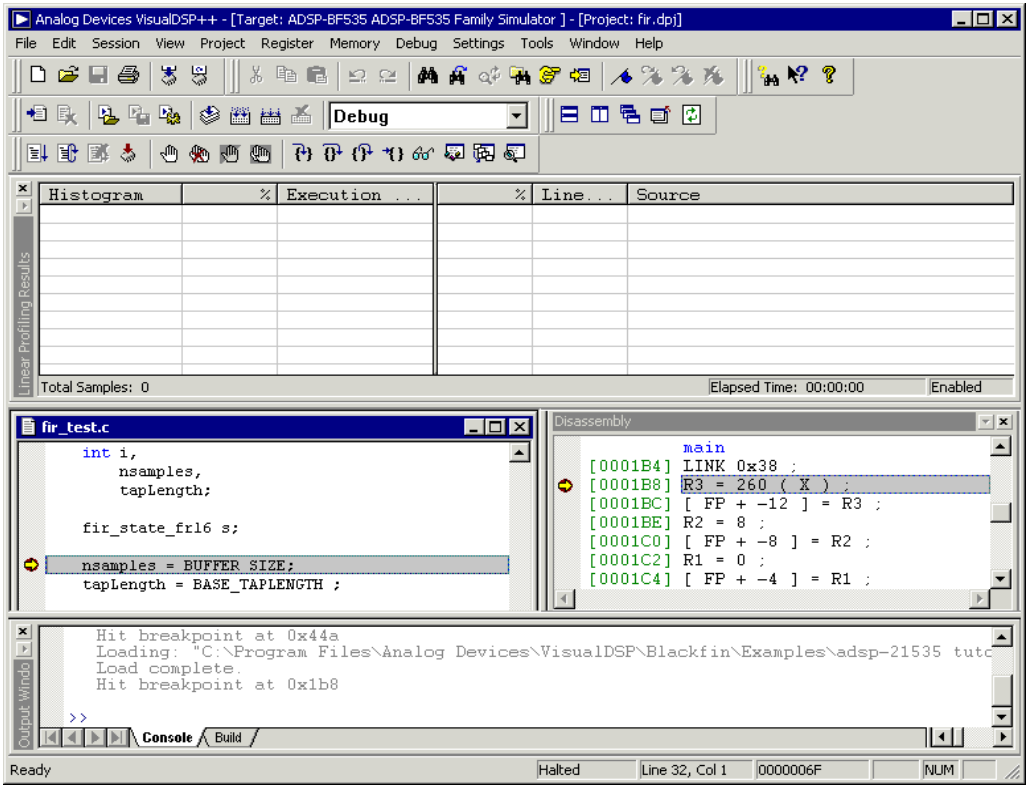

Figure 2-37. Linear Profiling Results Window (Empty)

The **Linear Profiling Results** window is intially empty. Linear profiling will be performed when you run the FIR program. After you run the program and collect data, this window displays the results of the profiling session.

You are now ready to collect and examine linear profile data.

Step 3: Collect and Examine the Linear Profile Data

To collect and examine the linear profile data:

1. Press **F5** or click  to run to the end of the program.

   When the program halts, the results of the linear profile appear in the **Linear Profiling Results** window.

2. Examine the results of your linear profiling session.

   The **Linear Profiling Results** window is divided into two, three-column panes. The left pane shows the results of the profile data. Double-clicking on a line in the left pane displays the corresponding source code for the profile data in the right pane, as shown in Figure 2-38.

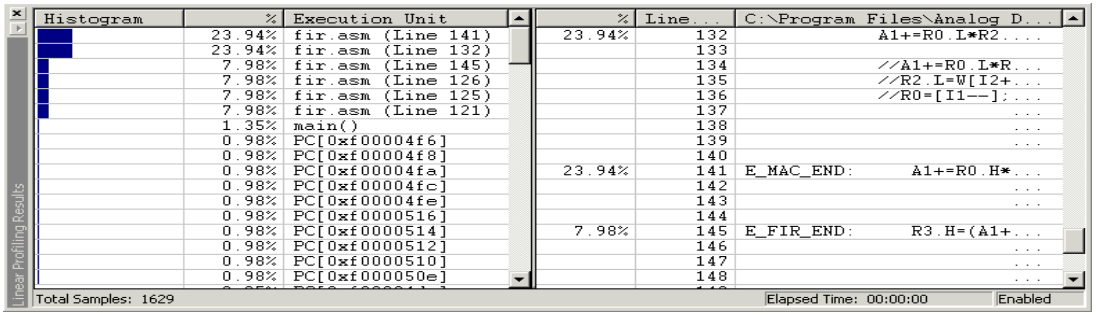

Figure 2-38. Linear Profiling Results of Analyzing the Performance of the FIR Program

The field values in the left pane are defined as follows.

Histogram	A graphical representation of the percentage of time spent in a particular execution unit. This percentage is based on the total time that the program spent running, so longer bars denote more time spent in a particular execution unit.The **Linear Profiling Results** window always sorts the data with the most time-consuming (expensive) execution units at the top.
%	The numerical percent of the same data found in the Histogram column. You can view this value as an absolute number of samples by right-clicking in the **Linear Profiling Results** window and by selecting **View Sample Count** from the popup menu.
Execution Unit	The program location to which the samples belong. If the instructions are inside a C function or a C++ method, the execution unit is the name of the function or method. For instructions that have no corresponding symbolic names, such as hand-coded assembly or source files compiled without debugging information, this value is an address in the form of `PC[xxx]`, where `xxx` is the address of the instruction.

If the instructions are part of an assembly file, the execution unit is the assembly file followed by the line number in parentheses.

In Figure 2-38 on page 2-50 the left pane shows that lines 141 and 132 in the `fir.asm` file have each consumed over 23% of the total execution time. If you double-click one of these lines, the right (source) pane displays `fir.asm` and jumps to that line. The source pane displays data for each line of executable code in the file for which linear profile data has been collected. Figure 2-39 shows the linear profile data for `fir.asm`.

%	Line...	C:\Program Files\Analog D...
23.94%	132	A1+=R0.L*R2....
	133	
	134	//A1+=R0.L*R...
	135	//R2.L=W[I2+...
	136	//R0=[I1--];...
	137	...
	138	...
	139	...
	140	
23.94%	141	E_MAC_END: A1+=R0.H*...
	142	...
	143	...
	144	
7.98%	145	E_FIR_END: R3.H=(A1+...
	146	...
	147	...
	148	...

Figure 2-39. Linear Profile Data for fir.asm

The details of `fir.asm` show that lines 132 and 141 each spend 23.94% of the time in the inner loop of the FIR routine.

You are now ready to begin Exercise Five.

Exercise Five: Installing and Using a VCSE Component

In this exercise, you will complete the following tasks.

- Start up the VisualDSP++ environment
- Open an existing project
- Install a VCSE component on your system
- Add the component to the project
- Build and run the program with the component

The sources for the exercise are in the `vcse_component` folder. The default installation path is:

```
Program files\Analog Devices\VisualDSP\Blackfin\Examples\
Tutorial\vcse_component
```

Step 1: Start VisualDSP++ and Open the Project

To start VisualDSP++ and open the project:

1. Click the Windows **Start** button and select **Programs**, **VisualDSP**, and **VisualDSP++ Environment**.

   The VisualDSP++ main window appears.

   If you have already run VisualDSP++ and the **Reload last project at startup** option is selected in the **Project Options** dialog box, VisualDSP++ opens the last project that you worked on.

   To close this project, choose **Close** from the **Project** menu and then click **No** when prompted to save the project. Since you have made no changes to the project, you do not have to save it.

2. From the **Project** menu, choose **Open**.

   VisualDSP++ displays the **Open Project** dialog box.

3. In the **Look in** box, open the `Program Files\Analog Devices` folder and double click the following sub-folders in succession.

   `VisualDSP\Blackfin\Examples\Tutorial\vcse_component`

This path is based on the default installation.

4. Double-click the `useg711.dpj` project file.

   VisualDSP++ loads the project. The environment displays messages in the **Output** window as it processes the project settings.

The first time that you open projects installed from the software kit, VisualDSP++ may detect that files, folders, or both have moved. If you receive a "`Project has been moved`" message, click **OK** to continue.

The useg711 project contains a single C language source file `useg711.c`, which contains the code needed to create an instance of the CULawc component and to invoke the methods of the IG711 interface.

Step 2: Install the EXAMPLES::CULawc Component on Your System

The EXAMPLES::CULawc component is distributed as part of VisualDSP++ and is ready to be installed on your system:

1. From the **Tools** menu, select the **VCSE** submenu and then choose **Manage Components**.
2. In the **Display** field, select **Downloaded component package…** from the drop-down list.

   The **Open** dialog box is displayed.
3. In the **Look in** box, open the `Program Files\Analog Devices` folder and double-click the following subfolders in succession.

   `VisualDSP\Blackfin\Examples\Tutorial\vcse_component`

This path is based on the default installation.

4. Double-click the `examples_culawc_BF535.vcp` file.

   VisualDSP++ opens the file, extracts the information about the component, and shows it as a downloaded component in the **Component Manager** dialog box (Figure 2-40 on page 2-56).

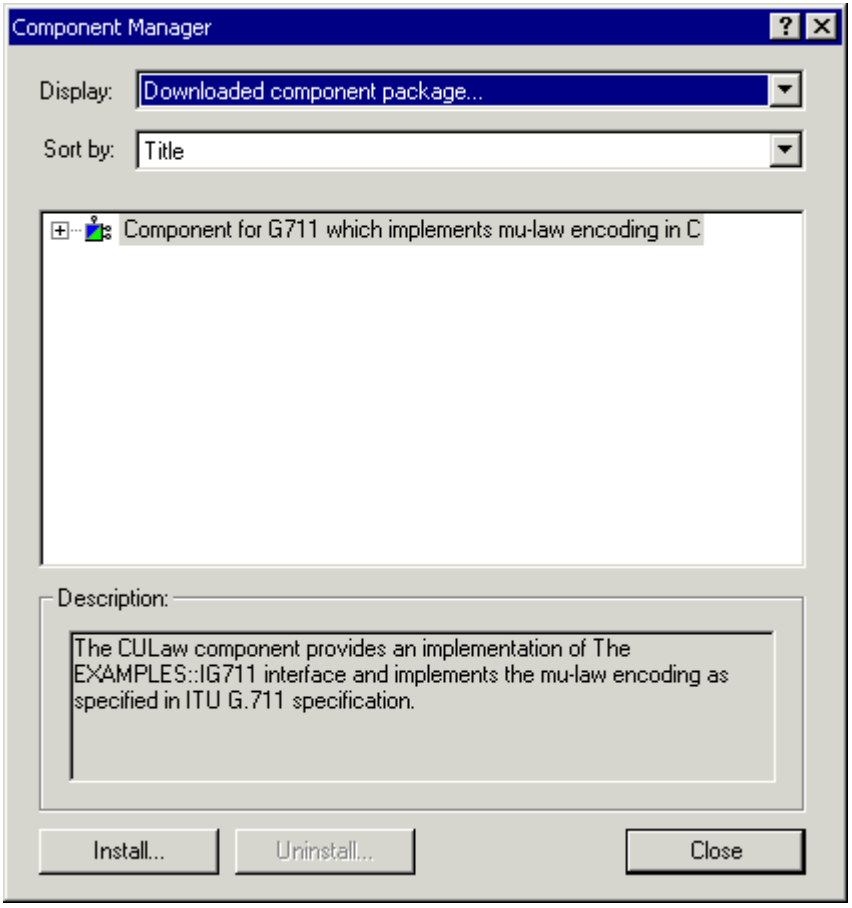

Figure 2-40. Component Manager Dialog Box – Downloaded Component

5. Click the **Install…** button. Component Manager installs the component on your system. Once the component is installed, click **OK**.

6. In the **Display** field select **Locally installed components** from the drop-down list, and in the **Sort by** field select **Title**.

   Select **Component for G711 which implements the mu-law encoding in C**, as shown in Figure 2-41.

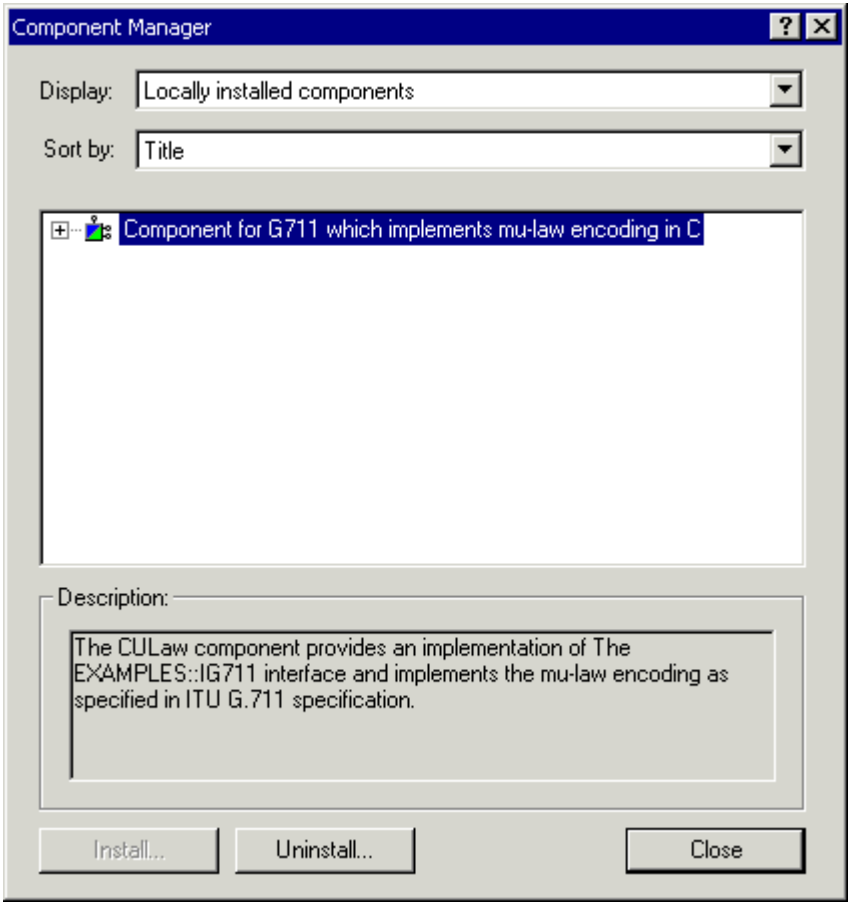

Figure 2-41. Component Manager Dialog Box – Selected Component

7. Click **Close** to close Component Manager.

Step 3: Add the Component to Your Project

To add the newly installed component to the project:

1. From the **Tools** menu, select the **VCSE** submenu and then choose **Add Component**.
2. Click **Component for G711 which implements the mu-law encoding in C** to select it.

If you have multiple components on your system and you are not sure which one to add, click the expand button ⊞ to display the component information (shown in Figure 2-42). Make sure that **ADSP-BF535** is designated as the **Processor** for the component that you are adding to your project.

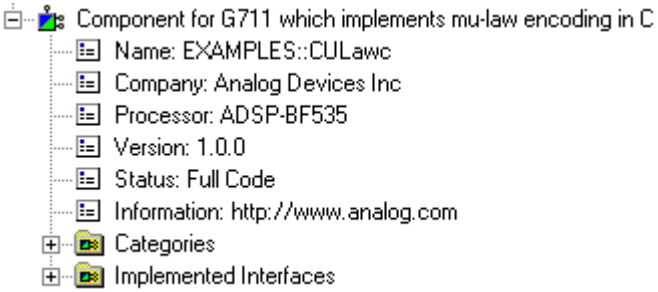

Figure 2-42. Expanded View of Component Information

3. Click **Add** to indicate that you want to add the component to the project.

   Component Manager displays the dialog box shown in Figure 2-43 on page 2-59.

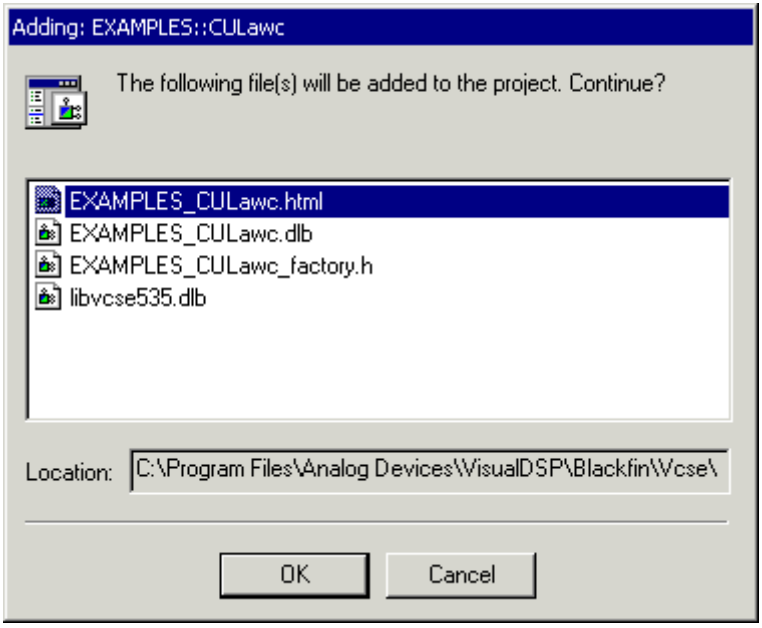

Figure 2-43. Adding Files to the Project

4. Click **OK** to add the component files to the project.

Step 4: Build and Run the Program

To build and run the program:

1. From the **Project** menu, choose **Build Project**.

   VisualDSP++ displays the message shown in Figure 2-44 on page 2-60.

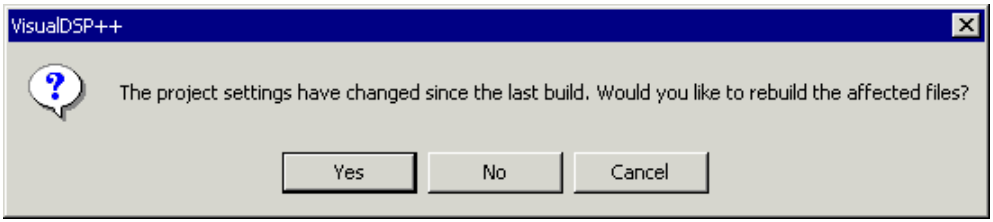

Figure 2-44. Rebuilding Files Affected by Changes to Project Settings

2. Click **Yes**. VisualDSP++ compiles the source files and creates the program.
3. From the **Debug** menu, choose **Run** to execute the program. The program generates the following output.

```
Harness to test component code generated by
EXAMPLES_CULawc.idl
Testing EXAMPLES::IG711
Test Completed result = MR_OK
```

You have now completed this tutorial.

I INDEX