APPENDIX A

CODE COMPOSER GUIDE
for
EE757
Control Systems Laboratory

Winter Quarter 2002
The Ohio State University
Department of Electrical
Engineering

About This Appendix

This appendix summarizes how to use the Code Composer development environment to build and debug embedded real-time software applications. The contents of this appendix are taken from the *TMS320C2xx/C24x Code Composer User's Guide*, and prepared so as to make the students familiar with the tools to be used in EE757 Control Interfacing and Design Laboratory. Its aim is not to serve as a complete guide for the code composer, it only covers necessary parts needed in the experiments of EE757. After making a fast start to Code Composer, the reader is encouraged to refer to the guide supplied by *Texas Instruments*

Oguz Hasan DAGCI and Umit Yusuf OGRAS

Table of Contents

1. Basic Tools of Code Composer

1.1 Launching Code Composer and Opening a Project

To launch the code composer, the following steps should be followed:

1. Connect the eZdsp™ to the PC parallel port using cable provided.
2. Connect the power supply to the eZdsp™. Verify the red power lights are on.
3. From the desktop, launch Code Composer. You should see the debugger window open.

If you do not perform step1 or step2, the computer will give an error message indicating that there is no response from the board. Click *Abort*, and check your connections, then launch again.
Once the Code Composer is started, we need to open a project to start working with the DSP board. If you want to open a new project:
Go to *Project* from pull-down menu bar, select new project.
If you want to open an existing project, perform the following steps:

1. From the pull-down menu bar, go to Project → Open
2. Browse to the project file directory, select the project and click Open.
3. Under the "Files" window, expand the project tree.

You are not required to learn how to develop new projects for this course. Therefore, we will directly explain how to compile existing projects and load them to the DSP chip. However, it is easy and for your benefit to learn the necessary steps to build a new project from the beginning by reviewing the example projects we will supply throughout the quarter.
After opening the project, we need to build and download the program to the board. These are accomplished as follows:

1. Build the project by going to the pull-down menu: Project → Rebuild All. The output window status should read: "Build complete, 0 Errors, 0 Warnings" when complete.
2. Download the generated machine code to board. From the pull-down menu go to File → Load Program. Select "*example_project.out*" from the dialog box in directory C:\tic2xx\myprojects\ *example_project* \ and click on "Open".
Now you are ready to run the compiled and linked code using Code Composer.
3. From the pull-down menu bar, go to Debug → Reset DSP and after that, go to Debug → Run.

While the program running we need to have debugging tools to check whether the program is running as expected or to observe some of the variables in the code. *Code Composer* provides a number of debugging tools. These tools will be explained in the following sections.

1.2 Resetting The Target Processor

To reset the target processor the following commands can be used:
Reset DSP: The Debug -> Reset DSP command initializes all register contents to their power-up state and halts the execution of your program.
Restart: The Debug -> Restart command restores the PC to the entry point for the currently loaded program. This command does not start program execution.

1.3 Single Stepping

Single stepping is a useful debugging tool to observe the program flow closely.
The following commands from the pull-down menu Debug can be used to single step through code.
Step Into: Single stepping can be done through the code by either clicking the button on the Debug toolbar or by selecting Debug->StepInto. This command steps through a single C instruction. In order to go step by step in a function called in main(), step into command should be executed just before the call of the function in main().
Step Over: The step over command can be used to step through and execute individual statements in the current function. If a function call is encountered, the function executes to completion unless a breakpoint is encountered before it stops after the function call. You can step over code by either clicking the button on the Debug toolbar or by selecting Debug->StepOver.
Run to Cursor: Run to cursor feature can be used to run the loaded program until it encounters the cursor position. You can run to cursor by selecting Debug->Run to Cursor.

1.4 CPU Registers

The CPU and peripheral registers of the target processor can be viewed using the Register window. The contents of registers can be edited using the Edit Registers dialog.

1.4.1 Viewing Registers

To view the contents of the CPU registers of the target processor, select the command View->CPU Registers->CPU Register.

1.4.2 Editing Registers

To Edit the Contents of a Register, select Edit->Edit Register. The Edit Registers dialog box appears.
The Edit Registers dialog offers the following options:
Register: Specify the register you want to edit by typing its name or by selecting a register from the drop-down list.
Value: This field contains the current value of the specified register displayed in hex. You can enter another value in this field in hex (with the prefix 0x) or in decimal (with no prefix). You can also enter any valid C expression. After modifying the value of a register, click Close to apply your changes.
Click Close again to close the dialog box.

1.4.3 Expression Input Fields

All input fields that require a numerical value are C expression input fields. In these input fields, you can enter any valid C expression, including the names of C functions or assembly language labels. The expression is then reduced to a single numerical value and displayed.

1.5 Editing Variables

To Edit a Variable
1) Select Edit->Edit Variable.
2) In the Edit Variable dialog box, enter the following information:
Variable. The name of the variable to be edited.
Value. The new value.
3) Click OK to perform the edit.

1.6 Refreshing Windows

All windows usually show the current status of the target board. However, if you connect a window to a Probe Point, the window may not contain the latest information (see Section 2, Breakpoints and Probe Points). To update a window, select Window->Refresh. This command updates the active window with the current target data.

1.7 Saving and Restoring Your Workspace

You can save and restore your current working environment, called a workspace, between debugging sessions.
To Save a Workspace
1) Select File->Workspace->Save Workspace.
2) In the Save As dialog box, enter a name for the workspace in the File name field.
3) Click Save.
To Load a Workspace
1) Select File->Workspace->Load Workspace.
2) In the Load Workspace dialog box, enter the name of the workspace file in the File name field.
3) Click Open.

2. Breakpoints and Probe Points

Breakpoints are the debugging tools that are used to stop the execution of the program at a desired point. When the program is stopped, you can examine the state of your program, watch or modify variables. A breakpoint can be set, by using the Toggle Breakpoint button on the Project toolbar or by selecting Debug->Breakpoints. The latter brings up the Break/Probe/Profile Points dialog box. When a breakpoint is set, you can enable or disable it.

2.1 Adding and Deleting Breakpoints

There are several ways to add a break point to a program.

Adding a breakpoint using the Breakpoint Dialog Box:
1) Select Debug->Breakpoints. The Break/Probe/Profile Points dialog box appears with the Breakpoints tab selected.
2) In the Breakpoint Type field, select either Break at Location (unconditional).
3) Enter the location where you want to set the breakpoint, using either of the following formats:
For an absolute address, enter any valid C expression, the name of C function, or a symbol name.
Enter a breakpoint location based on your C source file. This is convenient when you do not know where the C instruction ends up in the executable. The format for entering a location based on the C source file is as follows: *fileName*, line *lineNumber*
4) Click Add to create a new breakpoint. This causes a new breakpoint to be created and enabled.
5) Click OK to close the dialog box.

Adding a breakpoint using the Toolbar
Using the Toggle Breakpoint button on the Project toolbar is the easiest way to set and clear breakpoints at any location in the program. The breakpoint dialog box allows you to set more complex breakpoints, such as conditional breakpoints or hardware breakpoints.
1) Put the cursor in the line where you want to set the breakpoint. You can set a breakpoint in either a Dis-Assembly window or an Edit window containing C source code.
2) Click the Toggle Breakpoint button on the Project toolbar. The line is highlighted.
Removing a breakpoint using the Toolbar
1) Put the cursor in the line containing the breakpoint.
2) Click the Toggle Breakpoint button on the Project toolbar.

Changing an existing Breakpoint
1) Select Debug->Breakpoints. The Break/Probe/Profile Points dialog box appears with the Breakpoints tab selected.

2) Select a breakpoint in the Breakpoint window. The selected breakpoint is highlighted and the Breakpoint Type, Location, and Expression fields are updated to match the selected breakpoint.
3) Edit the breakpoint Type, Location, and/or Expression fields as required.
4) Click Replace to change the currently selected breakpoint.
5) Click OK to close the dialog box.

Deleting an existing Breakpoint
1) Select Debug->Breakpoints. The Break/Probe/Profile Points dialog box. appears with the Breakpoints tab selected.
2) Select a breakpoint in the Breakpoint window.
3) Click Delete to delete the breakpoint.
4) Click OK to close the dialog box.

Deleting all breakpoints using the Breakpoint Dialog Box
1) Select Debug->Breakpoints. The Break/Probe/Profile Points dialog box appears with the Breakpoints tab selected.
2) Click Delete All.
3) Click OK to close the dialog box.

Deleting all breakpoints using the Toolbar
Click the Remove All Breakpoints button on the Project toolbar.

2.2 Hardware Breakpoints

Hardware breakpoints differ from software breakpoints in that they do not modify the target program. Hardware breakpoints are useful for setting breakpoints in ROM memory or breaking on memory accesses instead of instruction acquisitions. A breakpoint can be set for a particular memory read, memory write, or memory read or write. Memory access breakpoints are not shown in the Edit or Memory windows.

Adding a Hardware Breakpoint
1) Select Debug->Breakpoints. The Break/Probe/Profile Points dialog box appears with the Breakpoints tab selected.
2) In the Breakpoint type field, choose H/W Break at location for instruction acquisition breakpoints.
3) Enter the program or memory location where you want to set the breakpoint. Use one of the following methods:
For an absolute address, you can enter any valid C expression, the name of a C function, or a symbol name.
Enter a breakpoint location based on your C source file. This is convenient when you do not know where the C instruction is in the executable. The format for entering in a location based on the C source file is as follows: *filename*, line *lineNumber*
4) Enter the number of times the location is hit before a breakpoint is generated, in the Count field. Set the count to 1 if you wish to break every time.

5) Click Add to create a new breakpoint. This causes a new breakpoint to be created and enabled.
6) Click OK to close the dialog box.

2.3 Probe Points

Probe Points allow you to cause an update of a particular window or to read and write samples from a file that occur at a specific point in your algorithm. This connects a signal probe to that point in your algorithm. When the Probe Point is set, you can enable or disable them just like breakpoints. When a window is created, by default, it is updated at every breakpoint. However, you can change this so the window is updated only when the program reaches the connected Probe Point. When the window is updated, execution of the program is continued.

2.4 Adding and Deleting Probe Points

Adding a Probe Point
You can create Probe Points by placing the cursor on the line in a source file or Dis-Assembly window and clicking the Probe Point button on the Project toolbar. Probe Points must be connected to a window or file.

Deleting an existing Probe Point
1) Select Debug->Probe Points. The Break/Probe/Profile Points dialog box appears with the Probe Points tab selected.
2) Select a Probe Point in the Probe Point window.
3) Click Delete.
4) Click OK to close the dialog box.

Deleting all probe points using the Probe Point Dialog Box
1) Select Debug->Probe Points. The Break/Probe/Profile Points dialog box appears with the Probe Points tab selected.
2) Click Delete All.
3) Click OK to close the dialog box.

2.5 Connecting Probe Points

Connecting a Display Window to a Probe Point
1) Open the window that you want to connect to.
2) Create the Probe Point by placing the cursor on the line where you want the point set and clicking on the Toggle Probe Point button on the Project toolbar.
3) Select Debug->Probe Points. The Break/Probe/Profile Points dialog box appears with the Probe Points tab selected. In the Probe Point window, the new Probe Point that you have created appears. This Probe Point indicates it currently has no connection. Select this Probe Point to make it current. You can now edit its fields in the dialog box.

4) Connect the window or file to the Probe Point. The Connect To drop-down list contains all the files and windows that can be connected to the Probe Point. From this list select the appropriate item.
5) Click Add to create the new Probe Point or click Replace to modify the existing Probe Point.

3 The Graph Window

Code Composer incorporates a signal analysis interface that enables developers to monitor signal data. This chapter describes how you can use the graphing capabilities of Code Composer to view signals on your target system.

3.1 Time/Frequency

The graph menu contains many options that allow you to be flexible in how you display your data. You can use a time/frequency graph to view signals in either the time or frequency domain. In time domain analysis, no preprocessing is done on the display data before it is graphed.

Adding Graph Window
Select View -> Graph -> Time/Frequency to view the Graph Property Dialog box. Field names appear in the left column. You can adjust the values as needed in the right column, then click OK. The graph window appears with the properties you have set. You can change any of these parameters from the graph window by right-clicking the mouse, selecting Properties, and adjusting the parameters as needed. You can also update the graph at any point in your program.
All input fields are C expression input fields. An expression containing a symbol name can be used for all fields requiring numerical inputs, such as the start address and acquisition buffer size.

3.1.1 How the Time/Frequency Graph works

There are two buffers associated with the graph window: the acquisition buffer and the display buffer. The acquisition buffer resides on the target board. It contains the data that you are interested in. When a graph is updated, the acquisition buffer is read from the target board and the display buffer is updated. The display buffer resides in the host memory so it keeps a history of the data. The graph is generated from the data in the display buffer.
When you enter your parameters and press OK, the graph window receives an acquisition buffer of DSP data of the length you entered in the Acquisition Buffer Size field. This begins at the location in the Start Address field in the DSP data memory space. A display buffer of size Display Data Size is allocated within the host memory with all its values initialized to 0.
If you enable the Left-Shifted Data Display field, the entire display buffer is left shifted by the value in the Acquisition Buffer Size field, with the values of the DSP acquisition buffer shifted in from the right end. The values of the display buffer are overwritten by the DSP acquisition buffer.
Left-shifted data display is useful when you process a signal serially. Although the samples are only available one at a time, left-shifted data display lets you
view a history of the samples. When the associated Probe Point is reached, a block of DSP data is read and the display is updated.

The following sections describe input fields in the Graph Property Dialog box.

3.1.2 Display Type

The Display Type option in the Graph Property Dialog box contains several options in the drop-down menu in the right column. Some options for this field are associated with constellation. Selecting some graph options causes additional fields to appear in the Graph Property Dialog box.
We will be using Single Time plots for this 757 lab.

Single Time: Plots the data in the display buffer on a magnitude versus time graph. A single time trace of a signal is displayed on the graph. When you enable this option, the following fields appear in the Graph Property Dialog Box:
Time Display Unit: Specifies the unit of measure for the time axis of the graph. Select among the values: s (second), ms (millisecond), us (microsecond), and sample (displays the values on the time axis in terms of the display buffer index).
Start Address: Starting location (on the target board) of the acquisition buffer containing the data to be graphed.
When the graph is updated, the acquisition buffer, starting at this location, is fetched from the target board. The acquisition buffer then updates the display buffer, which is graphed. You can enter any valid C expression in the Start Address field. This expression is recalculated every time samples are read from the target. This means that if you enter a symbol in this field and the symbol later changes value, you do not have to reenter this parameter.

3.1.3 Graph Title

You can identify each graph that you create with a unique title.

3.1.4 Data Page

If your target consists of multiple pages, such as program, data and I/O, you can specify pages using the Data Page options. From the list, select Data because in EE757 the variables will be displayed on the graph window. This indicates whether the page of variable/memory location graphically displayed is the program, data or I/O page.

3.1.5 DSP Data Type

This field allows you to select among the following data types:

1) 32-bit signed integer
2) 32-big unsigned integer
3) 32-bit floating point
4) 32-bit IEEE floating point
5) 16-bit signed integer
6) 16-bit unsigned integer
7) 8-bit signed integer
8) 8-bit unsigned integer

3.1.6 Sampling Rate (Hz)
This field contains the sampling frequency for acquisition buffer samples, such as for analog to digital conversion. The sampling rate is used to calculate the time and frequency values displayed on the graph. For a time domain graph, this field calculates the values for the time axis. The axis is labeled from 0 to (Display Data Size * 1/Sampling Rate).

3.1.7 Plot Data From
This field determines the ordering of the data within the acquisition buffer. You can toggle between the following options: Left to Right, where the first sample in the acquisition buffer is considered the newest or most recently arriving, and Right to Left where the first sample in the acquisition buffer is considered the oldest.

3.1.8 Left-Shifted Data Display
This option controls how the acquisition buffer is merged into the display buffer. When a graph is updated, the acquisition buffer is fetched from the target board and merged into the display buffer. If you enable Left-Shifted Data Display, the entire display buffer is left shifted, with the values of the target board acquisition buffer shifted in from the right end. Note that at start up, all values in the display buffer are initialized to 0. If the Left-Shifted Data Display option is not enabled, then the values of the display buffer are overwritten by the target board acquisition buffer.
The Left-Shifted Data Display option is useful when you are processing a signal in serial fashion. Although the samples are only available one at a time, the Left-Shifted Data Display option lets you view a history of the samples. When a Probe Point associated with the window is reached, a block of the target board data is read and the display is updated. If you left shift the data into the display, make sure that the graph window is only updated when the target board data is valid.

3.1.9 Autoscale
This option allows the maximum value of the Y axis to be determined automatically. If you enable Autoscale, the graph uses the maximum value in the display buffer to set the Y axis range and graphs all values accordingly. If you disable Autoscale, an additional field appears in the Graph Property Dialog box:
Maximum Y-Value. Sets the maximum value of the Y-axis displayed on the graph.

3.1.10 DC Value
This option sets the middle point of the Y axis range; the Y axis is symmetrical about the value you enter in the DC Value field. This value is enabled regardless of whether the Autoscale field is enabled.

3.1.11 Axes Display
This option turns the X and Y axes in the graph window on and off.

3.1.12 Status Bar Display
This option turns the status bar display at the bottom of the graph window on and off.

3.1.13 Magnitude Display Scale

This field sets the scaling function used for data values in the graph. You may choose between the following options:
Linear: Uses unmodified integer values
Logarithmic: Uses the function 20 log(x)

3.1.14 Data Plot Style

This field sets how the data is visually represented in the graph. You may choose between the following options:
Line: Connects data values linearly
Bar: Uses vertical lines to display values

3.1.15 Grid Style

This field sets the pattern of horizontal and vertical background lines in the graph. You may choose among the following options:
No Grid
Zero Line: Displays only the 0 axes
Full Grid: Displays the full grid

3.1.16 Cursor Mode

This field sets the cursor's appearance and function in the graph. You may choose among the following options:
No Cursor
Data Cursor: Appears on the graph screen with the cursor coordinates in the graph status bar.
Zoom Cursor: Allows you to enlarge areas of the graph. Place the cursor on one corner of the area, hold the left mouse button down, and draw a rectangle around the area of interest.

3.2 Constellation Diagram

The graph menu contains many options that allow you to be flexible in how you display your data. You can use a constellation graph to measure how effectively the information is extracted from the input signal. The input signal is separated into two components and the resulting data is plotted using the Cartesian coordinate system in time, by plotting one signal versus the other (Y source versus X source, where Y is plotted on the Y axis and X on the X axis).
Use the View -> Graph -> Constellation command to view the Graph Property Dialog box. Field names appear in the left column. You can adjust the values as needed in the right column, then click OK. The graph window appears with the properties you have set. You can change any of these parameters from the graph window by right-clicking the mouse, selecting Properties, and adjusting the parameters as needed. You can also update the graph at any point in your program.
All input fields are C expression input fields. An expression containing a symbol name can be used for all fields requiring numerical inputs, such as the start address and acquisition buffer size.

3.2.1 How the Constellation Diagram works

There are two buffers associated with the graph window: the acquisition buffer and the display buffer. The acquisition buffer resides on the target board. It contains the data that you are interested in. When a graph is updated, the acquisition buffer is read from the target board and updates the display buffer. The display buffer resides in the host memory so it keeps a history of the data. The graph is generated from the data in the display buffer. When you have entered all your option choices and press OK, the graph window receives an acquisition buffer of DSP data of length you entered in the Acquisition Buffer Size field, starting at DSP location Start Address in the data memory space. A display buffer of size Constellation Points is allocated within the host memory, with no data to display initially.

When the graph is updated, the entire display buffer is left shifted by the value in the Acquisition Buffer Size field, with the values of the DSP acquisition buffer shifted in from the right end. This is useful when you are processing a signal in serial fashion. Although the samples are only available one at a time, this lets you view a history of the samples. When the associated Probe Point is reached a block of DSP data is read and the display is updated.

The following sections describe input fields in the Graph Property Dialog box.

3.2.2 Display Type

The Display Type option in the Graph Property Dialog box contains several options in the drop-down menu in the right column. The Constellation option appears by default when you use the command View -> Graph -> Constellation.

3.2.3 Graph Title

You can identify each graph that you create with a unique title. This helps to differentiate results when there are many windows open.

3.2.4 Interleaved Data Sources

Specifies whether the signal sources are interleaved or not. Toggling this display option allows a single buffer input to represent two sources. Setting this option to Yes implies a 2-source input buffer, where the odd samples represent the first source (X source) and even samples represent the second (Y source). Setting this option to Yes creates the following additional field in the Graph Property Dialog box:

Start Address. Starting location (on the target board) of the acquisition buffer containing the data to be graphed. When the graph is updated, the acquisition buffer, starting at this location, is fetched from the target board. This acquisition buffer then updates the display buffer, which is graphed. You can enter any valid C expression in the Start Address field. This expression is recalculated every time samples are read from the target. This means that if you enter a symbol in this field and the symbol later changes value, you do not have to reenter this parameter.

Setting Interleaved Data Sources to No creates the following additional fields:

Start Address - X Source

Start Address - Y Source

3.2.5 Data Page

If your target consists of multiple pages, such as program, data and I/O, you can specify pages using the Data Page options. From the list, select Data because in EE757 the variables will be displayed on the graph window. This indicates whether the page of variable/memory location graphically displayed is the program, data or I/O page.

3.2.6 Acquisition Buffer Size

This is the size of the acquisition buffer you are using on your target board. For example, if you are processing samples one at a time, enter a 1 in this field. Make sure that you connect your display to the correct location in your program.

When a graph is updated, the acquisition buffer is read from the target board and updates the display buffer. The display buffer is graphed. You can enter any valid C expression for the Acquisition Buffer Size field. This expression is recalculated every time samples are read from the target. Therefore, if you enter a symbol in this field and the symbol later changes values, you do not have to reenter this parameter.

3.2.7 Index Increment

This field allows you to specify the sample index increment for the data graph. A specification in this field is equivalent to a sample offset for noninterleaved sources. This permits you to extract signal data from multiple sources using a single graph. An index increment of 2, for instance, corresponds to a sample offset value of 2, which in turn graphically displays every other sample in the acquisition buffer. You can, therefore, specify multiple data sources for display by entering the corresponding offset value in this field.

This option provides a general specification for interleaved sources. If you enable the Interleaved Data Sources option, the Index Increment option is disabled.

3.2.8 Constellation Points

This is the size of the display buffer that is graphed on your screen. The display buffer resides on the host, so a history of your signal can be displayed even though it no longer exists on the target board. Constellation points are the maximum number of samples that the graph displays. Usually the Constellation Points field is greater than or equal to the Acquisition Buffer Size field. If Constellation Points is greater than the Acquisition Buffer Size, the acquisition buffer data is left shifted into the display buffer. You can enter any valid C expression for the Constellation Points field. This expression is calculated when you click OK in the Graph Property Dialog box.

3.2.9 Symbol Size

This property provides a way to set the display size of each symbol. Each constellation is displayed as an X symbol. The following options are associated with this display property:

Dot: Displays each point as a dot instead of an X symbol
Small
Medium
Large
Extra Large

3.2.10 Axes Display
This option turns the X and Y axes in the graph window on and off.

3.2.11 Status Bar Display
This option turns the status bar display at the bottom of the graph window on and off.

3.2.12 Grid Style
This field sets the pattern of horizontal and vertical background lines in the graph. You may choose among the following options:
No Grid
Zero Line: Displays only the 0 axes
Full Grid: Displays the full grid

3.2.13 Cursor Mode
This field sets the cursor's appearance and function in the graph. You may choose among the following options:
No Cursor
Data Cursor: Appears on the graph screen with the cursor coordinates in the graph status bar.
Zoom Cursor: Allows you to enlarge areas of the graph. Place the cursor on one corner of the area, hold the left mouse button down, and draw a rectangle around the area of interest.

4 The Watch Window

4.1 Adding and Deleting Expressions in the Watch Window

Adding an expression in the Watch window, follow the steps below:
1) Select View -> Watch Window.
or
2) Click the Watch Window button on the Debug toolbar:

Adding a new expression to the Watch window, use any of the following methods:
1) Select one of the four Watch window tabs. Press the Insert key on the keyboard. This brings up the Watch Add Expression dialog box. Type the expression you wish to examine in the Expression field and press OK.
or
2) In the Watch window, right-click and select Insert New Expression from the context menu. The Watch Add Expression dialog box appears. In the Expression field, type the expression you wish to examine and press OK.

Deleting expressions in the Watch window,
1) Select the expression you wish to remove from the active Watch window by clicking on it with the mouse or using the up/down arrow keys to move to the expression.
2) Press the Delete key on the keyboard. If the expression is expanded, all subexpressions are removed from the Watch window.

4.1.1 Expanding and Collapsing Watch Variables

Variables that contain more than one element, such as arrays, structures, or pointers, are displayed with either a + or - sign preceding them. The + symbol indicates that the variable contains elements and can be expanded. The - symbol indicates that the variable is fully expanded and can be collapsed.

Expanding or Collapsing a variable
1) Select the variable you wish to expand by using the mouse or by using the up/down arrow keys.
2) When the variable is current, you can toggle its expansion state by pressing the Enter key.

4.2 Editing Variables in the Watch Window

You can modify the Watch window expression and its value as follows:
1) Select the tab for theWatch window you wish to use. In the window, select the variable you wish to edit by clicking on it with the mouse or by using the up/down arrow keys.
2) Double-click on the variable to obtain the Edit Variable dialog box.
3) Edit the information in the Value field as desired.

You cannot edit the Variable field if the variable is an expanded expression or if it is an element of an expanded variable. If you want to change the variable, you must first collapse the variable and then edit it.