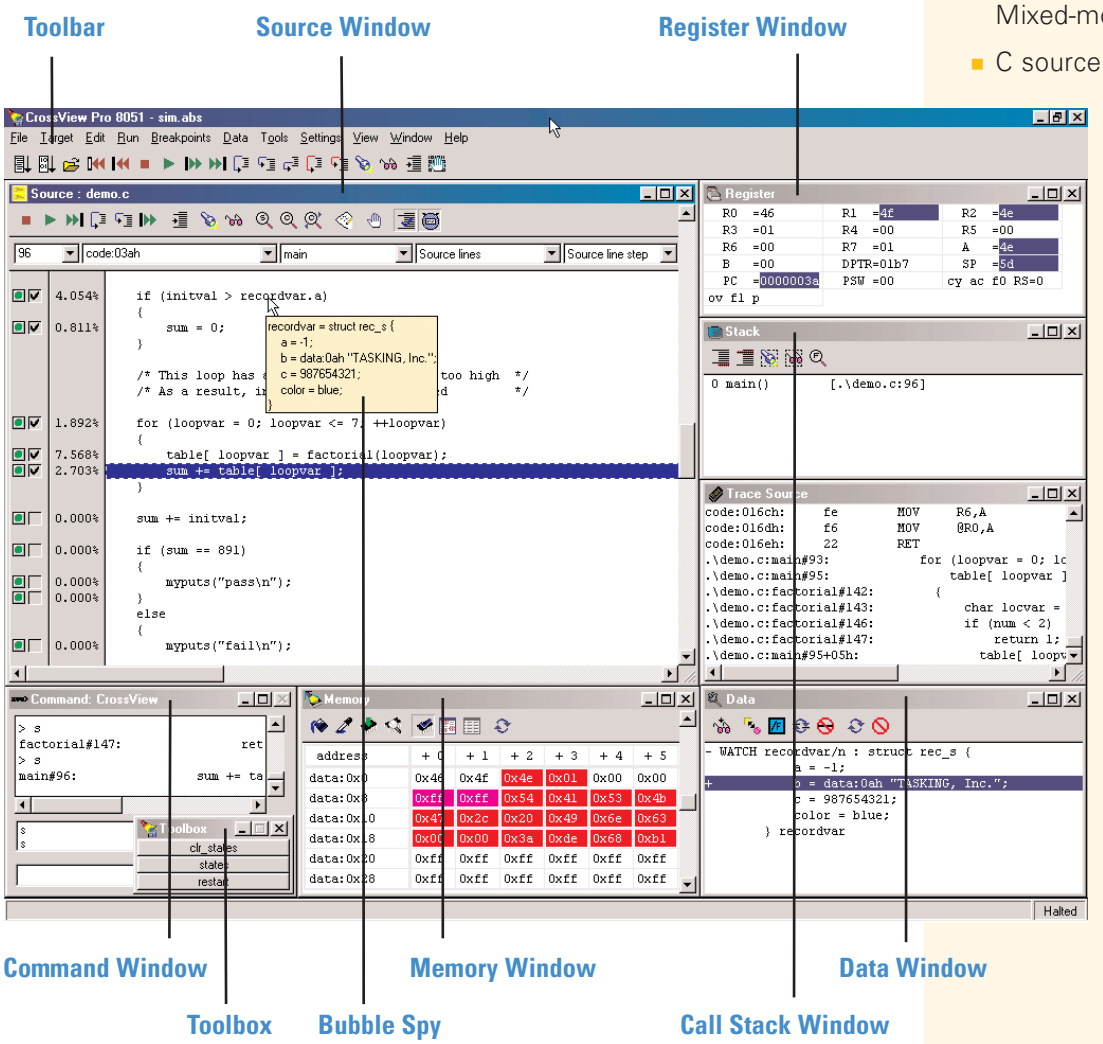
Debugging your 8051 Project with CrossView Pro™

TASKING
Embedded software development from Altium

The **TASKING CrossView Pro Debugger**, with powerful and extensive debugging features in an intuitive user interface, helps you to debug your application faster. The CrossView Pro debugger is a true Windows application complete with multiple, resizable, and independently controlled windows. It combines the flexibility of the C language with the control of code execution found in assembly language, bringing functionality that reduces the time spent testing and debugging.

THE CROSSVIEW PRO DEBUGGING ENVIRONMENT

Through multiple, resizable and independently controlled windows that have local menus available with the right mouse click, you can keep all the information you need and consider important under control for your debugging session.



You can open multiple windows by selecting View from the menu, then choosing the one you need to activate for your debugging session.

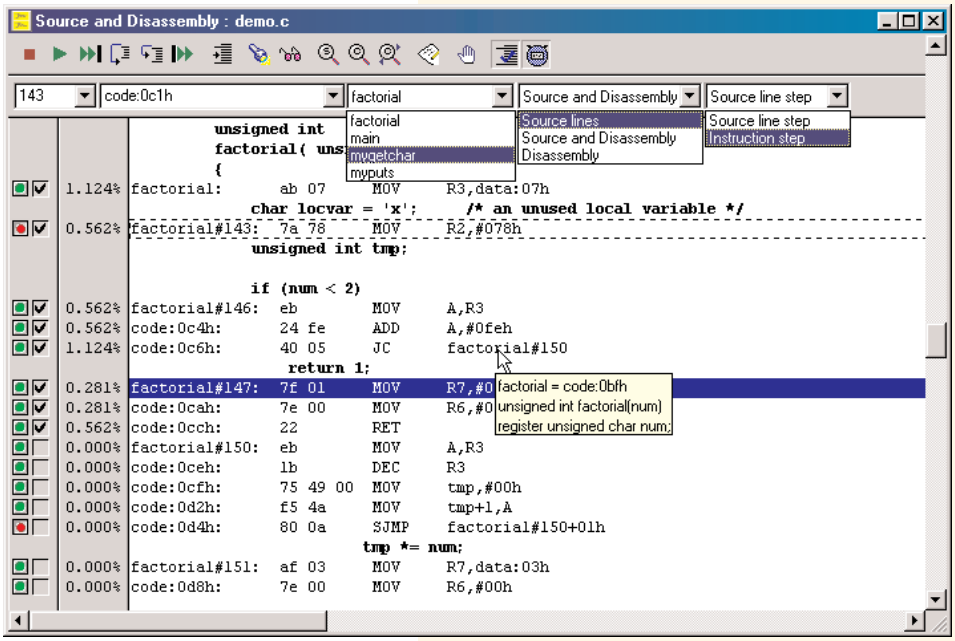
Available features include:

- Multiple Data, Memory and Register windows (with different display mode - i.e. hex, ASCII, decimal, binary, float)
- Code displayed in Source, Assembly or Mixed-mode (source/assembly)
- C source line highlighted in mixed assembly mode
- Bubble-Spy™ for both variables and functions
- Code and data breakpoints
- Direct memory and register access
- Register Grouping
- C, assembly and stack level Trace
- Programmable Signal Data Analysis
- Code and Data Coverage, Profiling
- Record and playback debug sessions
- C expression evaluation with powerful macro language.

Source Window

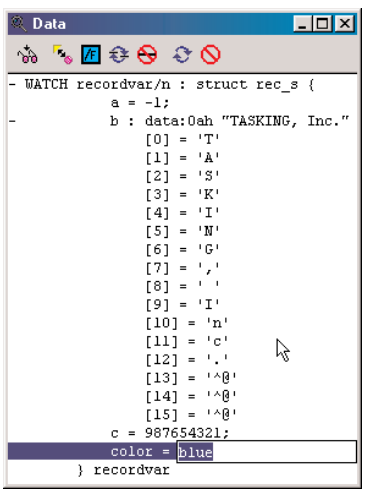
The **Source Window** controls the execution of the program. From its local toolbar icons (or through the right mouse click) you can view the source, control the execution of your program (step, go, halt), set and clear breakpoints (simply by pressing the green/red button on the left side of the window) and enable and disable code coverage or profiler. Many other commands are available.

The Source Window allows you to view your code at C level, Assembly level, or in Mixed mode where the C code and corresponding assembly code display simultaneously. From the Source Window, you can jump directly into the editor within the Embedded Development Environment (EDE) and you will find yourself positioned at the source line where you had your cursor in the debugger. This gives you immediate access to the problem that you want to correct. By positioning the mouse over a variable or function in the Source Window, the value of the variable or function is displayed in a popup box called a *Bubble-Spy*. This feature enables you to immediately view values by simply moving your mouse.



Data Windows

Multiple **Data Windows** can be opened simultaneously. These windows enable you to watch or show data, browse for local or global variables, double-click to modify values, or expand and contract complex data structures. Within these windows you can reformat (change display radix and type) on an element-by-element basis. You can show or watch locals from any stack level, automatically track and display locals and easily copy any variable to a new window as show or watch.



To open a *Data Window* for either global or local variables, use the **View > Data** command available from the menu. To add a new variable to the Data Window, you can double-click on the name of the variable from the Source Window, or you can browse for it using the **Search > Browse Variable** command. Select the variable you need and press **OK**. The Expression Evaluation window will pop up and the **Add Watch** and **Add Show** buttons will be available to respectively monitor and inspect the variable.

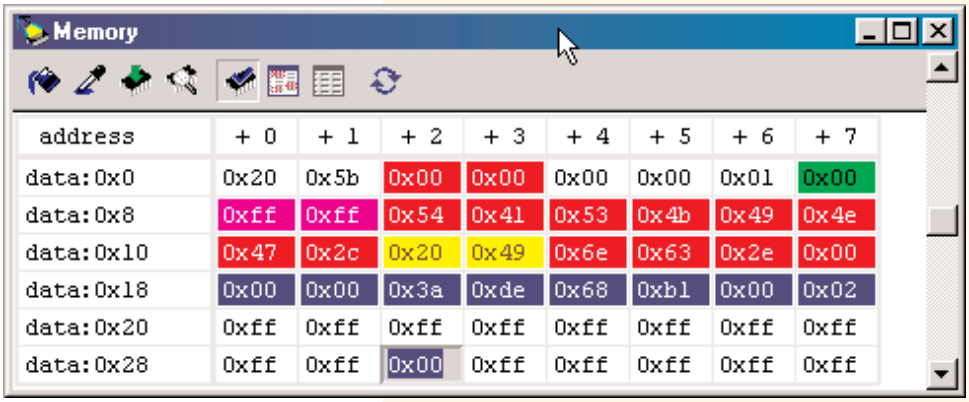
WATCH expressions are monitored and updated after every halt in execution and are marked with the text "WATCH" at the start of the display line in the Data Window. SHOW expressions, on the other hand, are one-shot inspections of an expression's value and are not updated except by direct user action.

TIP

The control bar at the top of the source window allows you to see the assembly code intermixed with the corresponding C-code lines of the example program.

Memory Windows

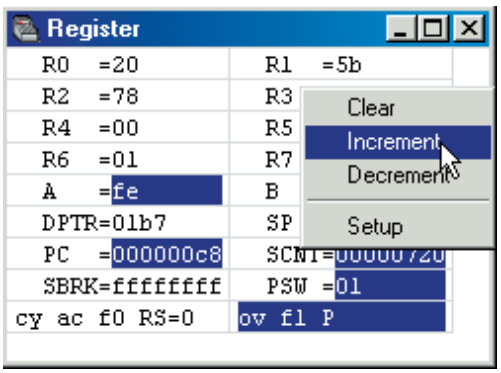
Multiple **Memory Windows** can be simultaneously opened, each pointing to a different address and in a different display mode. These windows enable you to monitor and modify any memory location and have complete control over the size and format of the data. Highlighted contents indicate what has changed since last stop. To open a Memory Window, use the **View > Memory** command available from the menu.



You can change the address you need to display by first clicking on the address field and then typing either the symbolic or the hexadecimal value. You can change the value at a particular location by first clicking on the corresponding cell. This will put the Memory Window in edit mode. You can then type in a new value.

The cell in the Memory Window are color-coded by function. The colors represent:

Green → Memory Read
Blue → Memory Fetch
Red → Memory Write
Yellow → Memory Read/Write/Fetch
Purple → Memory Write/Fetch

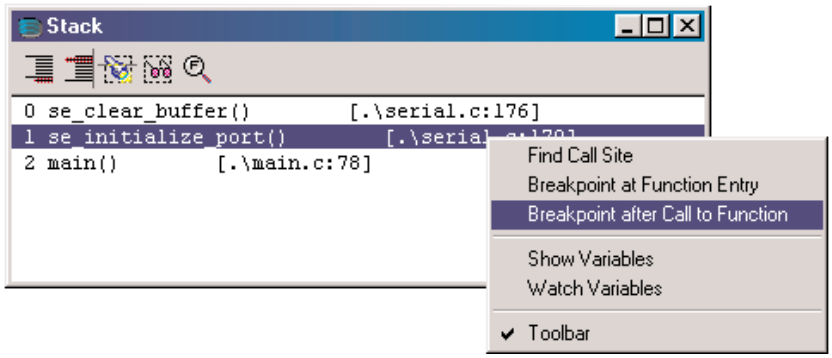


Register Window

The **Register Window** displays and modifies CPU registers values. The window is fully configurable and is updated every time the program is stopped. Highlighted registers indicate what has changed since the last stop. You can configure the Register Window into meaningful sets such as General Purpose, Timers or I/O Ports and display all of them simultaneously. To modify a register value, select the register to be modified by double clicking on its line. Entering a value and pressing the **Enter** key updates the register value.

Stack Window

The **Stack Window** displays the state of the current stack frame. With simple point and click operations you can set up level breakpoints (e.g. set breakpoint at function entry or exit), display source for function calls and display local variables for the selected functions.

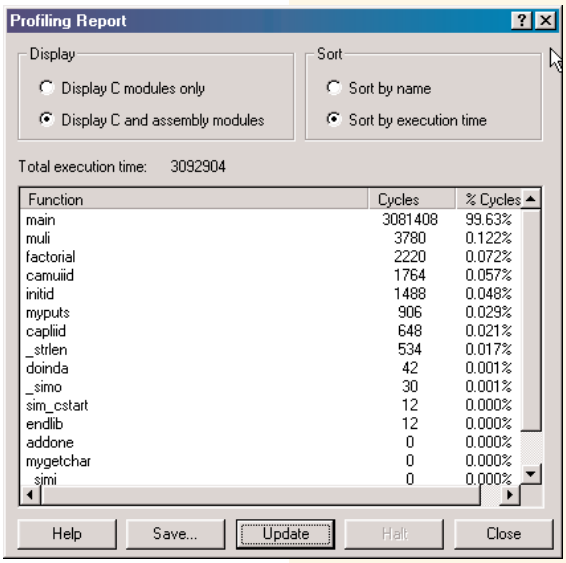


TIP

Remember that the Stack breakpoints are temporary, so they will be automatically removed when the breakpoint is hit.

Profiling Report

The **Profiling Report** allows you to perform timing analysis by providing timing information (in state counter value) about a particular function, set of functions or instruction. You can see how often a function is called and how much time (in percentage or state counter value) is spent in each function. To open the window, select **Debug > Profiling report** then click the **Update** button to display the information. Code Profiling, also known as Performance Analysis, is a feature of the CrossView Pro Simulator.



Code Coverage Window

Through the **Code Coverage Window**, you can find executed and non-executed areas of the application program. A graphical column that shows the executed instructions is also available on the left side of the Source Window (see the section describing the Source Window above). To open the window, select **Debug > Code Coverage** then click the Update button to display the information. Code Coverage is a feature of the CrossView Pro Simulator.

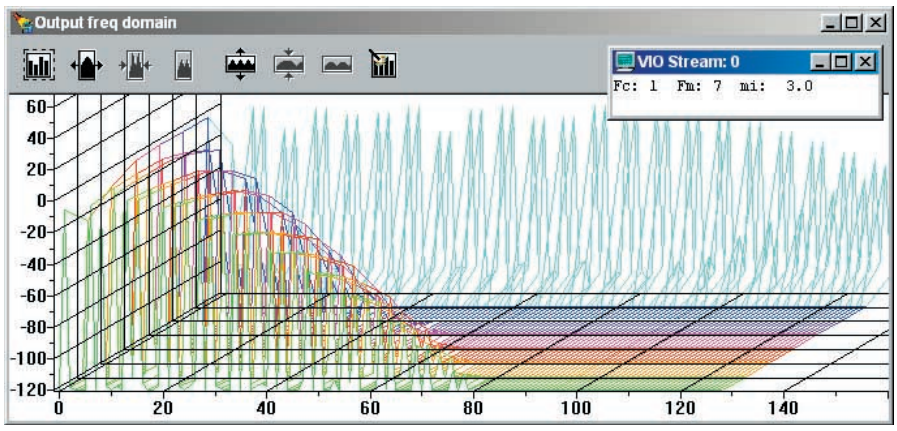
Simulated I/O

Simulated I/O allows you to observe the input and output of your program before the hardware is in place. Through the use of two special functions, I/O is automatically redirected to the CrossView Pro Debugger. You can debug most of your application's I/O dependant code even before you have your target hardware available.

Graphical Data Analysis Window

The **Graphical Signal Data Analysis** feature reduces large sets of data into meaningful visual diagrams to enable quick detection of gross errors in the input signal. The CrossView Pro Debugger analyzes the data according to pre-defined or user-defined specifications and then displays the data the way you need it. This eliminates the need for reviewing or post-processing large files of raw data. You can also view the same set of data in several ways at the same time.

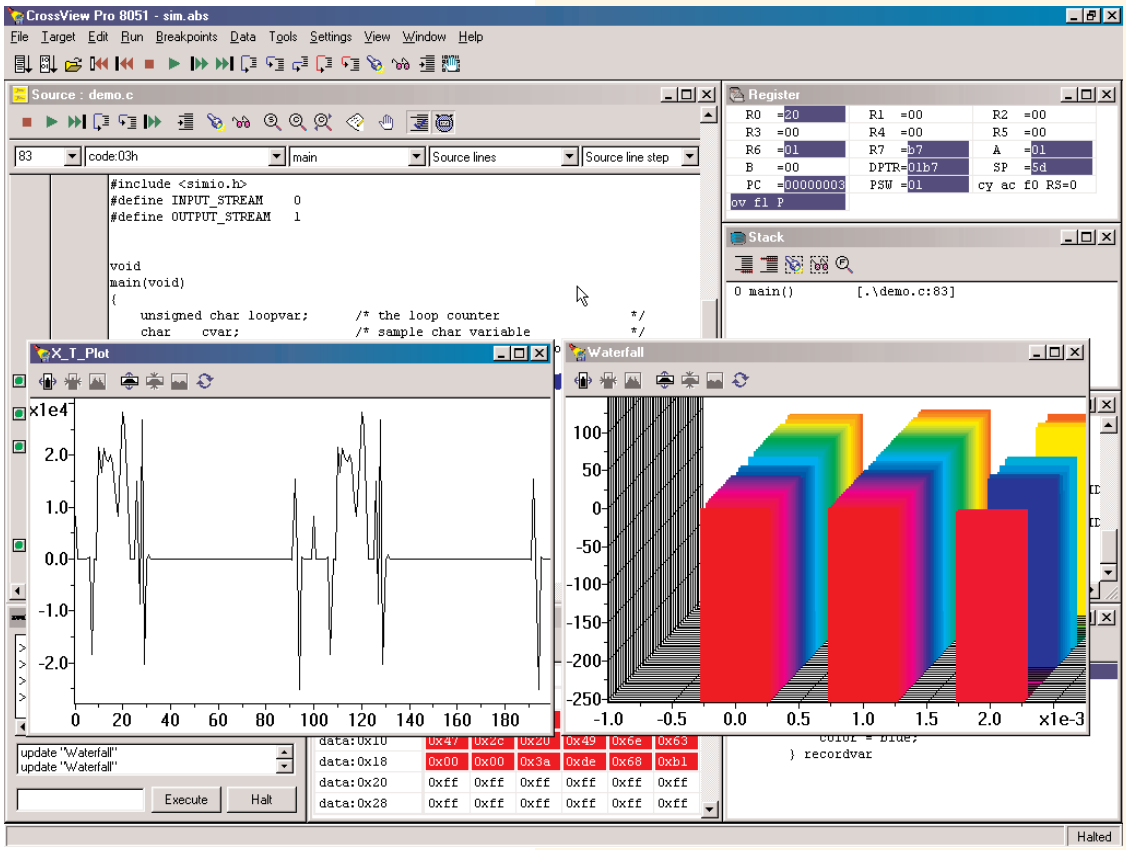
Four different analysis types are available: x-t plotting; x-y plotting; FFT power spectrum and Eye diagram. The C-language script files for these pre-defined graphs can easily be used as the basis for custom data analysis windows. Select **Debug > Data Analysis Window Setup** to activate and configure a new window with the setting you need to apply.



Recording and Playback

The CrossView Pro Debugger allows you to record your debugging session, i.e. all the commands provided with the keyboard or the mouse, to a command file. This file can be subsequently used to repeat part or the entire debugging session. This feature is known as Playback. To activate the recording section, select **Options > Record** and provide the filename you need to save the commands executed. Then press **Start.** The **Stop** button will stop the recording phase. To execute the recorded commands, select **Options > Playback > CrossView**, specify the command filename and click on the **Execute** button.

After you've downloaded your application file, you can use the local toolbar icons in the Source Window to control the execution of your debugging session. This can be done using **Step Into** (execution halted at the next instruction in the called function), **Step Over** (execution halted at the instruction immediately following the called function), **Continue** (execution of the program) and **Halt** (stop the program execution).



From the Source Window toolbar, you can always enable or disable in the CrossView Pro Simulator the **Code Coverage** and the **Profiler**, shown in two new columns on the left side next to the green/red breakpoint buttons, that will help you to set the Code breakpoint on the line you need it to be set.
If you have set the breakpoint in your code, click on the **Continue** button and the program will start the execution. It will stop if the breakpoint is reached. The windows opened on your desktop will be updated in value accordingly.

To add new breakpoints (both Code and Data), you can click on the **Edit Breakpoints** button on the Main Toolbar or select **Debug > Breakpoints**. The Breakpoints Window is now open and the **Code Breakpoint** and **Data Breakpoint** buttons are available to add new breakpoints.

Other commands to control the debugging session of your application are locally available using the right mouse click. These include **Return from Function**, **Program Reset**, **Jump to Cursor**, **Run to Cursor** and **Synchronize**, important when you are viewing in a different part of your source and you need to go back to the Program Counter address.

For detailed information on the usage of the above mentioned commands and all the other available features and commands, please consult the on-line manual which can be accessed from the main EDE toolbar.

664-TCGEETG-4