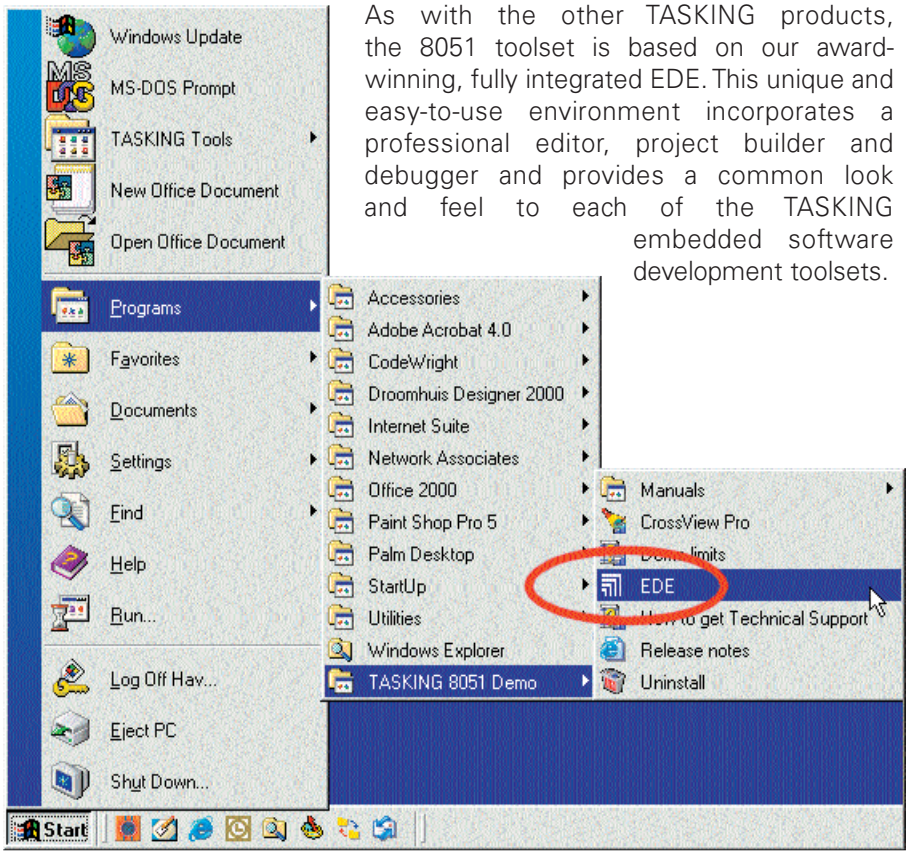
Getting Started with your 8051 Demo

TASKING
Embedded software development from Altium

Thank you for your interest in the TASKING 8051 Software Development Toolset, one of the wide range of 8-, 16-, 32-bit and DSP embedded software development products from Altium. This Getting Started tutorial will guide you through the most important parts of the toolset, and will teach you the basic steps of how to manage a project, compile / build, run and debug an example program in less than 10 minutes. It will also take you through one of the available examples, highlighting some of the features of the EDE (Embedded Development Environment) and briefly showing you how you can use the project environment.

A more detailed guide, Getting Started with your 8051 Project, is also available from the TASKING website at *www.tasking.com/support/8051/*.

THE EMBEDDED DEVELOPMENT ENVIRONMENT

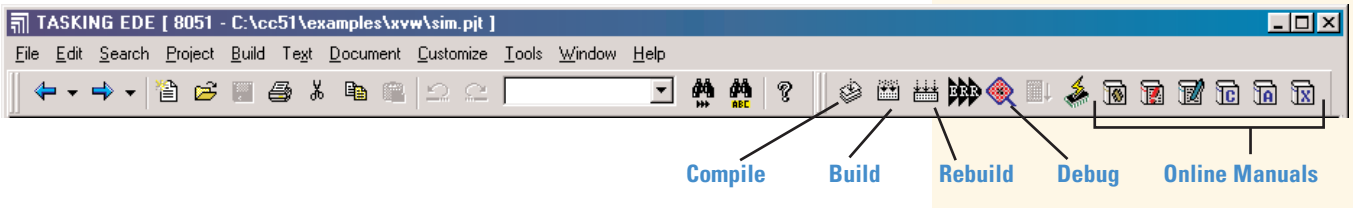


As with the other TASKING products, the 8051 toolset is based on our award-winning, fully integrated EDE. This unique and easy-to-use environment incorporates a professional editor, project builder and debugger and provides a common look and feel to each of the TASKING embedded software development toolsets.

After you have installed the 8051 toolset, click on the **EDE** icon on your desktop to launch the EDE. Alternatively, you can start the EDE from the **Start > Programs** menu. The full demo package includes all documentation in HLP, HTML and Windows HLP format; the reduced web demo comes with HLP files only to minimize download times. Two documents worth checking are the Release Notes, with information on the latest additions to the product, and the Demo Limits Summary that describes the demo versions'

TIP

Extensive on-line documentation is available, and can be accessed from either the Programs menu or by clicking on the appropriate manual buttons located on the EDE toolbar.

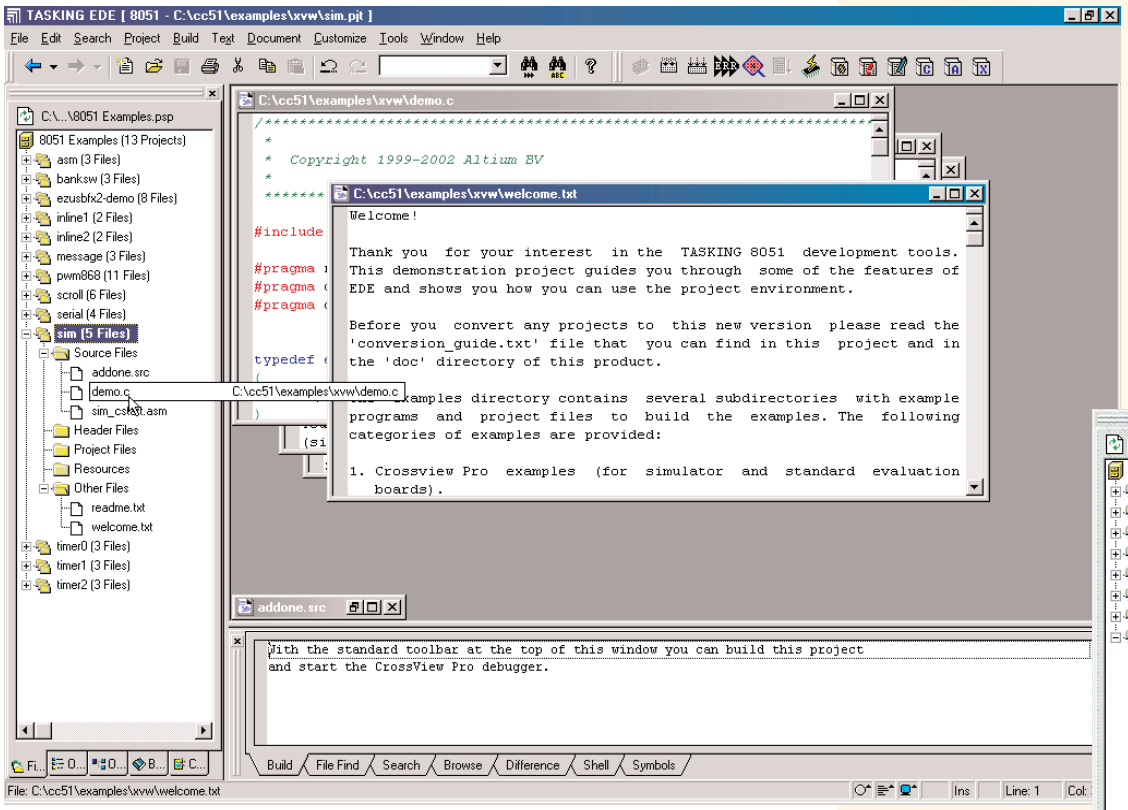


restrictions.
Once you have started the EDE the editing workbench will open up and display the Tip of the Day. After acknowledging this helpful hint, you will end up in the day-to-day coding development environment.

This 'workbench' allows you to write your application code as well as to steer tools such as the compiler and linker. It offers you advanced editing techniques like ChromaCoding text coloring and the powerful CodeSense type-ahead (auto-completion) advanced coding assistance feature.

TIP

By clicking on the [+] you can expand the trees and browse through the different files. Double-clicking on a file will open the file in the editor or bring the file forward when it was already opened in the background.



The **Examples** space on the left-hand side of the EDE window is where you manage your projects and all the files that make up your application. You will notice that here the project **sim** has been highlighted.

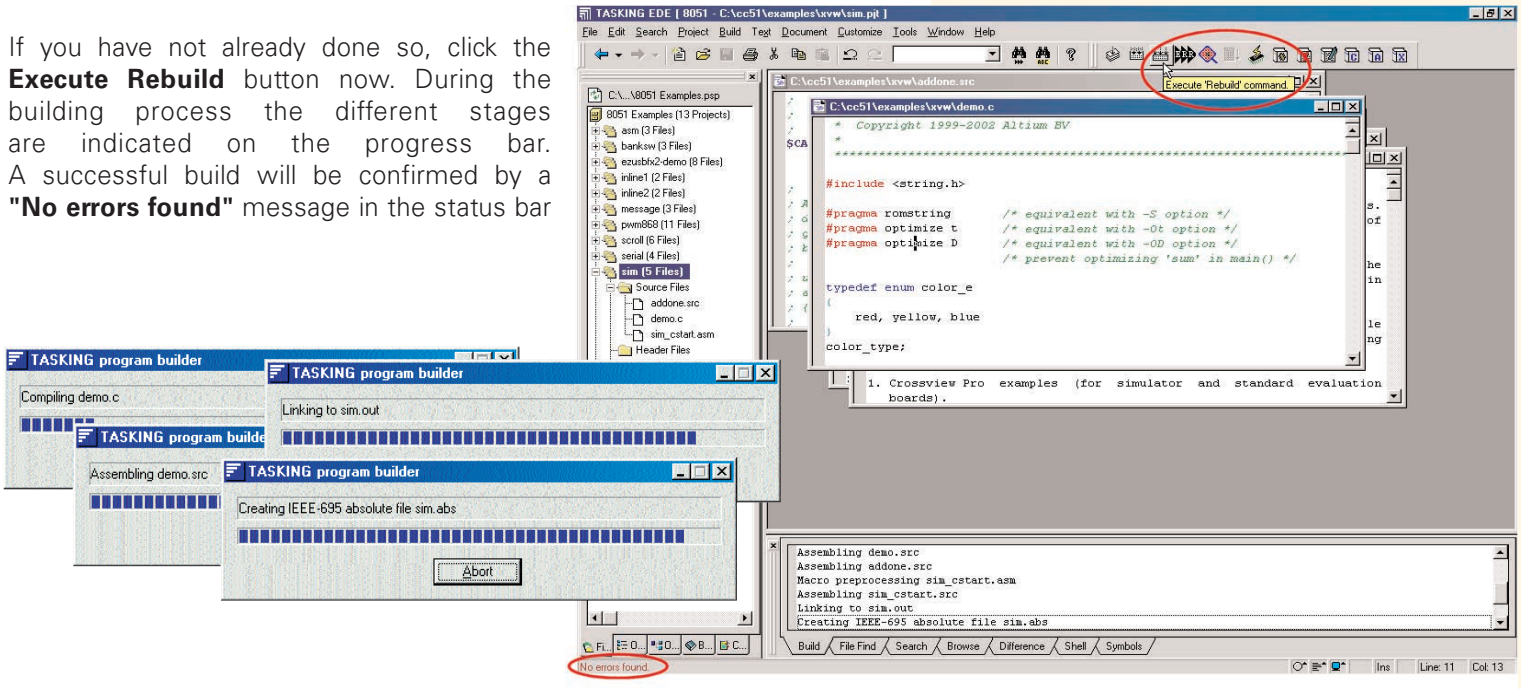
Click on **sim > Source Files** to see the source files for the sim example project. Now double-click on **demo.c** in the source files folder. You can build this sim example by clicking on the **Execute Rebuild** button on the toolbar. **Execute Rebuild** compilers, assembles and links, providing you with the end result - the absolute file (`.abs`).

The TASKING tools come with many ready-to-test example projects such as this sim example. Each project consists of C or assembly files, header files and readme files that include some additional hints or help for you. The properties of a project are saved in a special file and include the compiler, assembler and linker settings. The sim and the other example projects already include all the required settings to run.

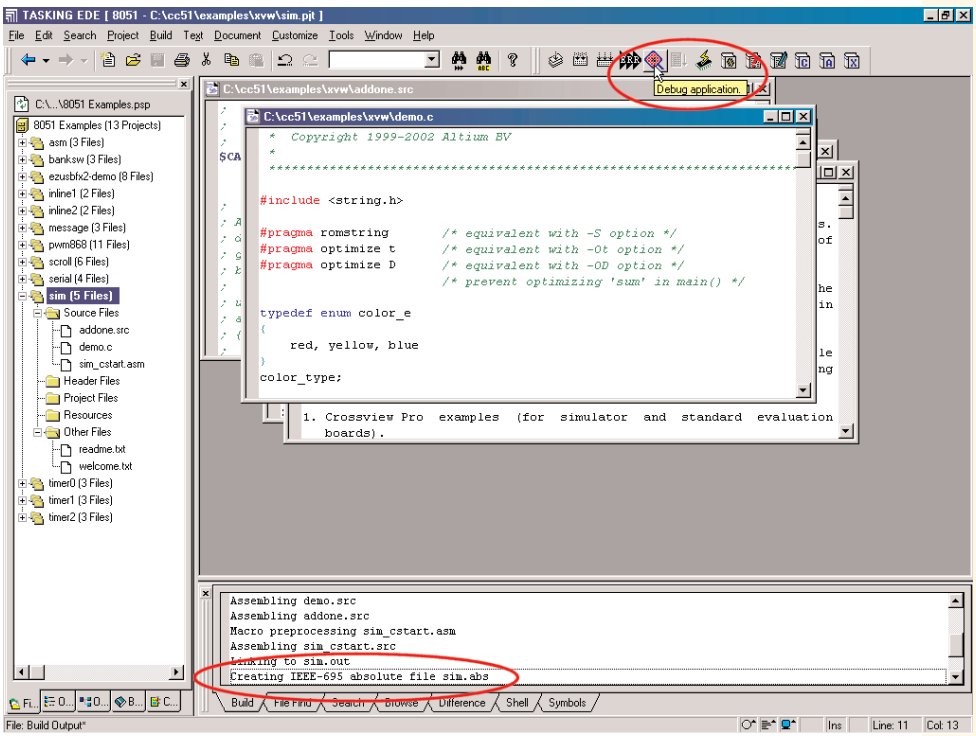
TIP

In the project management area you can quickly change to another project by right-clicking on a project name and setting it as current.

If you have not already done so, click the **Execute Rebuild** button now. During the building process the different stages are indicated on the progress bar. A successful build will be confirmed by a **"No errors found"** message in the status bar



at the bottom of the screen.
A successful build can also be recognized by the creation of an IEEE-695 absolute file, shown in the **Build** output window at the bottom of the screen - the below screenshot should more or less reflect your screen. To verify all



results in the **Build** output window you may have to scroll down.
Now you are ready to launch the **CrossView Pro Debugger**, so click on the **Debug Application** button on the toolbar. The debugger has two versions - a simulator debugger and a ROM monitor debugger. The ROM monitor debugger needs hardware, including an evaluation board, to run the project on, which is beyond the scope of this Getting Started document. The example projects are already set to start the simulator debugger

TIP

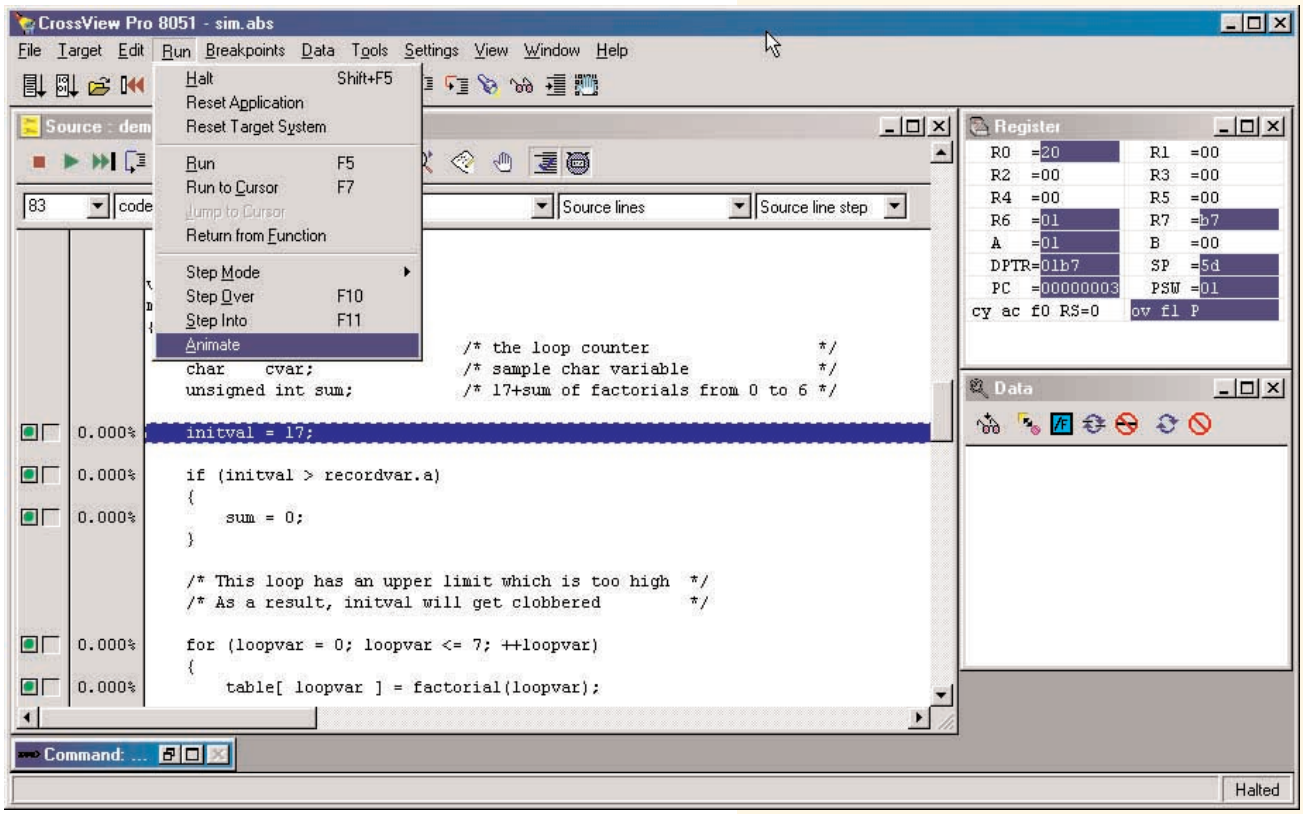
From the Project > Project Options menu you can configure the various tools to build your application correctly for your specific hardware environment. You can set the optimization level of the compiler, select your own 8051 derivative, configure your preferred programming model etc. There is no need to make any settings during this Getting Started session, but you can experiment a little afterwards.

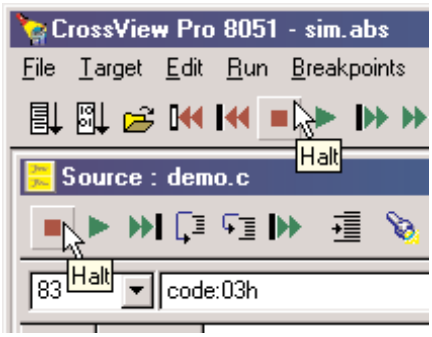
by default.

DEBUGGING WITH THE CROSSVIEW PRO DEBUGGER

After you have launched the CrossView Pro debugger, you will have the option to step through the program in either source or assembly mode. You will be able to display and modify memory, registers and either local or global variables. Other operations available, such as setting advanced breakpoints, are described in depth in the CrossView Pro user manual.

When CrossView Pro is started a few windows are opened. The most prominent window shows you the source code of your example program, and the blue horizontal line shows you where the program will start. At this stage, it is best to start the simulator in a step-by-step mode; try the **Animate** option from the **Run** menu.





The simulator runs one line, stops, updates the windows and continues again. This is a very simple way to verify whether the example runs correctly. As it can take a very long time to run all the code, however, it is better to stop the simulator at this point by clicking on either of the two **HALT** buttons and to take a closer look at some of the options of the source window.

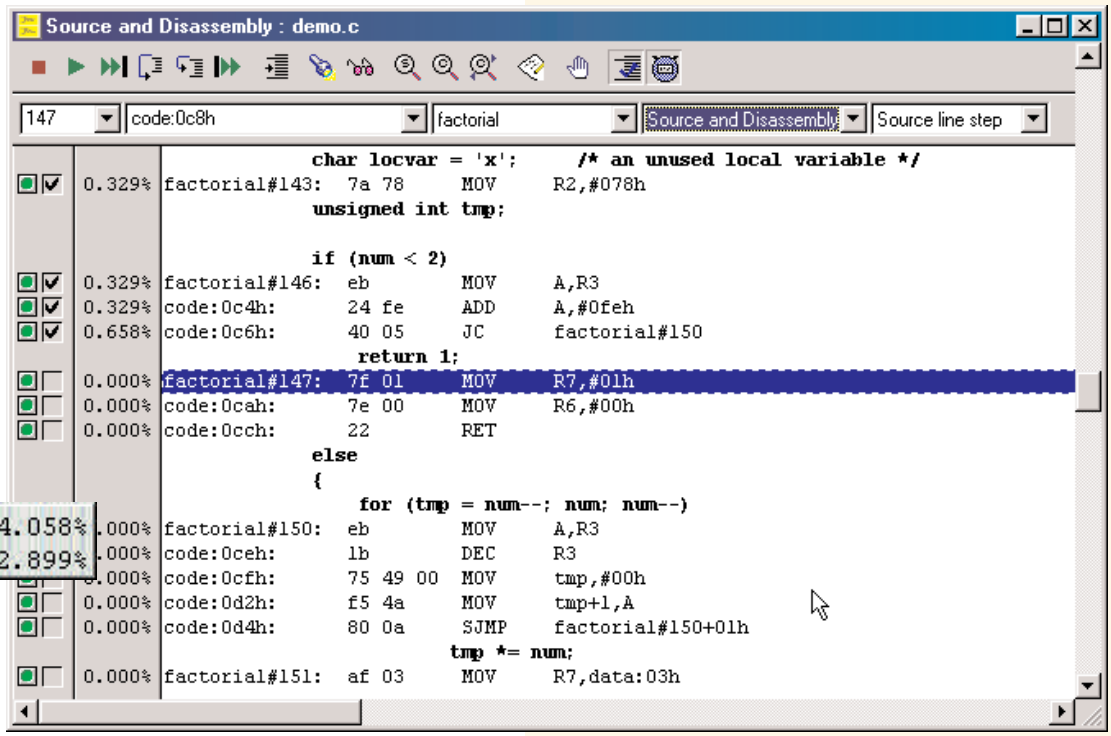
TIP

The TASKING ROM monitor debugger can be used with a range of standard off-the-shelf evaluation boards including Phytec, Infineon, Ceibo and Rigel. As the demo version of the tools include the simulator as well as the ROM monitor debugger, you can use the execution environment that you prefer.

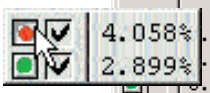
TIP

If you intend to test your application on a specific evaluation board running a ROM monitor, please read the BUILD.TXT file included in the directory EXAMPLES/XVW. This file explains how to build the application with the correct memory parameters. To activate the ROM monitor debugger, you need to change the setting from the Project > Project Options > CrossView Pro > Execution Enviroment Menu.

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.

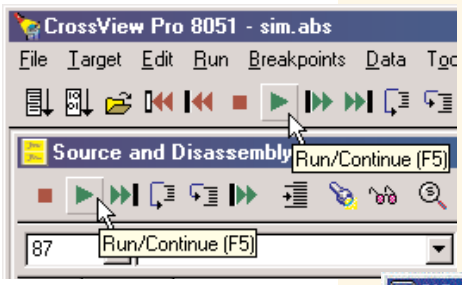


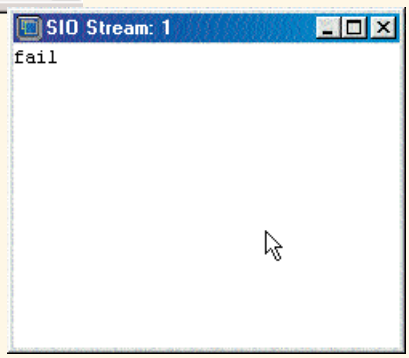
To the left of the source code are little green traffic lights that represent positions where you can set breakpoints. By clicking on one it will turn red, and you will have set a breakpoint for that particular line.



The check mark next to the breakpoint indicator shows whether a line has been reached (executed) and the percentage figure lists the relative time spent in that part of the source code.

By clicking on the **Run/Continue** button on the toolbar, the program execution will continue from the point where it was stopped and a **SIO** (Simulated I/O) window will pop up with **fail** reported in it. Relax, this does not mean that something went wrong with the debugger. Actually, everything worked perfectly and you brought this Getting Started example to a successful end!





What has happened is that the demonstration program has executed some simulated output and has activated the Simulated I/O window. The program was supposed to print **pass** here, but it shows **fail**. There must be a bug in the program and you may want to continue playing with the debugger to find the problem.

(Hint: what is the value of variable 'sum'?)

At this point a stopwatch appears and you will be in a continuous loop. Click **HALT** then **File > Exit** to close the debugger.

Now that you've made your way through this demo, you can try one of the other pre-prepared examples. You can also try to start using the 8051 development tools for your specific project. The 'Getting Started with your 8051 Project' tutorial will help you do just that. Alternatively, you can test CrossView Pro more extensively using the 'Debugging your Application with CrossView Pro' tutorial.

664-TCGEETG-3

TIP

*To display the information you need to stay in control, you can open multiple windows by selecting **View** from the menu, then choosing the one you need to activate for your debugging session.*