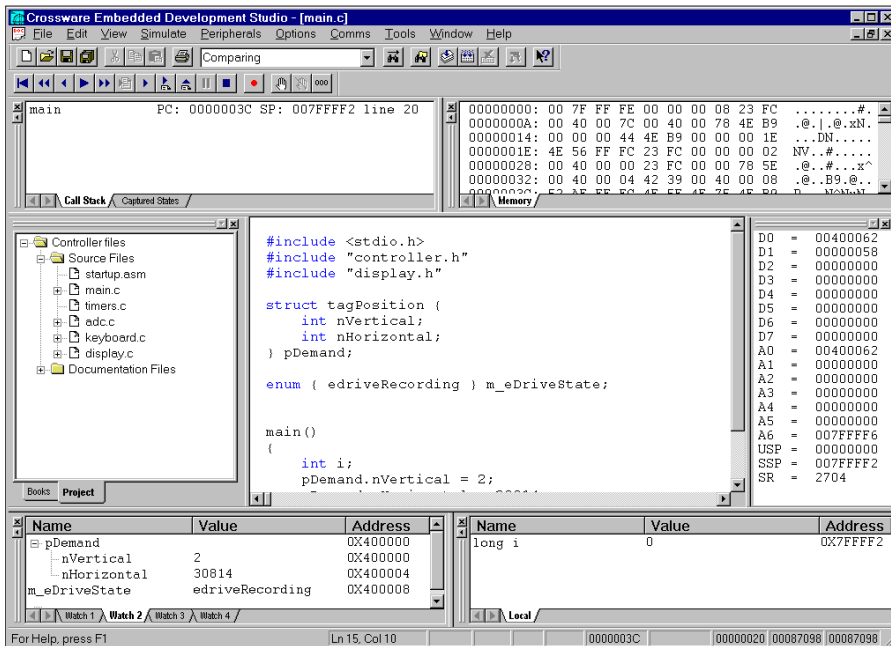


CROSSWARE® S68000NT

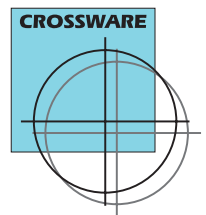
68000 Simulator for Windows

The Crossware S68000NT Simulator creates a virtual 68000 microprocessor that runs on your PC. It allows you to run your 68000 program without any 68000 hardware and watch your code execute in the graphic environment of Windows 9x, Windows NT4.0, Windows 2000 and Windows XP.



HIGHLIGHTS

- Full source level debugging
- Seamless integration with the Embedded Development Studio
- Dockable windows
- Multiple watch windows for local and global variables
- Views of all memory areas, registers and disassembled program
- State capture to capture and restore the complete internal state of the microcontroller
- Source code profiling
- Code and data coverage analysis
- Multiple cycle counters
- Operates stand-alone or with the Crossware C680X0NT and A680X0NT



Crossware Products
Old Post House
Silver Street
Litlington
Royston
Herts
SG8 0QE
United Kingdom

Telephone
+ 44 (0) 1763 853500

Facsimile
+ 44 (0) 1763 853330

Web
<http://www.crossware.com>

E-mail
sales@crossware.com

REF: SCFNT/0009

Seamless Integration with the Embedded Development Studio

The Embedded Development Studio forms an integrated environment that binds together the various Crossware tools. It automatically detects the presence of the Simulator and loads it to create a single unified application. The text windows that you see as you single step through your program are exactly the same as those you use to create and edit your source code. If you set a breakpoint as you edit your code, this is where execution will stop when you instruct the Simulator to Go.

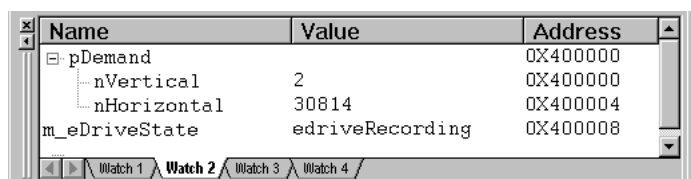
When the simulator is used 'stand-alone', the Embedded Development Studio provides a customisable interface to your tool set. You can therefore edit and keep your project up-to-date using all of the sophisticated features of the Embedded Development Studio using your existing compiler, assembler and linker.

Full Source Level Debugging

The IEEE695 records generated by the Crossware C compiler and assembler contain full debugging information. These debug records are used by the Simulator to provide comprehensive source level debugging. You can single step and trace through your C source code files, you can single step and trace through your assembler source code files and you can set breakpoints at the source code level to control the execution of the Simulator. Alternatively, if your program does not contain any debugging information or if you want to step through your C source at the assembly level, the disassembly window supports your needs.

Multiple Watch Windows for Local and Global Variables

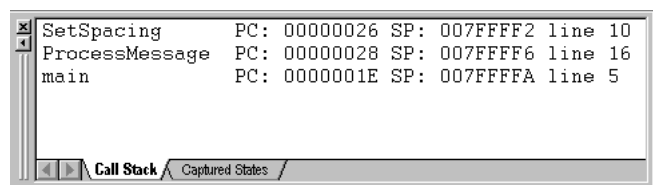
You can open up to five watch windows to display the values of your C variables. One of these watch windows will automatically display local variables and be automatically updated as variables move in and out of scope. The other four can display global and local variables of your choice. You can freeze the contents of a watch window to prevent it from being updated by the simulator. This feature allows you to compare historic and current values of your variables.



Name	Value	Address
pDemand		0X400000
nVertical	2	0X400000
nHorizontal	30814	0X400004
m_eDriveState	edriveRecording	0X400008

Call Stack

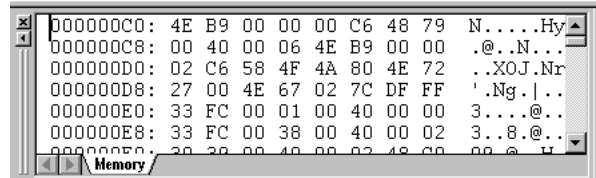
The call stack allows you to follow the progress of your program through the layers of function calls. The call stack window shows the name of each active function, and the values of the program counter, stack pointer and the source code line number reached in each function. You can navigate through the functions in the call stack - double clicking on the function name in the call stack window will take you to the location in your source code reached in that function.



SetSpacing	PC: 00000026 SP: 007FFFF2 line 10
ProcessMessage	PC: 00000028 SP: 007FFFF6 line 16
main	PC: 0000001E SP: 007FFFA line 5

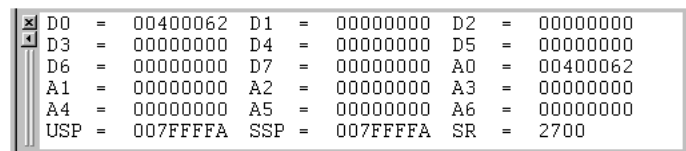
Views of Memory Areas

You can open multiple views and watch all memory areas, setting breakpoints when memory locations are written, read or both. Coloured highlighting shows you which locations are changing and optionally you can enable data coverage so that you can clearly see which memory locations have been accessed and which have not. You can edit any memory location in hexadecimal or ASCII format.



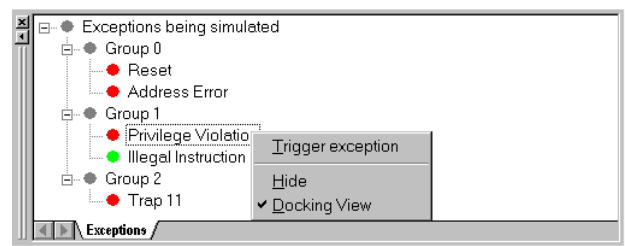
View Named Registers

You can open a view to display and edit named registers.



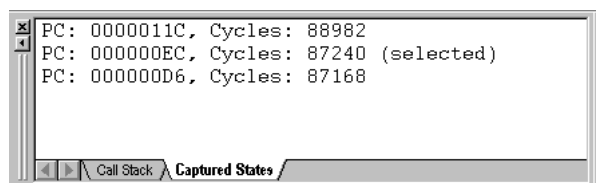
Exception Simulation and Views

You can either trap or simulate exceptions. You can choose which exceptions to trap and which to simulate on an individual basis. Exceptions that are being simulated can be triggered either by a condition that occurs in your program (eg. address error or trap) or from a tree view that displays the exception being simulated.



State Capture

The state capture facility allows you to store the complete internal state of the simulated microcontroller. You can capture the internal state any number of times and select and view them at a later stage. When you select and view a captured state, all of the memory views are updated to show the captured state. You can fully restore a captured state. Then, when you next run the simulator it will start with all conditions, memory, cycle counters, program counter, etc restored to the values current at the moment you captured the state. You can use this feature to repeatedly run a small segment of your code.



Source Code Profiling

As the simulator runs your program, it can optionally capture information about the execution history. It can use this information to produce an execution profile of your source code. This will tell you how much time your program spent executing in the different parts of your code. The profile results are listed in a text window, sorted in accordance with your selected criteria. A double click on a listed item will take you to the relevant line of your source code.

Code and Data Coverage

The information captured during execution can also be used to provide a coverage analysis. Code coverage will show you which parts of your code have been executed and which have not. Data coverage will show you which memory locations have been accessed during execution.

Multiple Cycle Counters

The status line contains three cycle counters. One of these displays the number of cycles executed since the start of the program, another displays the number of cycles executed since the last command was issued (such as Step, Trace or Go). The third is a trip counter and displays the number of cycles executed since you last reset it to zero.

Memory

The simulator allocates memory in 64k blocks as required by your program. The maximum memory is therefore limited only by the amount of memory on your PC.

Program File Formats

The file formats supported for the 68000 program code are Motorola S-records, IEEE695, Intel hex. and Binary.

Host System Requirements

IBM compatible PC with an Intel Pentium or above running Windows 9x, Windows NT4.0, Windows 2000 or Windows XP.