



Installing CSIM 20 for Microsoft Windows and Visual Studio

Mesquite Software, Inc.
PO Box 26306
Austin, TX 78755-0306
(800) 538-9153 or (512) 338-9153
info@mesquite.com
www.mesquite.com

CSIM is copyrighted by Microelectronics and Computer Technology Corporation, 1985-1994. Its use is covered by the CSIM Software License Agreement included with the software. This manual is copyrighted by Mesquite Software, Inc. It may not be copied without written permission from Mesquite Software, Inc.

Installing CSIM 20 for Microsoft Windows and Visual Studio

Introduction

This document describes how to install and run CSIM 20 on a Microsoft Windows system with Microsoft Visual Studio. Please note that this document assumes that users possess a basic proficiency in using Microsoft Visual C++.

CSIM 20 also supports the Cygwin environment on Windows systems. For information on using CSIM 20 with Cygwin on Windows, please refer to the document, *Installing CSIM 20 for UNIX / LINUX / MAC OS X / Cygwin Systems*.

Installation

The CSIM 20 package includes a set of directories and files that are either downloaded from the Internet or shipped on a compact disk (CD). Both distribution methods contain the same CSIM 20 files and documentation. If you downloaded CSIM 20 from the Mesquite website, you should have received a .zip archive file. You will need a program capable of extracting ZIP archives.

If you obtained a CD copy of CSIM 20, simply copy the contents of the CD to a location of your choice on your system's hard drive. For Internet download packages, simply extract the contents of the downloaded archive file to a location of your choice on your system's hard disk. For the purposes of this document, we will assume the CSIM 20 files have been placed in `C:\csim20`.

The CSIM 20 distribution contains a hierarchy of directories as follows:

```
Documentation
Windows
  <32-bit/64-bit>
    <build environment>
      <c/c++>
        Debug
        Release
        lib
          deplib.lib
          rellib.lib
          csim.h
        <project files>
```

where `<build environment>` refers to the version of Microsoft Visual Studio.

Building and Running CSIM 20 Programs

The CSIM 20 directory contains pre-configured project files for use with Microsoft Visual C++. There are separate projects for the C and C++ languages, for the various supported versions of Visual C++, and also for 32-bit and 64-bit architectures. The language, Visual C++ version, and the architecture are indicated in the directory path names where the project files are located. For example, the project files for the C language with Visual Studio 2008, 64-bit are located within the folder `csim20\64-bit\vs2008\c`. To begin using CSIM 20 with this configuration, simply open the Visual Studio solution file (`c.sln` for C or `cpp.sln` for C++) with the appropriate version of Visual C++. On most systems with only one version of Visual C++ installed, you may simply double-click the `.sln` file from Windows Explorer. However, on systems with multiple installed version of Visual C++, you will need to take care that the correct project is opened in the correct version of Visual C++.

Once you have opened the project in Visual C++, you can quickly begin by building the included test program. Upon initial installation, the Visual C++ project will be pre-configured to build and run a simple program called `Ex2.c` or `Ex2cpp.cpp`. This program simulates a simple M/M/1 queue and writes the results of the simulation to the file `csim.out`. In most cases, you should be able to simply select “Build Solution” from the “Build” menu. After the build completes, the executable will be located in the `Debug` folder under the project folder. You should then be able to run the executable, and the CSIM output report will be written to the `csim.out` file, which you may then examine.

To run the other included sample programs, or to begin building and running your own CSIM models, simply remove the `Ex2.c` source file from the Visual C++ project and insert your own source files. To do this, find the Visual C++ “Solution Explorer” pane, usually located on the left of the Visual Studio interface. Under “Source Files,” select the `Ex2.c` file and select “Remove” from the “Edit” menu, or simply press the Delete key. You may then add your own source files by selecting either “Add New Item” or “Add Existing Item” from the “Project” menu.

It is strongly recommended that you keep copies of the original Visual Studio projects from the CSIM 20 installation before you begin modifying the projects to fit your needs, in case the project is accidentally modified in a way that causes CSIM to cease building properly. You will then be able to refer back to the original project configuration for comparison.

Creating New Visual C++ Projects for CSIM Programs

If, for whatever reason, you need to create your own Visual C++ project for building CSIM programs, this is fairly straightforward. In essence, you simply need to create a project that includes the CSIM 20 library file in the linker input and a pointer to the folder containing the correct CSIM include file (`csim.h` for C and `cpp.h` for C++). In Visual Studio 2008, create a new C++ Win32 Console Application. When creating the new project, make it an empty project. Then, select “Properties” from the “Project” menu. Then, in the left pane of the project properties dialog, select “Configuration Properties,” then “Linker,” then “Input.” In the right pane, in the “Additional Dependencies” field, type the relative path to the `deplib.lib` library file, or for the Release configuration, the `rellib.lib` file. Finally, select “General” under Configuration Properties and then the “Additional include directories” item; a dialog window opens that allows you to enter the path name for the folder containing the include file(s) for CSIM. In the default CSIM project, the include directory is “lib”.

For the C++ version of CSIM, you will also need to be sure that 'CPP' is included in the preprocessor definitions by select “Properties” from the “Project” menu and then choosing “Preprocessor.” In the “Preprocessor Definitions” field, add 'CPP', being sure to separate this definition from the preceding ones with a semicolon.

The Release Version of the Library

In some cases, it is desirable to use the “Release” version of the CSIM 20 library, along with a “Release” version of the model. The reasons for this include the fact that the Release version of the model will probably execute in less time (faster). To do this, simply link to `lib\rellib.lib`, in place of the `lib\deplib.lib`. The “Release” configuration mode in the provided projects is already set up to do this. There is no Release version (`rellib.lib`) for the 64-bit version for Visual Studio 2008.

When using your own Visual C++ projects with the “Release” version of the library and model, it is important to note that many compiler optimizations cannot be used with CSIM programs. When configuring your CSIM project in Visual C++, you will need to ensure that compiler optimizations and function inlining are disabled under the compiler settings.

In Visual C++ 2008, Visual C++ Express, and Visual C++ 2005, select the “Project” menu, then choose “Properties.” In the left pane, expand “Configuration Properties,” then “C/C++,” then select

“Optimization.” In the right pane, set “Optimization” to “Disabled,” set “Inline Function Expansion” to “Only __inline,” and set “Enable Intrinsic Functions” to “No.”

In Visual C++ 6, select “Settings” from the “Project” menu. Navigate to the “C/C++” tab, and select “Optimizations” from the “Category” drop-down menu. Set “Inline function expansion” to “Disable.”

32-bit vs. 64-bit Code

CSIM 20 now supports 64-bit platforms in the professional version. The primary advantage of building CSIM programs with 64-bit support is that it allows for a much larger address space, which, in turn, allows much larger and more complex models to be built and run. Most use cases will not encounter the limits of 32-bit mode, but for very large models, 64-bit compilation may be required. In addition to allowing larger models, compiling CSIM programs with 64-bit support will often yield better performance, but this result is not always true. For example, on the x86 architecture, x86-64 offers many more improvements and optimizations than just 64-bit mode. Thus, on x86-64 platforms, compiling for 64-bit will usually result in faster performance of CSIM models. However, this performance improvement may not occur on all platforms. For detailed help determining the need, capability, and advantages of using 64-bit mode on your specific platform with your specific usage needs, please feel free to contact Mesquite Software.

Writing CSIM 20 Programs

The major objects in a CSIM 20 simulation model (program) are the resources of the model and the processes (entities) which will “use” these resources. Thus, designing a CSIM 20 model usually proceeds as follows:

1. Determine the resources needed by the model. In CSIM 20, these resources can be of the following types:

- Facilities – single server, multiserver-server or sets of facilities
- Storage blocks – either a single block or a set of blocks

These resources are usually “global;” that is, declared outside of the context of a procedure. In addition, these resources must be initialized prior to their use.

2. Determine the processes of the model:

- The first process is normally named *sim*. If this is not done, then the programmer must provide a *main* procedure, which is used instead of the *main* provided in the CSIM 20 library. *Sim* is called with the arguments *argc* and *argv*, just as *main* is called in a normal C program.
- A CSIM 20 process is a C procedure which executes a *create* statement. This *create* statement is executed at the beginning of the procedure. There is no predefined relationship between CSIM 20’s processes; any process can “invoke” any other process (except *sim* should only be invoked once).
- A CSIM 20 process is not an NT process. It is like a thread or lightweight process. A CSIM 20 process can not return a value (it is not a function).
- A CSIM 20 process terminates by doing a normal procedure exit: it either executes a “return” statement or it flows out of the procedure.
- A CSIM 20 process can synchronize its interactions with other processes by using events, mailboxes, facilities and storage blocks.

The *CSIM 20 User’s Guide* goes into more detail on how to write processes.

Testing CSIM 20 Programs

Testing and debugging CSIM 20 programs can proceed in several ways:

- A CSIM 20 program is a real C/C++ program, so *printf* statements, etc., can be used to give the state of various variables and procedures.
- A CSIM 20 event trace gives a detailed account of the actions of each process in the model. It can be turned on in two different ways:
 - The `-T` command line argument, which causes the program to start with the trace on. To specify this on an NT command line, specify “`-T`”
 - For a “Win32 Console Application,” parameters for the program are specified in the Debug Dialog of the Project -> Settings menu item.
 - The *trace-on* and *trace-off* statements in a CSIM program can selectively activate and deactivate the event trace.
- The *dump_status* statement prints the current status of all resources and processes in the model. There are separate status routines for facilities (*status_facilities*), storage blocks (*status_storages*), etc.

The CSIM 20 library routines all send their output to one of three files. The files all default to `stdout`, but can be redirected by using one of the following statements:

- set_output_file* – redirects reports and status information
- set_trace_file* – redirects the event trace
- set_error_file* – redirects error messages

Note that all CSIM documentation is available on the Mesquite website at <http://mesquite.com/documentation/index.htm>

For Further Information

If you have difficulties installing the CSIM 20 library or with writing, testing and executing CSIM 20 programs, please contact:

Customer Support
Mesquite Software, Inc.
PO Box 26306
Austin, TX 78755, USA

Tel: 1-800-538-9153 (for US customers), 1-512-338-9153 (for all others)

Fax: 1-512-338-4966

E-mail: info@mesquite.com

www.mesquite.com