

# Project Tutorials

---

The tutorials of this chapter introduce you to the CodeWarrior tools and shows you how to use them when developing software for the NINTENDO GAMECUBE game console.

Click any of the following links to jump to the corresponding section of this chapter:

- [Creating an IDE Project](#)
- [Debugging an IDE Project](#)
- [Debugging Command Line Tool ELFs](#)

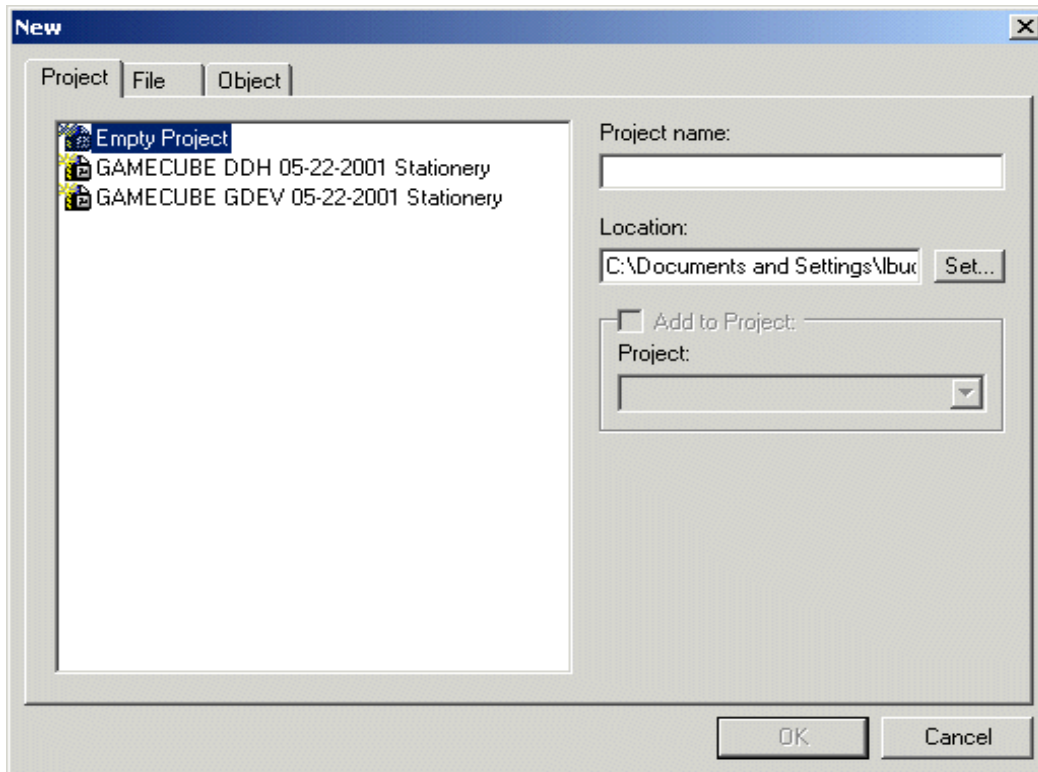
## Creating an IDE Project

This section shows you how to create a project, and then compile and link your code.

1. Launch the CodeWarrior IDE.
  - a. Locate the icon for the CodeWarrior IDE.
  - b. Launch the IDE.
2. Create a project from stationery by selecting **File > New** in the CodeWarrior Menu Bar. The Stationery is a program template that is included with your software.

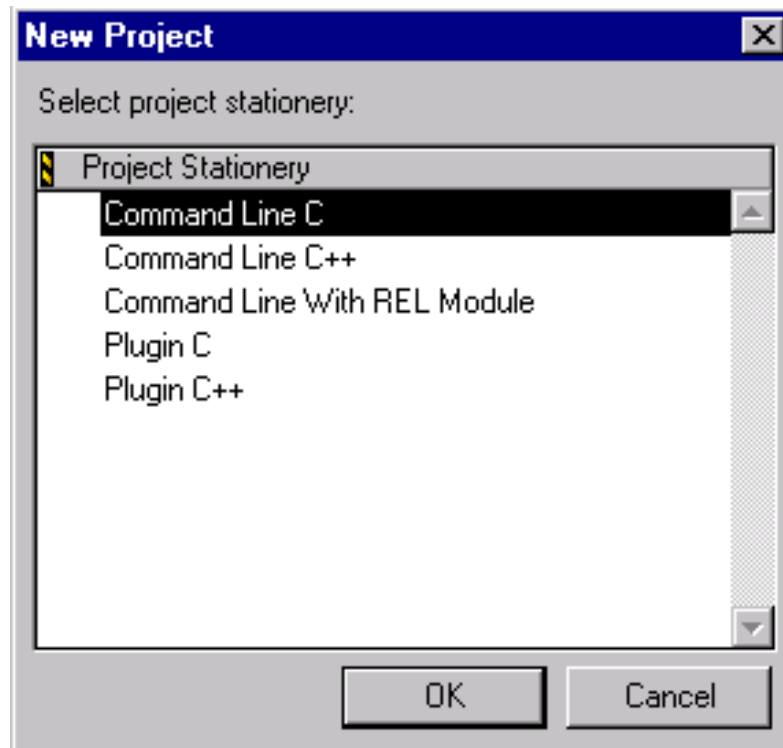
3. Select the **GAMECUBE HW/SDK**, either GDEV or DDH, depending on which hardware you are using. ([Figure 3.1](#)).
  - a. Name the new project Tutorial.
  - b. Click the **OK** button.

**Figure 3.1** Selecting GAMECUBE Stationery



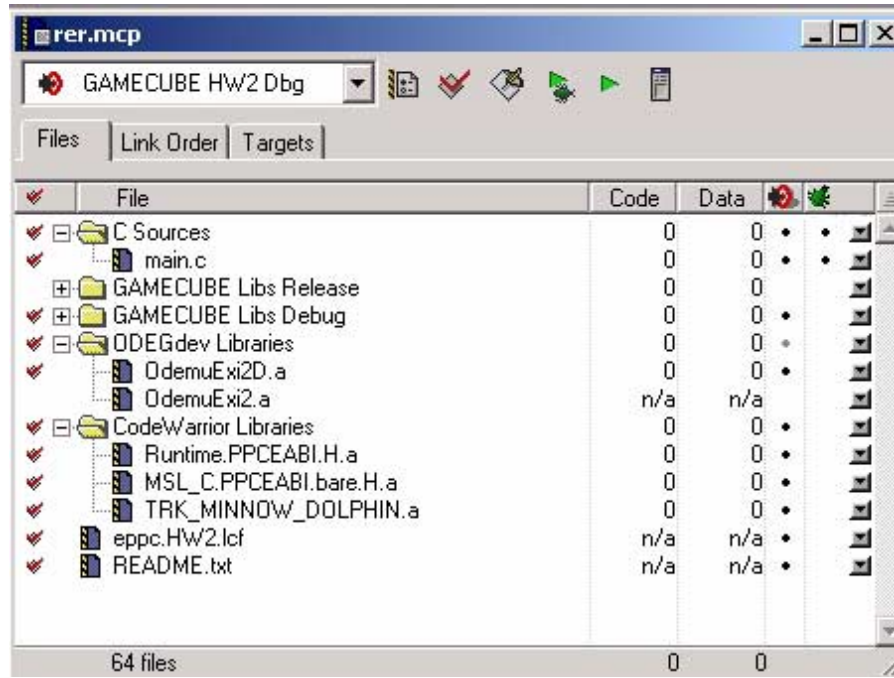
4. Select the **Plugin C** or **Command Line C** project stationery ([Figure 3.2](#)). For more information on command line adapters see [“Command Line Adapters” on page 71](#).
5. Click the **OK** button.

Figure 3.2 Selecting Command Line C Language Stationery



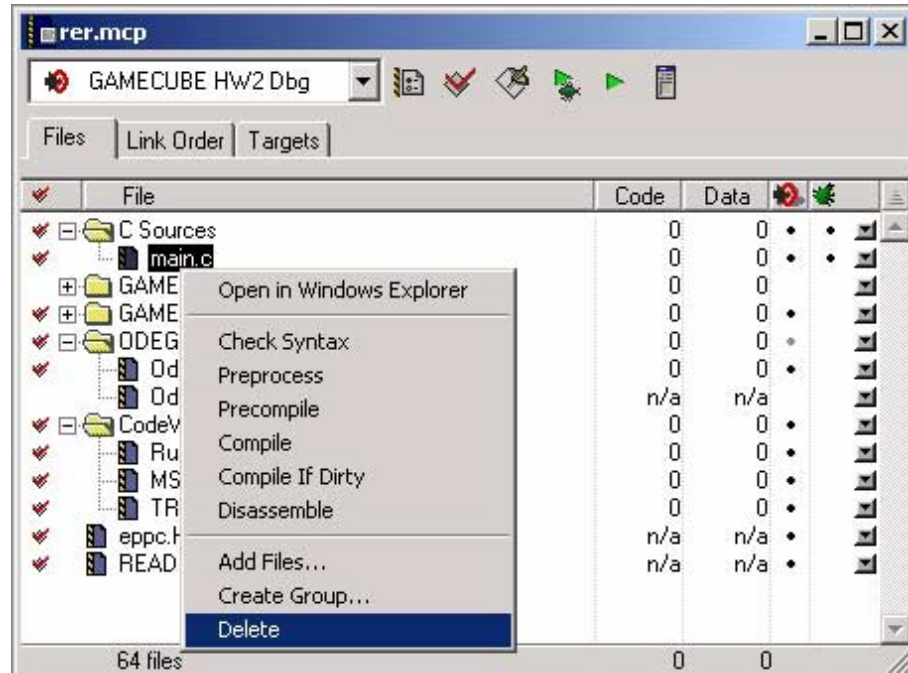
6. Click the hierarchy control of the **C Sources** group to view its contents ([Figure 3.3](#)). Doing this enables you to add and view source files.

Figure 3.3 Contents of the C Sources Group



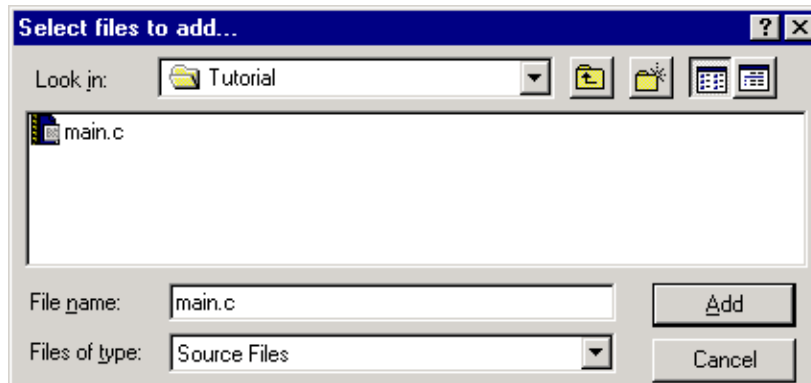
7. Right-click `main.c` ([Figure 3.3](#)).
8. Select **Delete** ([Figure 3.4](#)).

**Figure 3.4** Deleting `main.c`



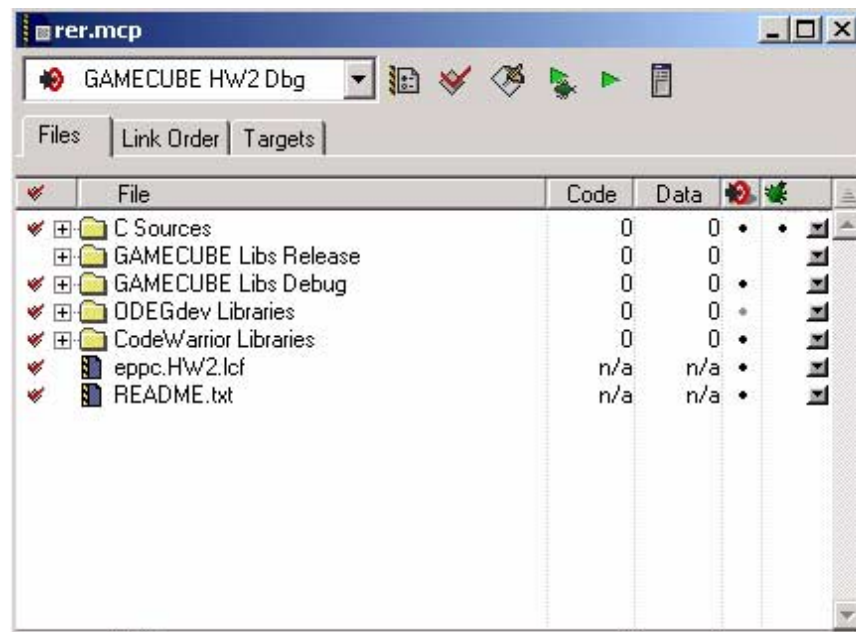
9. Select **Project > Add Files...** to locate the `main.c` in the tutorial folder.
  - a. For GDEV hardware, the tutorial folder is in the path:  
Examples\GAMECUBE GDEV\Tutorial.
  - b. For DDH hardware, the tutorial folder is in the path:  
Examples\GAMECUBE DDH\Tutorial.
10. Select the file.
11. Click **Add** ([Figure 3.5](#)).

**Figure 3.5** Adding the New main.c to the Project



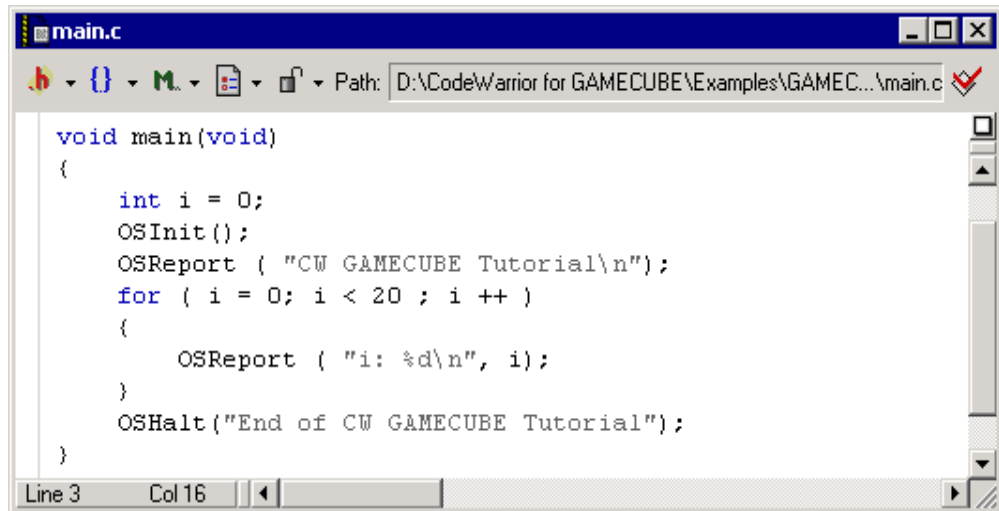
12. Drag and drop `main.c` to the **C Sources** group ([Figure 3.6](#)).

**Figure 3.6** Organizing Project Files by Category



13. Double-click `main.c` to view its contents. ([Figure 3.7](#)).

Figure 3.7 main.c Program

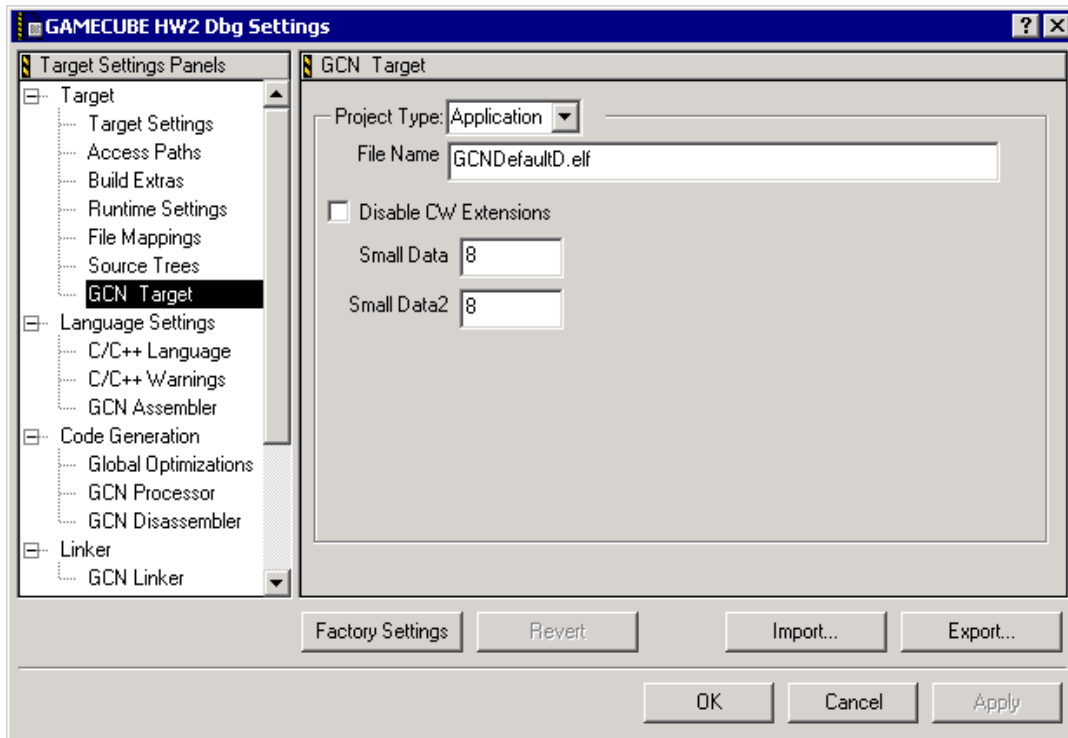


```
void main(void)
{
    int i = 0;
    OSInit();
    OSReport ( "CW GAMECUBE Tutorial\n");
    for ( i = 0; i < 20 ; i ++ )
    {
        OSReport ( "i: %d\n", i);
    }
    OSHalt("End of CW GAMECUBE Tutorial");
}
```

14. Set the target options.

- a. Select **Edit > GAMECUBE Target Settings** to change and inspect these options.
- b. Inspect the **GCN Target** settings panel ([Figure 3.8](#)). The output filename is `GCNDefaultD.elf`.

Figure 3.8 Inspecting GCN Target Settings



15. Generate debugging information.

- a. Instruct the compiler to include DWARF debugging symbolics when it compiles the output file.
- b. Click the space in this column allocated to `main.c` (Figure 3.9) (represented by a green bug).

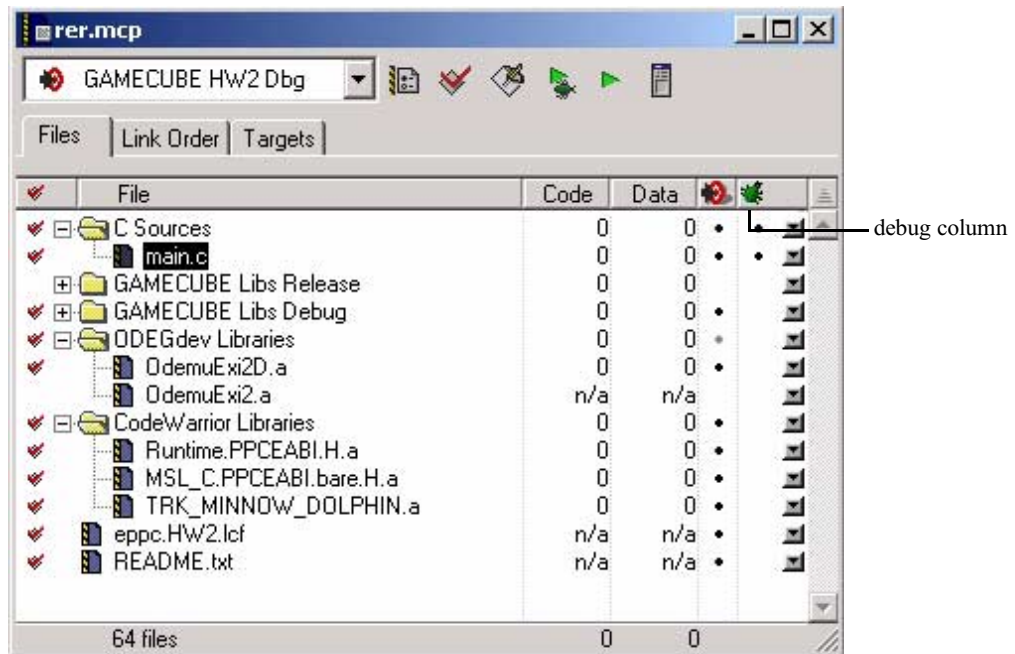
The bullet point that appears across from the filename means the compiler generates DWARF data for this file.

---

**NOTE** To get debug info in the final elf, **Generate DWARF info** must be enabled in the Linker panel.

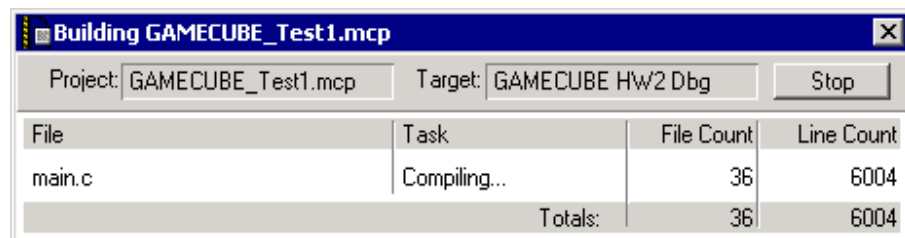
---

Figure 3.9 Generating Debugger Information for main.c



16. Select **Project > Make** to compile and link your project ([Figure 3.10](#)).

Figure 3.10 Compiling and Linking the Project

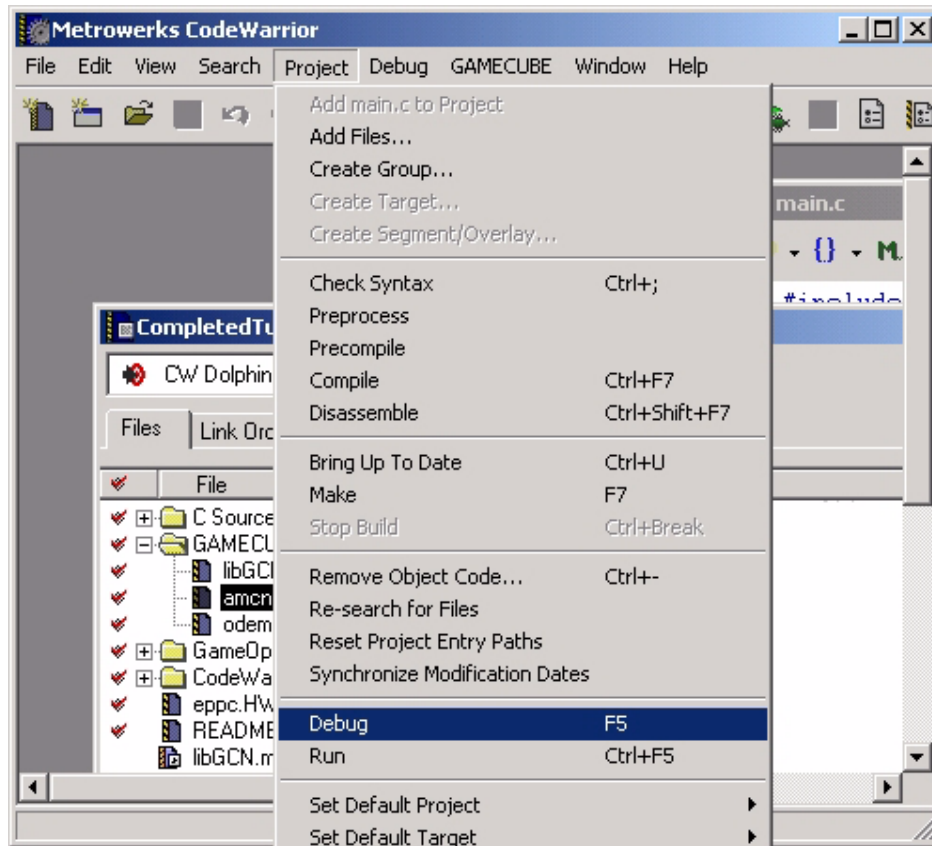


## Debugging an IDE Project

This section introduces you to the CodeWarrior debugger.

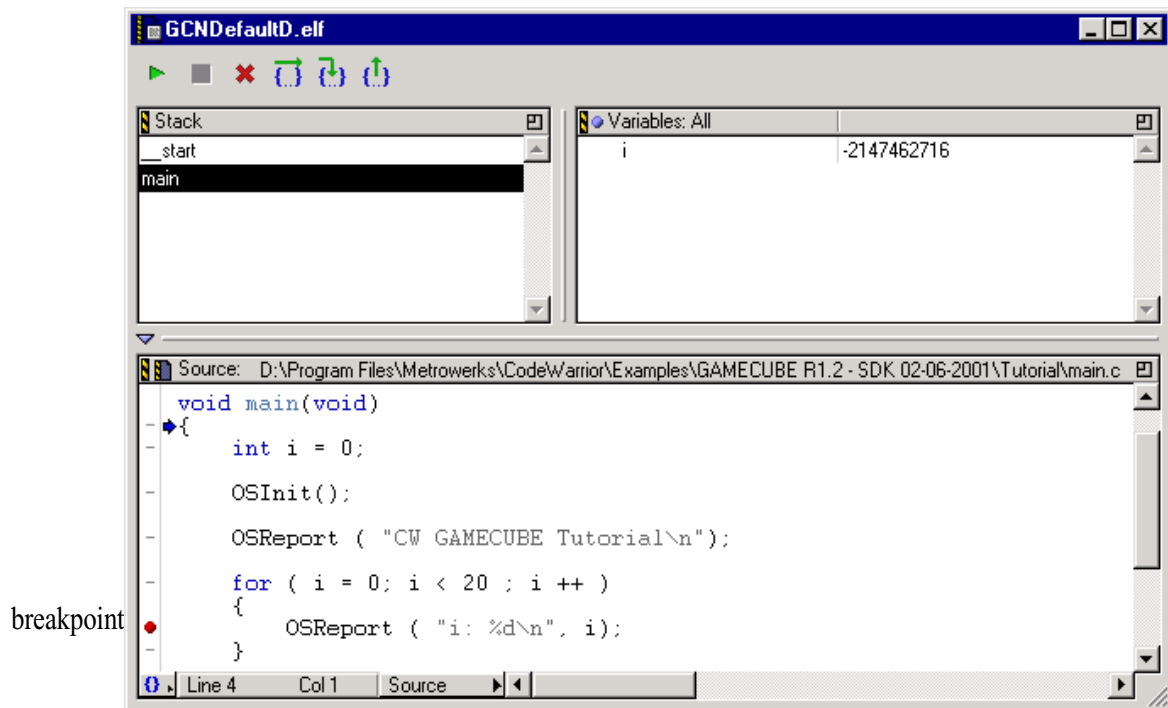
1. Select **Project >Debug** ([Figure 3.11](#)).

**Figure 3.11** Enable Debugger



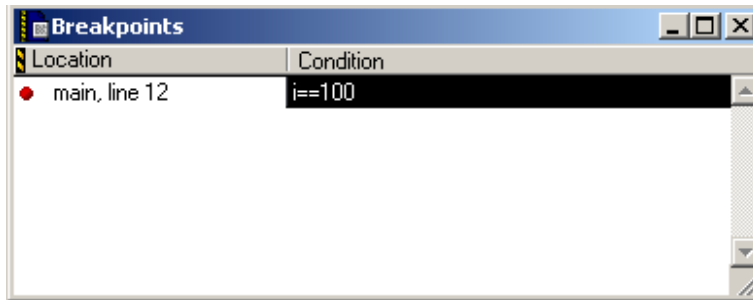
2. Set a conditional breakpoint.
  - a. First set a regular breakpoint then modify its parameters.
  - b. Set the breakpoint within the loop by clicking the tick-mark associated with the `OSReport()` function. (Figure 3.12). The red dot represents the breakpoint.

Figure 3.12 Setting a Breakpoint in the Loop



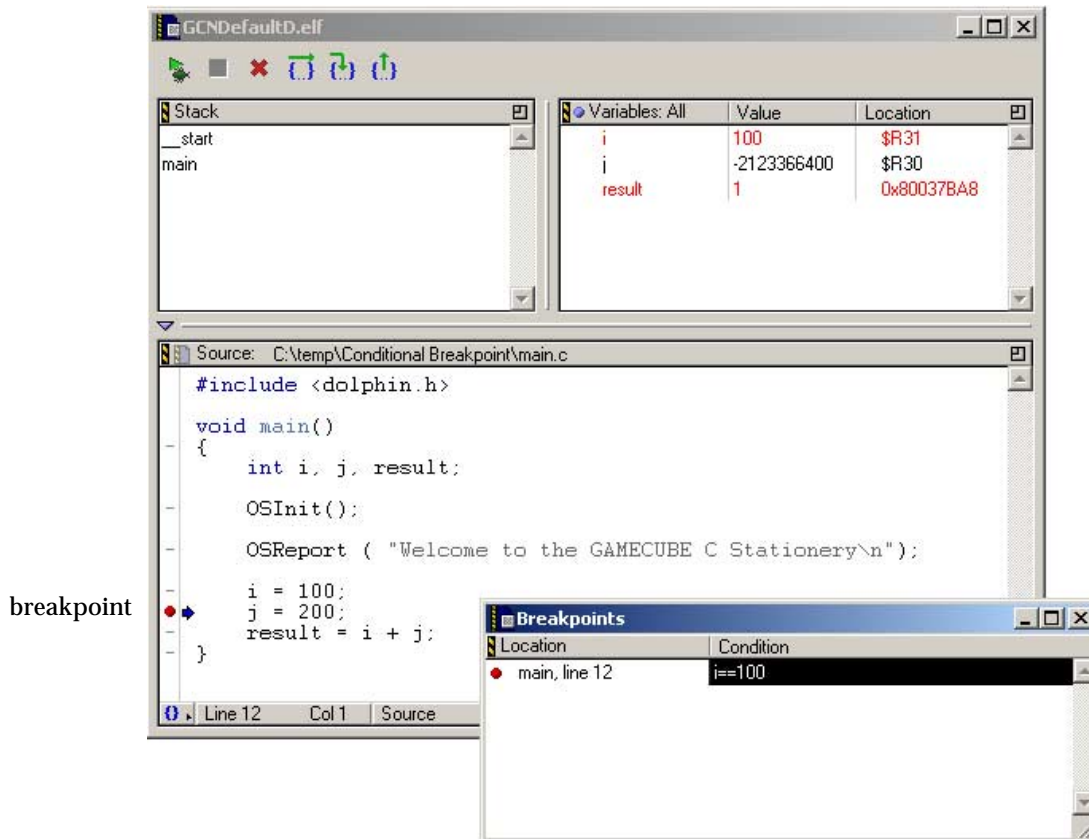
3. After setting the breakpoint, select **View > Breakpoints** Window. This breakpoint must be active when the loop counter `i` equals 100.
4. Make this breakpoint active by clicking in the **Condition** column opposite the breakpoint and enter `i==100` as the condition (Figure 3.13).

Figure 3.13 Setting the Breakpoint Condition



5. Click **Run** to run your application from the debugger.  
The breakpoint activates when *i* equals 100, verified by a glance at the variables pane (Figure 3.14).

Figure 3.14 Conditional Breakpoint at Work



6. Click **Kill** to quit your application and the debugging session.

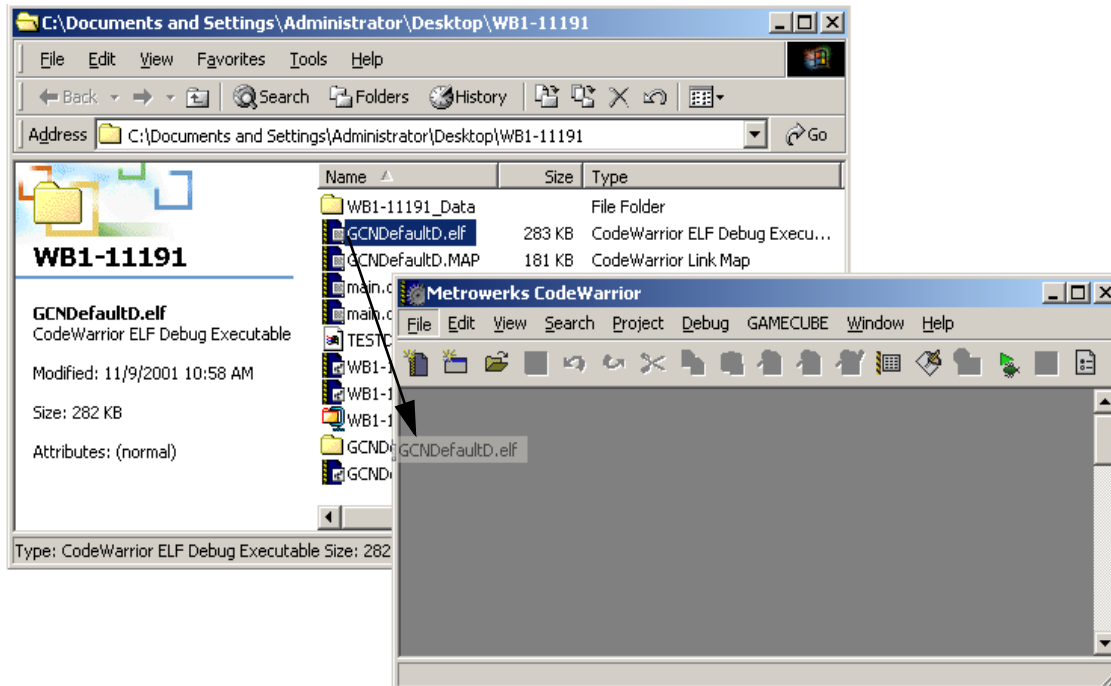
## Debugging Command Line Tool ELF's

The IDE cannot debug an executable file without the settings panel information found in an IDE project. Executable files do not have any IDE projects associated with them. Therefore, it is difficult to debug executables built with command line tools.

The IDE works around this limitation by loading a boilerplate Extensible Markup Language (XML) project template when it encounters an orphaned ELF file. The CodeWarrior IDE uses the XML template to automatically create a project that the IDE can debug.

1. Open the orphan ELF file with the IDE.
2. Drag the `VerifySetup_Orphan.elf` file from the `OrphanElf` folder onto the CodeWarrior IDE ([Figure 3.15](#)).
  - a. For GDEV hardware, the `OrphanElf` folder is in the path: `Examples\GAMECUBE SDK - GDEV\OrphanElf`.
  - b. For DDH hardware, the `OrphanElf` folder is in the path: `Examples\GAMECUBE SDK - DDH\OrphanElf`.

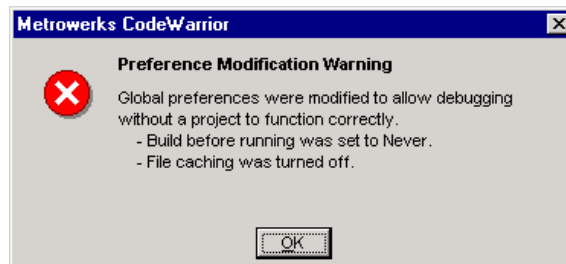
Figure 3.15 Dragging the ELF File Onto the IDE



The IDE automatically creates a project for this ELF file. Do not compile this project. It is intended only to provide the settings for the debugger.

3. The first time you open an orphan ELF file, the IDE disables the file caching and build-before-run options. Confirm these changes by clicking **OK** when the warning window in [Figure 3.16](#) appears.

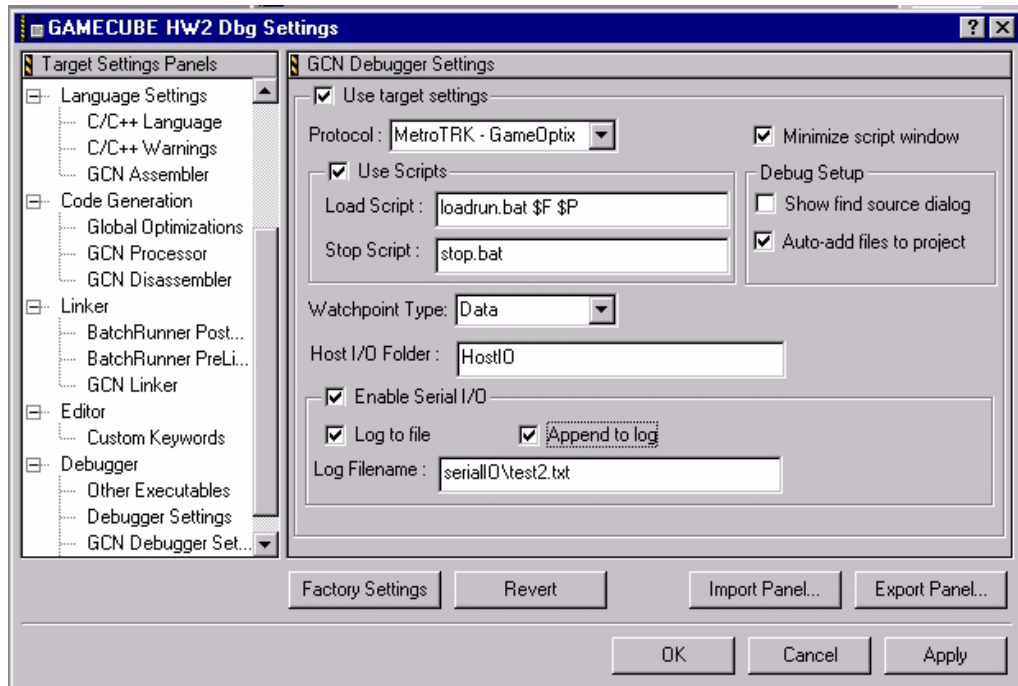
Figure 3.16 Preferences Modification Warning Window



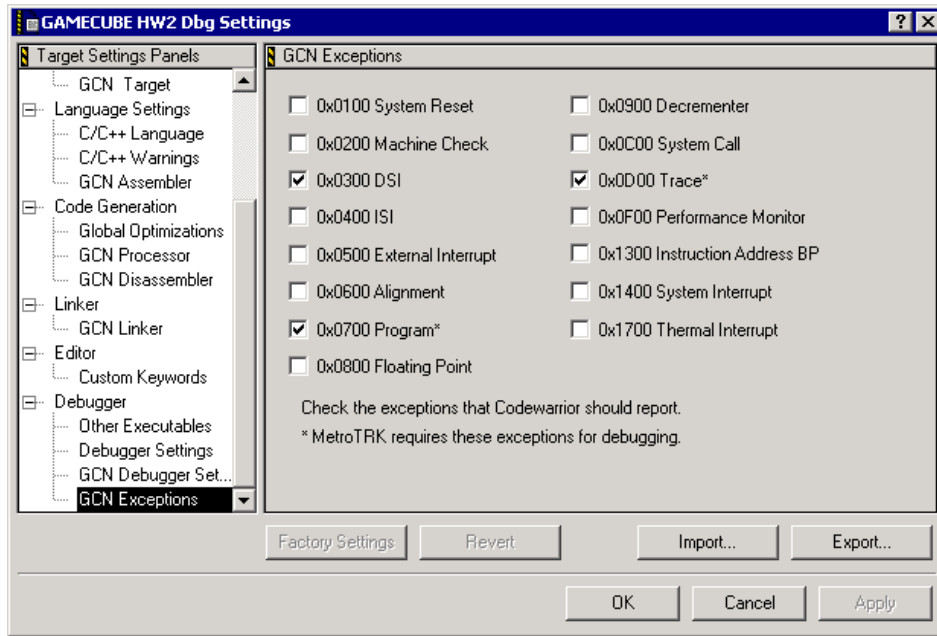
4. Verify that the default debugger settings are correct.

There are two debugger settings panels that must be verified to contain the settings most appropriate for the debugging session: the **GCN Debugger Settings** (Figure 3.17) and **GCN Exceptions** (Figure 3.18) panels.

Figure 3.17 GCN Debugger Settings Panel



**Figure 3.18 GCN Exceptions Panel**



5. Select **Project > Debug** from the CodeWarrior Menu Bar to debug the new project.