TWL-System

Build System

Description of the Build System and Source Tree

2008/05/30

TWL-06-0078-001-A
Released: February 27, 2009

Table of Contents

Tables

Figures

Revision History

Revision Date	Description
2008/05/30	• Made revisions in line with the NITRO-System name change (from NITRO-System to TWL-System). • Revised section 2.2 Build Tools. • Revised section 3.2.1 Library File Naming Conventions. • Revised all figures.
2008/04/08	• Separated out Quick Start and put it in `QuickStart.pdf`. • Added support for TWL-SDK.
2006/05/29	Deleted the description of the NITRO-SDK sound patch.
2004/10/12	• Added a description of `nnslibdefs` and `commondefs.cctype.CW` to “Files Required to Build.” • Changed the term “variable” to “macro switch” according to the notation in the SDK. • Changed the word “TEG” to “TS” in descriptions and figures.
2004/08/10	Corrected Figure 3-4 (Build directory structure).
2004/05/28	Changed “interface” to “include” on page 8.
2004/04/12	Corrected typo.
2004/04/08	• Standardized terminology for NITRO-SDK, IS-NITRO-CHARACTER, etc. • Revised trademark description.
2004/04/02	• Added an item in Chapter 4 related to installing the NITRO-SDK sound driver. • Supplemented the description of the build tool in Chapter 6 (about placing commondefs and modulerules in include statements).
2004/02/20	Deleted description related to build on the sub processor (ARM7).
2004/02/20	Added a depend directory to Figure 3-5 and Figure 3-6.
2004/02/13	Major revision of the source tree description because the storage location of the sub processor source files and header files changed.

1 Introduction

This document describes the build system and source tree used with the TWL-System. The TWL-System library is built on the TWL-SDK build system. Please read the TWL-SDK documentation along with this document.

2 Build System

2.1 Environment Variable

To build an application that uses the TWL-System library, you must set the absolute path to the TWL-System root directory (`TwlSystem`) in the environment variable `TWLSYSTEM_ROOT`. If the environment variable `TWLSYSTEM_ROOT` is not specified, the default value is `C:\TwlSystem`. In this document, this directory is shown as `$TwlSystem`.

2.2 Build Tools

TWL-System contains files with a description of often-used procedures, making it easy to write a makefile for applications that use the TWL-System library. The filenames and directory in which they are found are shown in the following list.

• Directory:	`$TwlSystem/build/buildtools/`
• Definition file for macro switch, etc.:	`commondefs`
• Definition file for compile procedures:	`modulerules`

When you make an application that uses the TWL-System library, use these two files by placing them in include statements in the makefile. For information on using these files, refer to the makefiles that are used for compiling the sample programs in the TWL-System library.

The TWL-System build system is built on the TWL-SDK build system. In the setting files for these build systems, `commondefs` and `modulerules` in TWL-SDK are placed in include statements, and TWL-System library-specific settings are added to the TWL-SDK settings. Therefore, if the TWL-System versions of the `commondefs` and `modulerules` files are placed in include statements, there is no need to place the TWL-SDK `commondefs` or `modulerules` files in include statements.

2.2.1 Describing the Makefile

You code and build a makefile for TWL-System in almost the same way as when you develop using only the TWL-SDK. The only difference to the TWL-SDK makefile is the section in which the `commondefs` and `modulerules` files are placed in include statements. Only the name of the environment variable is different.

- Use the following `include` statements in a TWL-SDK makefile

```
include $(TWLSDK_ROOT)/build/buildtools/commondefs
include $(TWLSDK_ROOT)/build/buildtools/modulerules
```

- Use the following `include` statements in a TWL-System makefile

```
include $(TWLSYSTEM_ROOT)/build/buildtools/commondefs
include $(TWLSYSTEM_ROOT)/build/buildtools/modulerules
```

2.2.2 Setting the Platform

To build an application that uses the TWL-System library, you must set the platform used as the build target. The platform used as the build target is set in the environment variable `TWLSDK_PLATFORM`.

Values that can be set for the environment variable `TWLSDK_PLATFORM` are shown in Table 2-1.

Table 2-1 Values That Can Be Set For TWLSDK_PLATFORM

Environment Variable Value	Generated Code
TWL	Generates code for TWL.
NITRO	Generates code for NITRO.
ALL	Generates code for both TWL and NITRO.
TWL NITRO	

The environment variable `TWLSDK_PLATFORM` must be set because it does not have a default value.

2.2.3 Selecting Code Generation for TWL

To generate code for TWL based on the platform setting, you can select to generate either TWL-dedicated code or TWL/NITRO hybrid code. To select the code to be generated, set the TWL-SDK build switch `TWL_ARCHGEN`.

Values that can be set for the build switch `TWL_ARCHGEN` are shown in Table 2-2.

Table 2-2 Values That Can Be Set in TWL_ARCHGEN

TWL_ARCHGEN Values	Generated Code
LIMITED	Generates TWL-dedicated code.
HYBRID	Generates TWL/NITRO hybrid code.
ALL	Generates both TWL-dedicated code and hybrid code.
LIMITED HYBRID	

If the build switch `TWL_ARCHGEN` is not explicitly set, TWL/NITRO hybrid code is generated.

2.2.4 Setting the Compile Target

With TWL-System you can select the same three build targets (DEBUG, RELEASE, and FINALROM) as with the TWL-SDK. To specify a compile target, set the appropriate value (such as `TRUE`) for any one of the build compile options shown in Table 2-3. If no compile target is explicitly specified, RELEASE version code is generated.

Table 2-3 Build Switches You Can Use

Command	Process
`% make TWL_DEBUG=TRUE`	Builds the final target of the debug version.
`% make TWL_RELEASE=TRUE`	Builds the final target of the release version.
`% make TWL_FINALROM=TRUE`	Builds the final target of the final ROM version.

2.2.5 Other Build Switches

Build switches of the TWL-SDK can be used because the TWL-System library is built on the TWL-SDK build system. Many build switches in addition to those described so far have been prepared for the TWL-SDK build system. For details on TWL-SDK build switches, see the TWL-SDK document `$TwlSDK/docs/SDKRules/Rule-Defines.html`.

2.2.6 Target

With the TWL-System library, you can use some of the targets provided by the TWL-SDK. Table 2-4 shows the targets that you can use.

Table 2-4 Usable Targets

Command	Process
`% make build`	Starts compiling and creates final target.
`% make install`	Installs (copies) the files created by make build in another directory.
`% make run`	If IS-NITRO-EMULATOR can be used in this environment, begins to run the target files generated by make build.
`% make full`	Generates files for all versions of each compile target.
`% make clean`	Deletes files generated by make build.
`% make clobber`	Completely deletes files generated by make build.

3 Source Tree

Figure 3-1 First Directory Level of the Source Tree

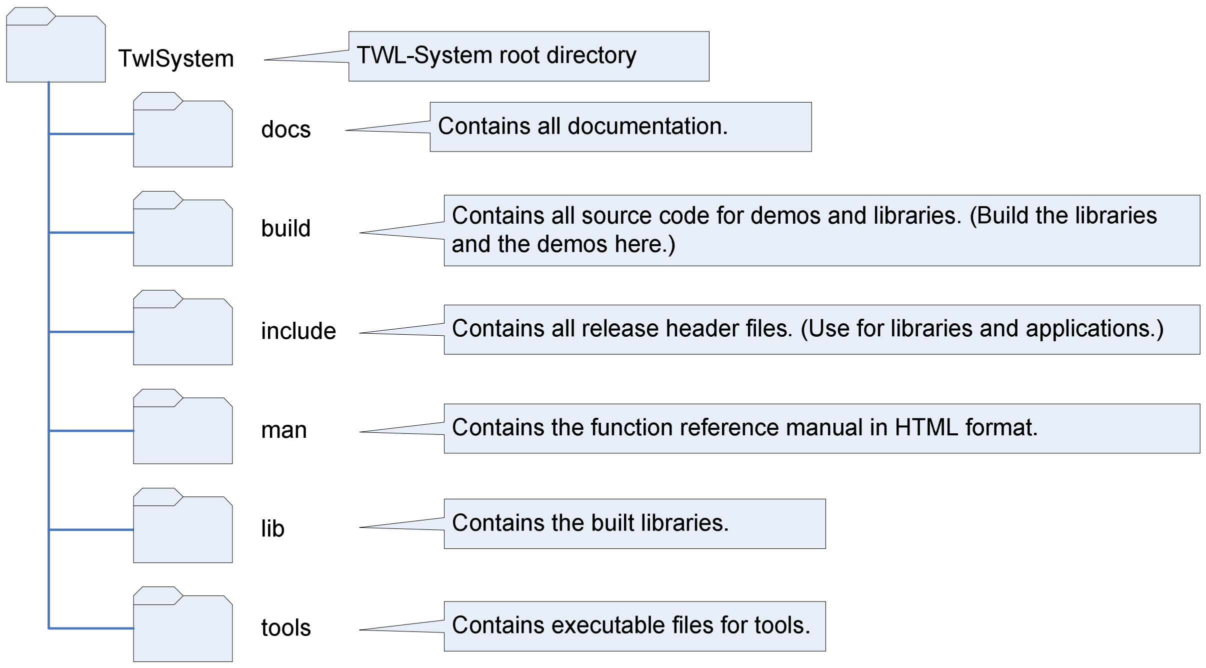

TwlSystem
TWL-System root directory
docs
Contains all documentation.
build
Contains all source code for demos and libraries. (Build the libraries and the demos here.)
include
Contains all release header files. (Use for libraries and applications.)
man
Contains the function reference manual in HTML format.
lib
Contains the built libraries.
tools
Contains executable files for tools.

Figure 3-1 shows the directories in the first level of the TWL-System source tree. There are six directories in the `TwlSystem` directory. Of these six directories, the following sections describe the structure of those directories related to building applications.

3.1 Include

Figure 3-2 Include Directory Structure

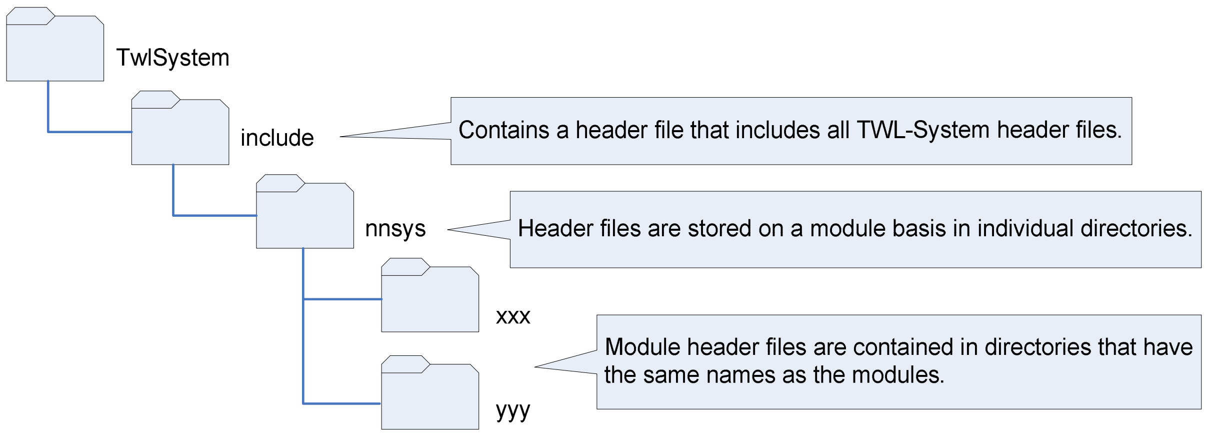

TwlSystem
include
Contains a header file that includes all TWL-System header files.
nnsys
Header files are stored on a module basis in individual directories.
xxx
yyy
Module header files are contained in directories that have the same names as the modules.

Figure 3-2 shows the structure of the include directory. All system include paths in TWL-System library are relative paths from `$TwlSystem/include`.

The `$TwlSystem/include` directory contains a header file, `nnsys.h`, to place all header files in the TWL-System library in an include statement. To place this file in an include statement, use the following specification.

`#include <nnsys.h>`	Places all header files in an include statement.

Headers for each module are stored in the `nnsys` directory that is in `$TwlSystem/include`. They are stored in directories that are unique to each module. To place a header file (Foundation library, NITRO-Composer, and so forth) for a specific module in an include statement in the application, use the following specification.

`#include <nnsys/fnd.h>`	Places all Foundation library header files in an include statement.
`#include <nnsys/snd.h>`	Places all NITRO-Composer header files in an include statement.

3.2 Library

Figure 3-3 Library Directory Structure

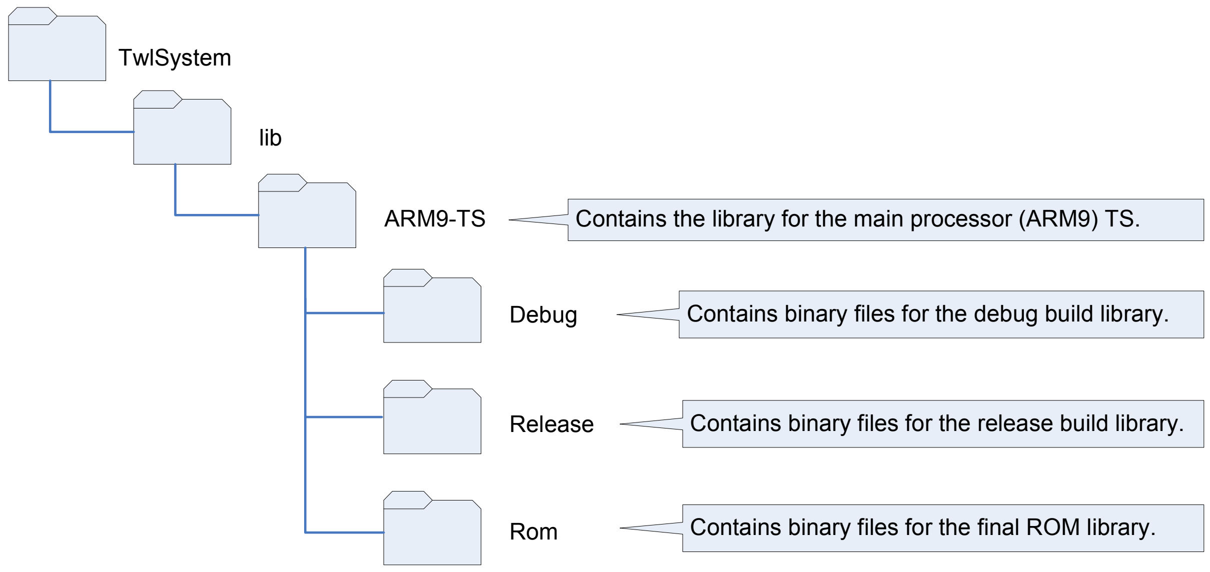

TwlSystem
lib
ARM9-TS
Contains the library for the main processor (ARM9) TS.
Debug
Contains binary files for the debug build library.
Release
Contains binary files for the release build library.
Rom
Contains binary files for the final ROM library.

Figure 3-3 shows the structure of the library directory. All of the TWL-System library binary files are included in the `$TwlSystem/lib` directory. In the build system of TWL-System library, switch the library used according to the specified build switch. All necessary libraries will be passed to the linker. Therefore, the developer does not need to consider which libraries should be linked.

3.2.1 Library File Naming Conventions

TWL-System library names begin with the prefix `lib` (indicating library), followed by `nns` (indicating that the file belongs to TWL-System), followed by a 2-3 alphabetic-character module name (library name). If a library can be used by TWL, the library platform is indicated by a word appended to the end of the module name. Libraries built in THUMB mode also have `thumb` appended.

Table 3-1 Library File Naming Conventions

Library Name	Library Type
`libnns + <module name>.a`	Library for NITRO (ARM mode)
`libnns + <module name>.thumb.a`	Library for NITRO (THUMB mode)
`libnns + <module name>.TWL.HYB.a`	Library for both TWL and NITRO (ARM mode)
`libnns + <module name>.TWL.HYB.thumb.a`	Library for both TWL and NITRO (THUMB mode)
`libnns + <module name>.TWL.LTD.a`	Library for TWL (ARM mode)
`libnns + <module name>.TWL.LTD.thumb.a`	Library for TWL (THUMB mode)

The following examples are actual library names:

`libnnsfnd.thumb.a`	Foundation library for NITRO (THUMB mode)
`libnnsg2d.TWL.HYB.a`	2D library for both TWL and NITRO (ARM mode)
`libnnsg3d.TWL.LTD.thumb.a`	3D library for TWL (THUMB mode)

3.3 Build Tree

Figure 3-4 Build Directory Structure

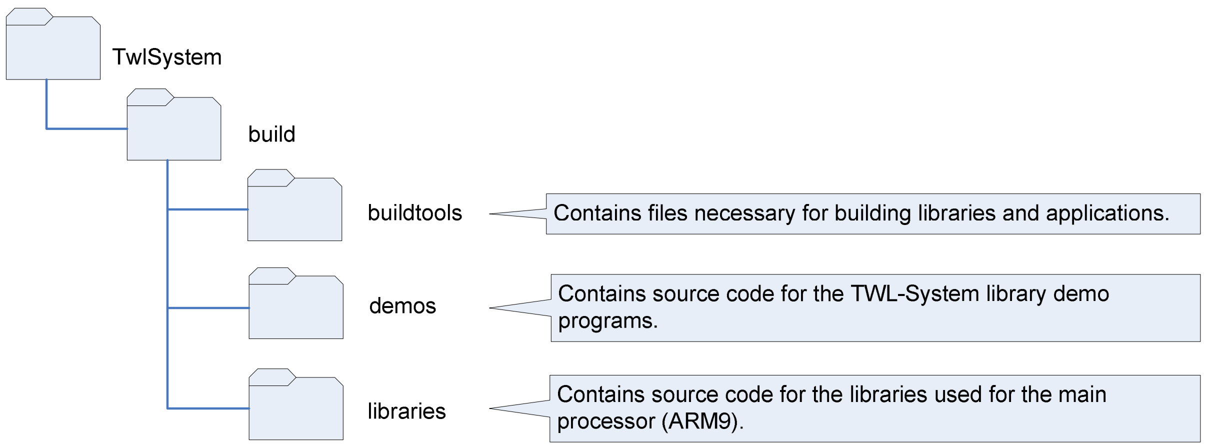

TwlSystem
build
buildtools
Contains files necessary for building libraries and applications.
demos
Contains source code for the TWL-System library demo programs.
libraries
Contains source code for the libraries used for the main processor (ARM9).

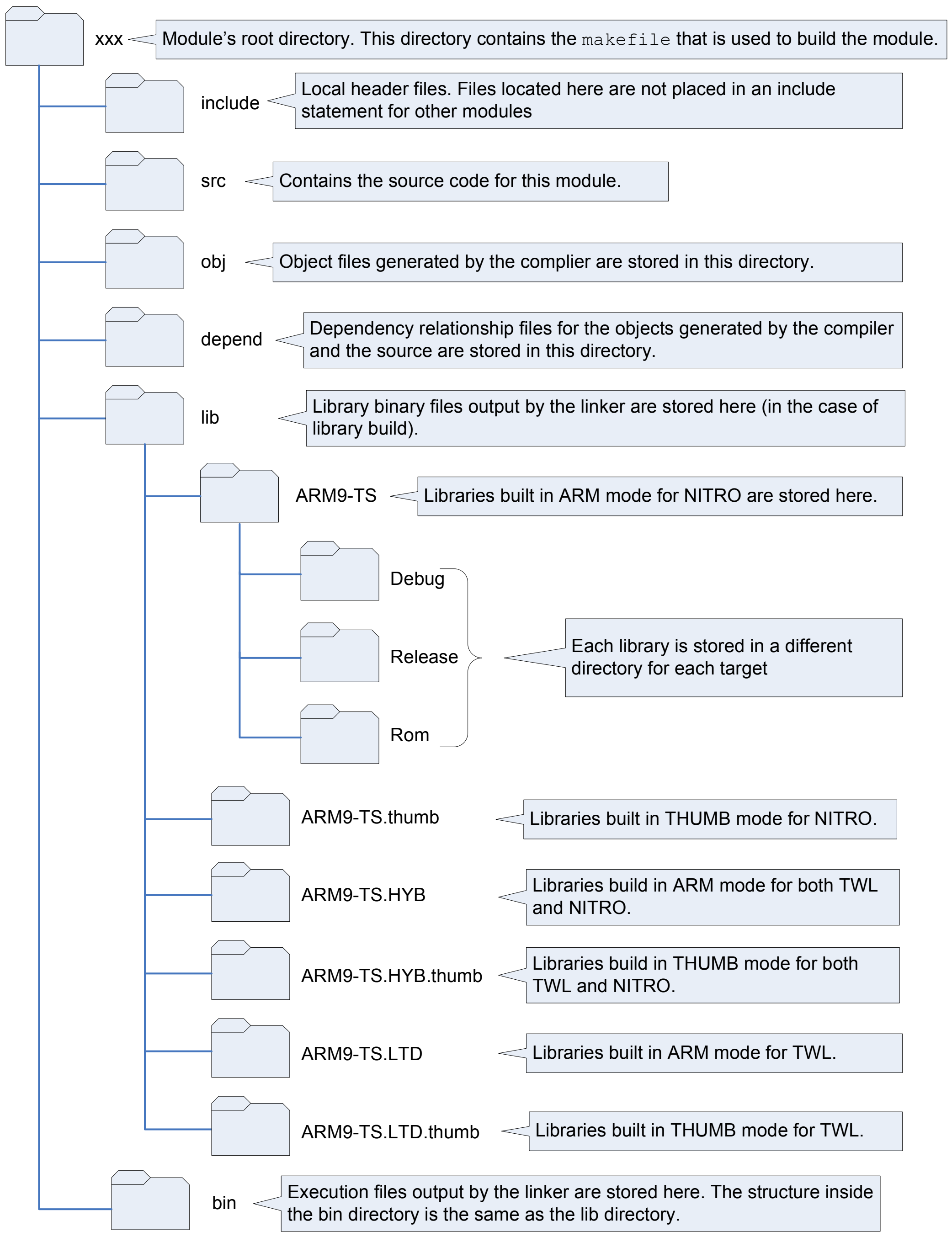
Figure 3-4 shows the structure of the `build` directory. Library demo program source code is stored under the `build` directory. The library and demo programs are built here.

Library source code is stored under the `libraries` directory. There is a different directory for each module. The `buildtools` directory contains the `commondefs` and `modulerules` files that are placed in include statements in the makefile that is used to build libraries and application software.

3.4 Library and Demo Sub-Directory Structure

Libraries, demo programs, and individual modules for test programs share the same basic directory structure shown in Figure 3-5.

Figure 3-5 Library and Demo Basic Directory Structure


xxx
Module's root directory. This directory contains the `makefile` that is used to build the module.
include
Local header files. Files located here are not placed in an include statement for other modules
src
Contains the source code for this module.
obj
Object files generated by the complier are stored in this directory.
depend
Dependency relationship files for the objects generated by the compiler and the source are stored in this directory.
lib
Library binary files output by the linker are stored here (in the case of library build).
ARM9-TS
Libraries built in ARM mode for NITRO are stored here.
Debug
Release
Rom
Each library is stored in a different directory for each target
ARM9-TS.thumb
Libraries built in THUMB mode for NITRO.
ARM9-TS.HYB
Libraries build in ARM mode for both TWL and NITRO.
ARM9-TS.HYB.thumb
Libraries build in THUMB mode for both TWL and NITRO.
ARM9-TS.LTD
Libraries built in ARM mode for TWL.
ARM9-TS.LTD.thumb
Libraries built in THUMB mode for TWL.
bin
Execution files output by the linker are stored here. The structure inside the bin directory is the same as the lib directory.

The makefile used to build a module is located in the root directory of that module. The makefile uses the `commondefs` and `modulerules` files in `$TwlSystem/build/buildtools` to generate a dependencies file and start the compiler and the linker.

Each directory that has a module name contains a local include directory. This directory is for exclusive header files that are not shared between modules.

3.5 Files Required for Build

The following files that are in the `$TwlSystem/build/buildtools/` directory are used to build the TWL-System library and applications that use the TWL-System library. These files are placed in an include statement in the makefile.

3.5.1 commondefs File

The `commondefs` file defines the macro switches needed to build the TWL-System library. The `commondefs` file of the TWL-SDK is placed in an include statement of the `commondefs` file of the TWL-System library. In addition to the settings made in the TWL-SDK `commondefs` file, this file sets macro switches that are related to the TWL-System library.

3.5.2 modulerules File

Currently the TWL-System library `modulerules` file does nothing more than place the TWL-SDK `modulerules` file in an include statement. In the future it is possible that some settings will be added. When you use the TWL-System library, therefore, use the TWL-System `modulerules` file instead of using the TWL-SDK `modulerules` file directly.

3.5.3 nnslibdefs File

The `nnslibdefs` file is included in an include statement in the TWL-SDK `commondefs` file. This file sets the include path and the library path in the NITRO-System library, as well as the library that is passed to the linker.

3.5.4 commondefs.cctype.CW File

The `commondefs.cctype.CW` file is included in an include statement in TWL-System's `commondefs` file. This file sets the macro switch being used in the TWL-System library.