

N I N T E N D O
NITRO-System

Build System

Source Tree Description

Version 1.2.0

**The contents in this document are highly
confidential and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo of America Inc. and/or Nintendo Company Ltd. and are protected by Federal copyright law. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	5
2	Quick Start.....	6
2.1	Preparing the Development Tools.....	6
2.2	Building the NITRO-System Library.....	6
2.2.1	Extracting the NITRO-System Package	6
2.2.2	Setting the Environment Variable	6
2.2.3	Building the NitroSystem Tree.....	7
2.2.4	Running the Demo Programs.....	7
2.2.4.1	Starting ensata	7
2.2.4.2	Loading Executable Files	7
2.2.4.3	Executing Files.....	7
2.2.4.4	Stopping Execution	7
2.3	Build Tools	7
2.3.1	Describing the Makefile	8
2.3.2	Build Switches.....	8
2.3.3	Target	9
3	Source Tree.....	10
3.1	Include.....	10
3.2	Library	11
3.2.1	Library File Naming Convention	11
3.3	Build Tree	11
3.4	Library and Demo Sub Directory Structure	12
3.5	Files Required for Build	13
3.5.1	commondefs File	13
3.5.2	modulerules File	13
3.5.3	nnslibdefs File	13
3.5.4	commondefs.cctype.CW File.....	13

Tables

Table 2-1	Build Switches You Can Use.....	8
Table 2-2	Targets You Can Use	9

Figures

Figure 3-1	First Directory Level of the Source Tree.....	10
Figure 3-2	Include Directory Structure.....	10
Figure 3-3	Library Directory Structure	11
Figure 3-4	Build Directory Structure	11
Figure 3-5	Library and Demo Basic Directory Structure	12

Revision History

Version	Revision Date	Description
1.2.0	5/29/2006	<ul style="list-style-type: none">Deleted the description of the NITRO-SDK sound patch.
1.1.0	10/12/2004	<ul style="list-style-type: none">Added a description of <code>nnslibdefs</code> and <code>commondefs.cctype.CW</code> to “Files Required to Build” on page 10.Changed the term “variable” to “macro switch” according to the notation in the SDK.Changed the word “TEG” to “TS” in descriptions and figures.
1.0.8	8/10/2004	Corrected Figure 3-4 (Build directory structure)
1.0.7	6/22/2004	Changed “interface” to “include” on page 8.
1.0.6	4/12/2004	Corrected typo.
1.0.5	4/8/2004	<ul style="list-style-type: none">Standardized terminology for NITRO-SDK, IS-NITRO-CHARACTER, etc.Revised trademark description.
1.0.4	4/2/2004	<ul style="list-style-type: none">P. 4 Added an item related to installing the NITRO-SDK sound driver.P. 6 Supplemented the description of the build tool (about placing <code>commondefs</code> and <code>modulerules</code> in include statements).
1.0.3	2/20/2004	Deleted description related to build on the sub processor (ARM7).
1.0.2	2/20/2004	Added a <code>depend</code> directory to Fig. 3-5, Fig. 3-6.
1.0.1	2/13/2004	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 procedure for building the NITRO-System library and demo programs, and describes the NITRO-System source tree. The NITRO-System library is built on the NITRO-SDK. Please read the NITRO-SDK documentation along with this document.

2 Quick Start

This section describes the creation of waveform data.

2.1 Preparing the Development Tools

The NITRO-System is built on the NITRO-SDK. Therefore, the environment for building the NITRO-System library is matched to environment for building the NITRO-SDK. When you use the NITRO-System library, you must have an environment that allows the build of the NITRO-SDK.

The NITRO-System can be built in the Microsoft Windows 2000 Professional environment.

1. You need the following tools and SDK to build (compile, etc.) the NITRO-System:
 - CodeWarrior for NITRO
 - Cygwin or MinGW (MSYS tool)
 - NITRO-SDK
2. You need one of the following tools to debug:
 - NITRO emulator ensata
 - IS-NITRO-EMULATOR

2.2 Building the NITRO-System Library

This section contains a description of the procedure for building the NITRO-System library and demo programs.

2.2.1 Extracting the NITRO-System Package

Extract the NITRO-System package anywhere on the local disk. The NITRO-System package is compressed in ZIP format. Use an appropriate tool to decompress it. When the package is decompressed, a directory called `NitroSystem` will be created.

2.2.2 Setting the Environment Variable

Set the absolute path to the extracted directory, `NitroSystem`, in the environment variable `NITROSYSTEM_ROOT`. If the environment variable `NITROSYSTEM_ROOT` is not specified, the path will be treated as if `C:\NitroSystem` had been set. Hereafter we will refer to this directory as `$NitroSystem`.

2.2.3 Building the NitroSystem Tree

After you have set the environment variable, run `make` in the NITRO-System library root directory (`NitroSystem` directory) to build the library and the samples.

2.2.4 Running the Demo Programs

To confirm that the build was performed properly, try running the sample programs. This section describes the procedure using the NITRO emulator `ensata` as an example.

2.2.4.1 Starting `ensata`

Double-click the `ensata` icon to start `ensata`. There are two `ensata` programs: `ensata_dx.exe`, which uses DirectX to perform the drawing process, and `ensata.exe`, which does not use DirectX to process drawing. Select according to the environment on your PC. Both require `DirectInput8` (`DINPUT8.DLL`) to support controller input. If DirectX is not installed on the PC you are using, get the DLL from the Microsoft Web site.

2.2.4.2 Loading Executable Files

Right-click in the `ensata` window, and select “Open NITRO File”. Use the dialog box to specify the `nef` file that you want to run.

2.2.4.3 Executing Files

Click the “Run” button (the second button to the right of the Frame button) button in the `ensata` window.

2.2.4.4 Stopping Execution

To stop execution click the “Stop” button (the button that shows a hand) in the `ensata` window.

2.3 Build Tools

The NITRO-System contains files with a description of often-used procedures, making it easy to write a makefile for applications that use the NITRO-System library. The file names and the directory in which they are found are listed below.

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

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

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

2.3.1 Describing the Makefile

You code and build a makefile for the NITRO-System in almost the same way as when you develop using only the NITRO-SDK. The only difference to the NITRO-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).

- Placing the file in an include statement in the NITRO-SDK makefile

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

- Placing the file in an include statement in the NITRO-System makefile

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

2.3.2 Build Switches

With the NITRO-System you can use the three build switches that are provided in the NITRO-SDK. The TS board release version library is linked by default. However, you can use macro settings during building to build a debug version or a final ROM version. The following table shows the NITRO-System build switches that you can use.

Table 2-1 Build Switches You Can Use

Command	Process
% make NITRO_DEBUG=TRUE	Builds the final target of the debug version.
% make NITRO_RELEASE=TRUE	Builds the final target of the release version.
% make NITRO_FINALROM=TRUE	Builds the final target of the final ROM version.

For detailed information on the build switches see this NITRO-SDK documentation.

`$NitroSDK/docs/SDKRules/Rule-Defines.html`

2.3.3 Target

With the NITRO-System library, you can use some of the targets provided by the NITRO-SDK. The following table shows the targets that you can use.

Table 2-2 Targets You Can Use

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

For detailed information on targets see the following NITRO-SDK documentation:

`$NitroSDK/docs/SDKHowTo/HowToBuildSDKTree.rtf`

3 Source Tree

Figure 3-1 First Directory Level of the Source Tree

Error! No topic specified.

Figure 3-1 shows the directories in the first level of the NITRO-System source tree. The following sections describe the main directory structures in the source tree.

3.1 Include

Figure 3-2 Include Directory Structure

Error! No topic specified.

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

The `$NitroSystem/include` directory contains a header file, `nnsys.h`, to place all header files in NITRO-System library in an include statement. To place this file in an include statement, specify as shown below.

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

Headers for each module are stored in the `nnsys` directory that is in `$NitroSystem/include`. They are stored in directories that are unique to each module. To place a header file (Foundation library, NITRO-Composer, etc.) 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

Error! No topic specified.

Figure 3-3 shows the structure of the library directory. It contains all of the NITRO-System library binary files. In the build system of NITRO-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 Convention

In principle a NITRO-System library names have the prefix `lib`, which indicates library, followed by `nns`, which indicates that the file belongs to the NITRO-System, followed by a 2-3 alphabetic-character module name (library name).

Libraries built in THUMB mode have `.thumb` appended to the module name.

<code>lib + nns + <module name>.a</code>	Library for the main processor (ARM mode)
<code>lib + nns + <module name>.thumb.a</code>	Library for the main processor (THUMB mode)

Here are some examples of actual library names:

<code>libnnsfnd.a</code>	// Foundation library (main processor, ARM mode)
<code>libnnsfnd.thumb.a</code>	// Foundation library (main processor, THUMB mode)

3.3 Build Tree

Figure 3-4 Build Directory Structure

Error! No topic specified.

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.

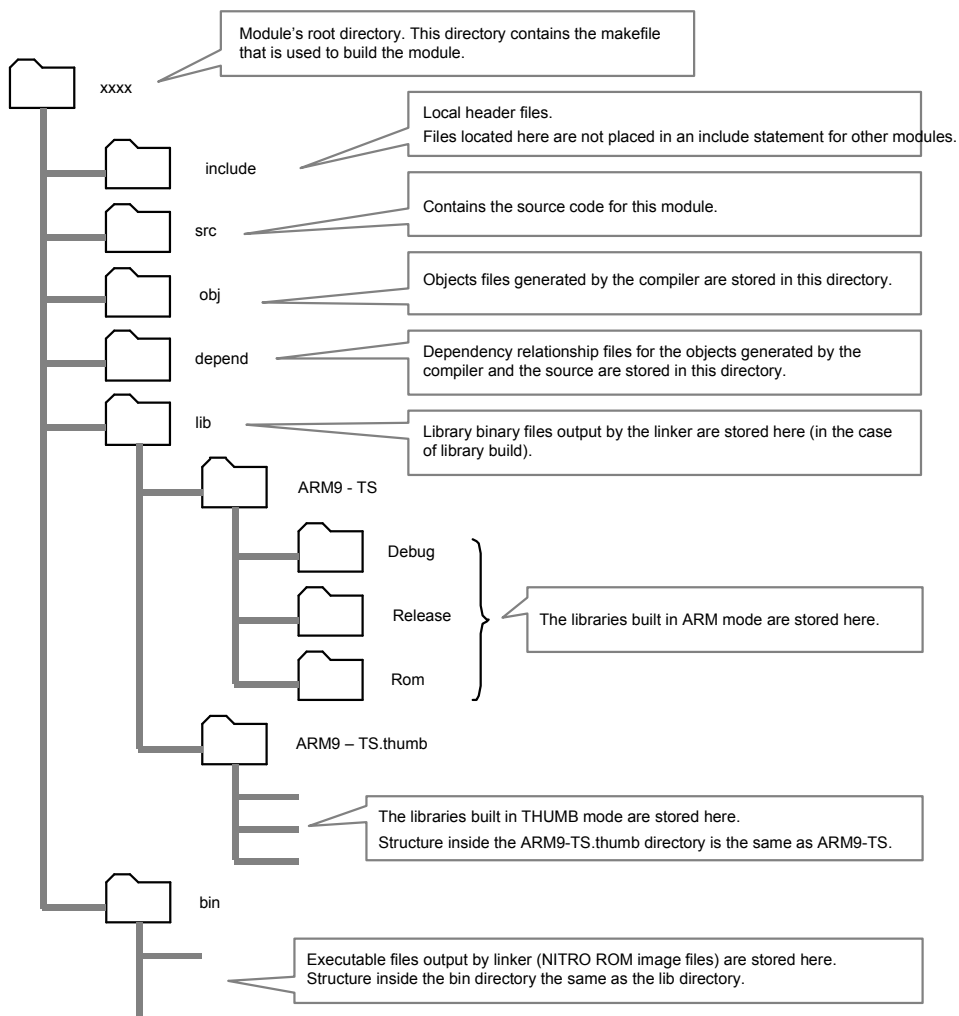
Library source code is stored under the libraries directory—there is a different directory for each module. The `buildtools` directory contains the `commondefs` and `modulerrules` files that are

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 have basically the directory structure shown in Figure 3-5.

Figure 3-5 Library and Demo Basic Directory Structure



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 `$NitroSystem/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 `$NitroSystem/build/buildtools/` directory are used to build the NITRO-System library and applications that use NITRO-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 that are needed to build the NITRO-System library. The `commondefs` file of NITRO-SDK is placed in an include statement of the `commondefs` file of NITRO-System library. In addition to the settings that are performed in the NITRO-SDK `commondefs` file, this file sets macro switches that are related to the NITRO-System library.

3.5.2 modulerules File

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

3.5.3 nnslibdefs File

The `nnslibdefs` file is placed in an include statement from in the NITRO-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.

In the NITRO-System library version 10/12/2004 and later, the NITRO-System library setting is done without using `LINCLUDES`, `LLIBRARY_DIRS`, and `LLIBRARIES`, which are macro switches in the NITRO-SDK. These macro switches are completely released so that the user can set the special library.

3.5.4 commondefs.cctype.CW File

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

Windows is a registered trademark or trademark of Microsoft Corporation (USA) in the U.S. and other countries.

CodeWarrior is a registered trademark or trademark of Metrowerks Inc. in the U.S. and other countries.

Other company names, product names, etc., in this document are registered trademarks or trademarks of each company.

© 2004-2006 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed or loaned in whole or in part without the prior approval of Nintendo.