

ビルドシステム

ビルドシステムとソースツリーの説明

2008-04-08

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、厳重な取り扱い、管理を行ってください。

目次

1	はじめに	4
2	ビルドシステム	4
2.1	環境変数	4
2.2	ビルドツール	4
2.2.1	Makefileの記述	4
2.2.2	ビルドスイッチ	5
2.2.3	ターゲット	5
3	ソースツリー	6
3.1	インクルード	6
3.2	ライブラリ	7
3.2.1	ライブラリファイルの命名規則	7
3.3	ビルドツリー	8
3.4	ライブラリとデモのサブディレクトリ構造	9
3.5	ビルドに必要なファイル	10
3.5.1	commondefsファイル	10
3.5.2	modulerulesファイル	10
3.5.3	nnslibdefsファイル	10
3.5.4	commondefs.cctype.CWファイル	10

表

表 2-1	利用可能なビルドスイッチ	5
表 2-2	利用可能なターゲット	5

図

図 3-1	ソースツリー第1階層のディレクトリ	6
図 3-2	インクルードディレクトリ構造	6
図 3-3	ライブラリディレクトリ構造	7
図 3-4	ビルドディレクトリ構造	8
図 3-5	ライブラリとデモの基本ディレクトリ構造	9

改訂履歷

改訂日	改訂内容
2008-04-08	<ul style="list-style-type: none"> ・クイックスタートを QuickStart.pdf に分離。 ・TWL-SDK に対応。
2006-05-29	<ul style="list-style-type: none"> ・NITRO-SDK のサウンドパッチに関する記述を削除。
2004-10-12	<ul style="list-style-type: none"> ・P.10 ビルドに必要なファイルに、nnslibdefs と commondefs.cctype.CW の説明を追加。 ・SDK の表記に合わせて、変数と言う語をマクロスイッチに変更。 ・文章や図中の TEG という表記を TS に変更。
2004-08-10	図 3-4 ビルドディレクトリ構造を修正。
2004-05-28	P.8 「インターフェース」と言う表記を「インクルード」に変更。
2004-04-12	<ul style="list-style-type: none"> ・誤字の修正。
2004-04-08	<ul style="list-style-type: none"> ・NITRO-SDK, IS-NITRO-CHARACTER 等の用語を統一。 ・商標表記を修正。
2004-04-02	P.4 NITRO-SDK サウンドドライバのインストールに関する項目を追加。 P.6 ビルドツール (commondefs と modulerules のインクルードについて) の説明を補足。
2004-02-20	サブプロセッサ (ARM7) 側のビルドに関する記述を削除。
2004-02-20	図 3-5、図 3-6 に depend ディレクトリを追加。
2004-02-13	サブプロセッサのソースとヘッダファイルの格納場所が変更された為、ソースツリーの説明を大幅に修正。

1 はじめに

このドキュメントでは、NITRO-System のビルドシステムとソースツリーについて説明しています。なお、NITRO-System ライブラリのビルドシステムは、TWL-SDK のビルドシステム上に構築されておりますので、TWL-SDK のドキュメントも合わせてご覧ください。

2 ビルドシステム

2.1 環境変数

NITRO-System ライブラリを使ったプログラムをビルドする場合には、環境変数 `NITROSYSTEM_ROOT` に、NITRO-System のルートディレクトリ (`NitroSystem`) の絶対パスが設定されている必要があります。環境変数 `NITROSYSTEM_ROOT` が指定されていない場合には、`C:\¥NitroSystem` が設定されたものとして扱われます。以後このディレクトリのことを `¥NitroSystem` と表記します。

2.2 ビルドツール

NITRO-System では、NITRO-System ライブラリを利用するアプリケーション用の `Makefile` を容易に記述できるようにするために、よく使われる手順についての記述をまとめたファイルを以下のディレクトリに用意しています。

- ディレクトリ: `¥NitroSystem/build/buildtools/`
- マクロスイッチなどの定義ファイル: `commondefs`
- コンパイル手順定義ファイル: `modulerrules`

NITRO-System ライブラリを利用したアプリケーションを作成される場合には、この2つのファイルを `Makefile` 中にインクルードして使います。これらのファイルの使用方法は、NITRO-System ライブラリのサンプルプログラム等のコンパイルに使用している `Makefile` をご参考ください。

なお、NITRO-System のビルドシステムは、TWL-SDK のビルドシステムの上に構築されています。これらの設定ファイルの中では、TWL-SDK の `commondefs` ファイルと `modulerrules` ファイルをそれぞれインクルードしており、TWL-SDK の設定に NITRO-System ライブラリ固有の設定を付け加えると言う形となっています。よって、NITRO-System 版の `commondefs` ファイルと `modulerrules` ファイルをインクルードする場合には、TWL-SDK の `commondefs` ファイルと `modulerrules` ファイルをインクルードする必要はありません。

2.2.1 Makefileの記述

NITRO-System を用いた `Makefile` の記述やビルドの仕方は、TWL-SDK のみを利用した開発の場合とほとんど同じとなっています。TWL-SDK の `Makefile` との唯一の違いは、`commondefs` ファイルと `modulerrules` ファイルをインクルードしている部分のみとなります(環境変数の名前のみが異なります)。

- TWL-SDK の Makefile でのインクルード
include \$(NITROSDK_ROOT)/build/buildtools/commondefs
include \$(NITROSDK_ROOT)/build/buildtools/modulerrules
- NITRO-System の Makefile でのインクルード
include \$(NITROSYSTEM_ROOT)/build/buildtools/commondefs
include \$(NITROSYSTEM_ROOT)/build/buildtools/modulerrules

2.2.2 ビルドスイッチ

NITRO-System では、TWL-SDK で用意されている3つのビルドスイッチを利用することが可能です。デフォルトでは、TS 用のリリース版ライブラリがリンクされますが、ビルド時のマクロの設定によってデバッグ版やファイナル ROM 版のビルドを行なうことができます。NITRO-System で利用可能なビルドスイッチを下表にまとめます。

表 2-1 利用可能なビルドスイッチ

コマンド	処理
% make NITRO_DEBUG=TRUE	デバッグバージョンの最終ターゲットをビルドします。
% make NITRO_RELEASE=TRUE	リリースバージョンの最終ターゲットをビルドします。
% make NITRO_FINALROM=TRUE	ファイナルROMバージョンの最終ターゲットをビルドします。

ビルドスイッチについての詳しい情報は、下記の TWL-SDK のドキュメントをご覧ください。

[\\$NitroSDK/docs/SDKRules/Rule-Defines.html](#)

2.2.3 ターゲット

NITRO-System ライブラリでは、TWL-SDK で用意されている幾つかのターゲットを利用する事ができます。下表に利用可能なターゲットを示します。

表 2-2 利用可能なターゲット

コマンド	処理
% make build	コンパイルを開始し、最終ターゲットを作成します。
% make install	make build によって作成されたファイルを他のディレクトリへインストール(コピー)します。
% make run	IS-NITRO-EMULATOR が使用可能な環境の場合、make build で生成したターゲットファイルの実行を始めます。
% make full	make build 各コンパイルターゲット毎全てのバージョンのファイルを生成します。
% make clean	make build によって生成されたファイルを削除します。
% make clobber	make build によって生成されたファイルを完全に削除します。

3 ソースツリー

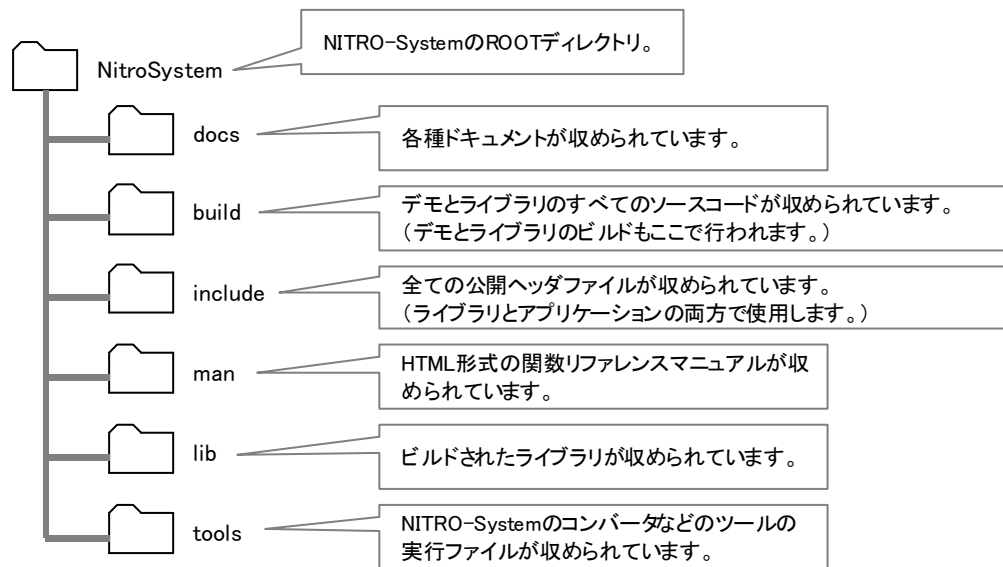


図 3-1 ソースツリー第1階層のディレクトリ

図 3-1 にNITRO-Systemのソースツリー第1階層にあるディレクトリを示します。次にソースツリーの中で主なディレクトリの構造について説明します。

3.1 インクルード

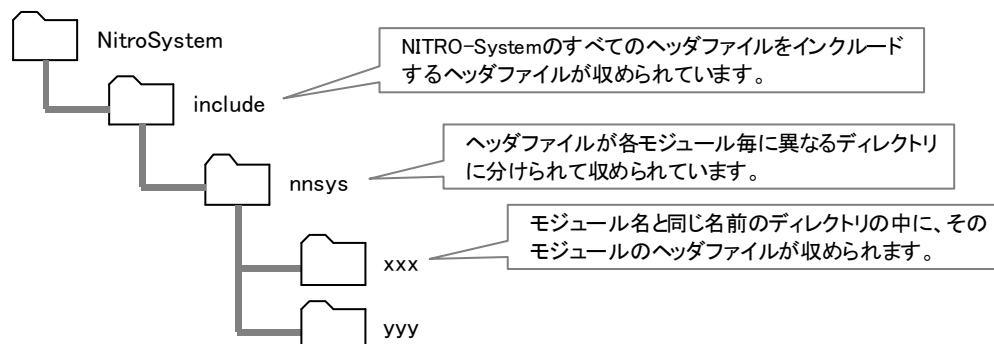


図 3-2 インクルードディレクトリ構造

図 3-2 に、インクルードディレクトリの構造を示します。NITRO-Systemライブラリのすべてのシステムインクルードパスは、\$NitroSystem/includeからの相対パスとなっています。

\$NitroSystem/include には、NITRO-System ライブラリのすべてのヘッダファイルをインクルードする為のヘッダファイルである nnsys.h が収められています。このファイルをインクルードする場合には、以下の様に指定します。

```
#include <nnsys.h>           // ヘッダファイルをすべてインクルード。
```

各モジュールのヘッダは、\$NitroSystem/include の中にある nnsys ディレクトリ内に、モジュール毎に異なるディレクトリに分けられて格納されています。アプリケーションでモジュールを特定したヘッダファイル(Foundation ライブラリ、

NITRO-Composer など) をインクルードしたい場合は、以下の様に指定します。

```
#include <nnsys/fnd.h>      // Foundationライブラリのヘッダファイルをすべてインクルード。
#include <nnsys/snd.h>      // NITRO-Composerのヘッダファイルをすべてインクルード。
```

3.2 ライブラリ

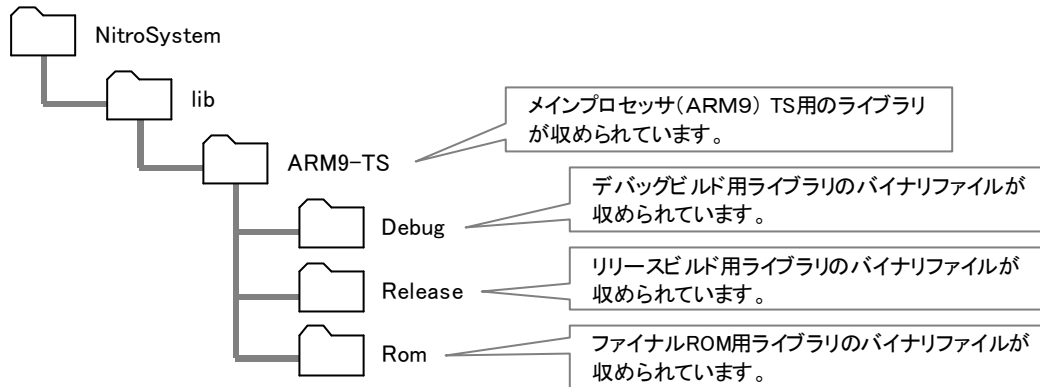


図 3-3 ライブラリディレクトリ構造

図 3-3 に、ライブラリディレクトリの構造を示します。NITRO-Systemライブラリのすべてのバイナリファイルがここに収められています。NITRO-Systemライブラリのビルドシステムでは、指定されたビルドスイッチに従って使用するライブラリを切り替えます。必要なライブラリはすべてリンカに渡されますので、開発者はどのライブラリをリンクすべきかを意識する必要はありません。

3.2.1 ライブラリファイルの命名規則

NITRO-System ライブラリの名前は、ライブラリを示す接頭辞 “lib” の後ろに、NITRO-System に属する事を示す語 “nns” が置かれ、その後にアルファベット2～3文字のモジュール名(ライブラリ名)が続いたものが基本となります。

THUMB モードでビルドされたライブラリには、モジュール名の後に “.thumb” という語が付きます。

lib + nns + モジュール名.a	メインプロセッサ用ライブラリ (ARMモード)。
lib + nns + モジュール名.thumb.a	メインプロセッサ用ライブラリ (THUMBモード)。

以下に、実際のライブラリ名の例を示します。

libnnsfnd.a	// Foundationライブラリ (メインプロセッサ ARMモード)。
libnnsfnd.thumb.a	// Foundationライブラリ (メインプロセッサ THUMBモード)。

3.3 ビルドツリー

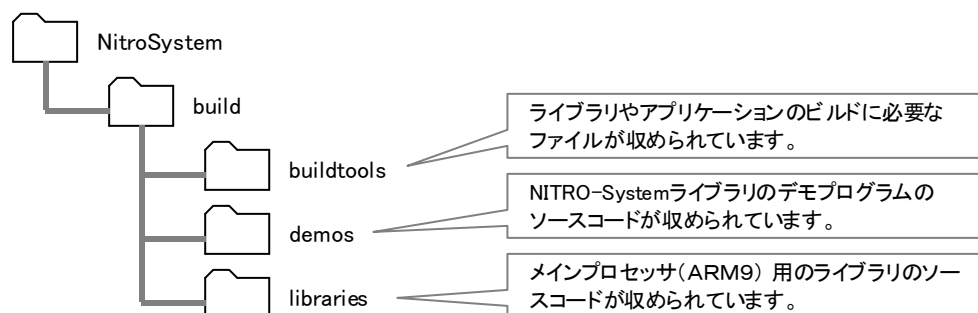


図 3-4 ビルドディレクトリ構造

図 3-4 に、ビルドディレクトリの構造を示します。buildディレクトリ以下には、ライブラリデモプログラムのソースコードが格納されており、ここで、ライブラリやデモプログラムがビルドされます。

ライブラリのソースコードは、libraries ディレクトリ内にモジュール毎に異なるディレクトリに分けられて格納されています。buildtools ディレクトリには、ライブラリやアプリケーションソフトのビルドに使用される Makefile にインクルードされる、commondefs と modulerules ファイルが収められています。

3.4 ライブラリとデモのサブディレクトリ構造

ライブラリとデモプログラム、及びテストプログラムの各モジュールは、基本的には 図 3-5 に示すようなディレクトリ構造となっています。

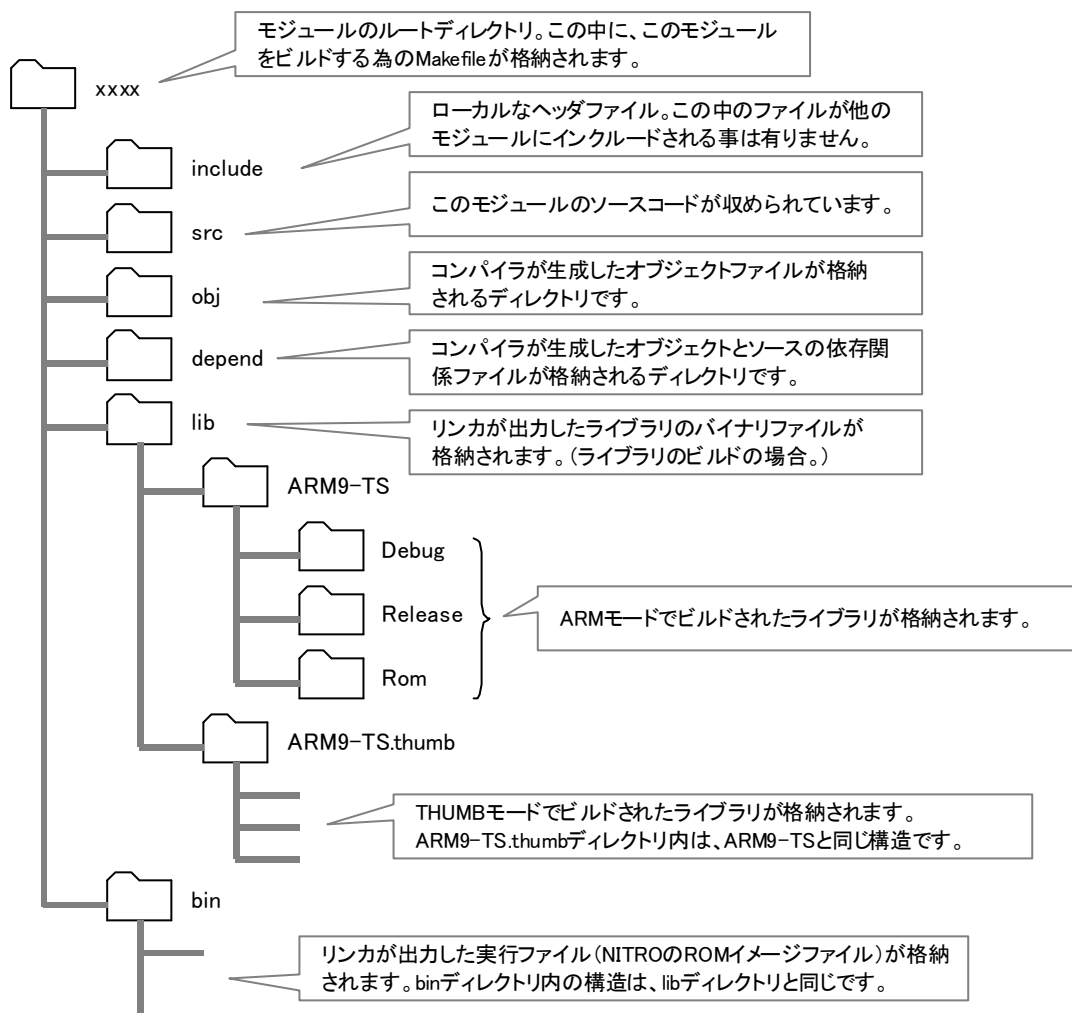


図 3-5 ライブラリとデモの基本ディレクトリ構造

各モジュールのルートディレクトリには、そのモジュールをビルドする為の Makefile が置かれます。この Makefile は、依存関係の生成とコンパイラとリンカの起動を行うために、\$NitroSystem/build/buildtools の中に格納されている commondefs ファイルと modulerules ファイルを使用します。

各モジュール名を持ったディレクトリ内にあるローカルな include ディレクトリは、モジュール間で共有しない専用のヘッダファイルのために存在します。

3.5 ビルドに必要なファイル

NITRO-System のライブラリや、NITRO-System ライブラリを使ったアプリケーションをビルドするためには、`$NitroSystem/build/buildtools/`ディレクトリに格納されている下記のファイルを用います。これらのファイルは、`Makefile` の中からインクルードされます。

3.5.1 commondefsファイル

`commondefs` ファイルでは、NITRO-System ライブラリのビルドに必要なマクロスイッチが定義されています。NITRO-System ライブラリの `commondefs` ファイルは、内部で `TWL-SDK` の `commondefs` ファイルをインクルードしています。このファイルでは、`TWL-SDK` の `commondefs` ファイルで行われている設定に加え、NITRO-System ライブラリに関するマクロスイッチの設定が行われています。

3.5.2 modulerulesファイル

現在、NITRO-System ライブラリの `modulerules` ファイルは、内部で `TWL-SDK` の `modulerules` ファイルをインクルードしているのみとなっています。しかし将来、なんらかの設定が付け加えられる可能性がありますので、NITRO-System ライブラリを利用される場合は、`TWL-SDK` の `modulerules` ファイルを直接使用せずに、NITRO-System の `modulerules` ファイルをお使い下さい。

3.5.3 nnslibdefsファイル

`nnslibdefs` ファイルは、`TWL-SDK` の `commondefs` ファイルからインクルードされます。このファイルでは、NITRO-System ライブラリのインクルードパスとライブラリのパスの設定、及びリンカに渡されるライブラリの設定を行っています。

NITRO-System ライブラリ 2004/10/12 版から NITRO-System ライブラリの設定を、`TWL-SDK` のマクロスイッチである `LINCLUDES`, `LLIBRARY_DIRS`, `LLIBRARIES` を使用せずに行うように変更されました。これらのマクロスイッチは、ユーザが独自のライブラリを設定することができるように、完全に解放されました。

3.5.4 commondefs.cctype.CWファイル

`commondefs.cctype.CW` ファイルは、NITRO-System の `commondefs` ファイルからインクルードされます。このファイルでは、NITRO-System ライブラリで使用されているマクロスイッチの設定が行われています。

© 2004-2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。