

# ビルドシステム

## ソースツリーの説明

Ver 1.2.0

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	クイックスタート .....	4
2.1	開発ツールの用意 .....	4
2.2	NITRO-System ライブラリのビルド .....	4
2.2.1	NITRO-System パッケージの展開 .....	4
2.2.2	環境変数の設定 .....	4
2.2.3	NitroSystem ツリーのビルド .....	5
2.2.4	デモプログラムの実行 .....	5
2.2.4.1	ensata を起動 .....	5
2.2.4.2	実行ファイルのロード .....	5
2.2.4.3	ファイルの実行 .....	5
2.2.4.4	実行の停止 .....	5
2.3	ビルドツール .....	5
2.3.1	Makefile の記述 .....	6
2.3.2	ビルドスイッチ .....	6
2.3.3	ターゲット .....	6
3	ソースツリー .....	7
3.1	インクルード .....	7
3.2	ライブラリ .....	8
3.2.1	ライブラリファイルの命名規則 .....	8
3.3	ビルドツリー .....	9
3.4	ライブラリとデモのサブディレクトリ構造 .....	10
3.5	ビルドに必要なファイル .....	11
3.5.1	commondefs ファイル .....	11
3.5.2	modulerules ファイル .....	11
3.5.3	nnslibdefs ファイル .....	11
3.5.4	commondefs.cctype.CW ファイル .....	11

## 表

表 2-1	利用可能なビルドスイッチ .....	6
表 2-2	利用可能なターゲット .....	6

## 図

図 3-1	ソースツリー第1階層のディレクトリ .....	7
図 3-2	インクルードディレクトリ構造 .....	7
図 3-3	ライブラリディレクトリ構造 .....	8
図 3-4	ビルドディレクトリ構造 .....	9
図 3-5	ライブラリとデモの基本ディレクトリ構造 .....	10

## 改訂履歴

版	改訂日	改 訂 内 容	承認者	担当者
1.2.0	2006-05-29	・NITRO-SDK のサウンドパッチに関する記述を削除。		西田泰
1.1.0	2004-10-12	・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。		西田泰
1.0.8	2004-08-10	図 3-4 ビルドディレクトリ構造を修正。		西田泰
1.0.7	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。		西田泰
1.0.6	2004-04-12	・誤字の修正。		西田泰
1.0.5	2004-04-08	・NITRO-SDK, IS-NITRO-CHARACTER 等の用語を統一。 ・商標表記を修正。		西田泰
1.0.4	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。		西田泰
1.0.3	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。		西田泰
1.0.2	2004-02-20	図 3-5、図 3-6 に depend ディレクトリを追加。		西田泰
1.0.1	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。		西田泰

# 1 はじめに

このドキュメントでは、NITRO-System のライブラリとデモプログラムをビルドする為の手順と、NITRO-System のソースツリーについてご説明しています。なお、NITRO-System ライブラリは、NITRO-SDK の上に構築されておりますので、NITRO-SDK のドキュメントも合わせてお読みください。

## 2 クイックスタート

### 2.1 開発ツールの用意

NITRO-System は、NITRO-SDK の上に構築されています。その為、NITRO-System ライブラリがビルドできる環境は、NITRO-SDK がビルドできる環境と合わされています。NITRO-System ライブラリを使用する場合には、NITRO-SDK をビルドすることができる環境が整っている必要があります。

- (1) 現在、NITRO-System では、以下の Windows 環境でビルドができることを確認しています。

Microsoft Windows 2000 Professional

- (2) NITRO-System のビルド(コンパイルなど)を行なうためには、下記のツールと SDK が必要です。

- CodeWarrior for NITRO
- Cygwin または MinGW(MSYS ツール)
- NITRO-SDK

- (3) デバッグを行なうためには下記のいずれかのツールが必要です。

- NITRO エミュレータ ensata
- IS-NITRO-EMULATOR

### 2.2 NITRO-System ライブラリのビルド

ここでは、NITRO-System のライブラリとデモプログラムをビルドする為の手順について記されています。

#### 2.2.1 NITRO-System パッケージの展開

NITRO-System のパッケージをローカルディスクの任意の場所に展開します。NITRO-System のパッケージは zip 形式で圧縮されていますので、各種解凍ツールを用いて展開してください。パッケージを展開しますと、NitroSystem という名前のディレクトリが作成されます。

#### 2.2.2 環境変数の設定

環境変数 NITROSYSTEM\_ROOT に、展開されたディレクトリ NitroSystem の絶対パスを設定します。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。

### 2.2.3 NitroSystem ツリーのビルド

環境変数の設定が終わりましたら、NITRO-System ライブラリのルートディレクトリ (NitroSystem ディレクトリ) で `make` を実行すれば、ライブラリとサンプルのビルドが行われます。

### 2.2.4 デモプログラムの実行

ビルドが正常に行なわれたことを確認するためにサンプルプログラムを動かしてみます。ここでは、NITRO のエミュレータである `ensata` を使用した場合の手順を説明しています。

#### 2.2.4.1 ensata を起動

`ensata` のアイコンをダブルクリックし、`ensata` を起動します。`ensata` には、DirectX を使用して描画処理を行なう `ensata_dx.exe` と、描画処理に DirectX を使用しない `ensata.exe` とがありますので、ご自分の PC 環境に応じてご使用ください。双方ともコントローラ入力のため `DirectInput8(DINPUT8.DLL)` が要求されます。DirectX がインストールされていない PC をお使いの方は、Microsoft の WEB サイトより取得してください。

#### 2.2.4.2 実行ファイルのロード

`ensata` のウィンドウ上でマウスの右クリックを行ない、「**NITRO ファイルの読み込み**」を選びます。表示されるダイアログで、実行させたい `nef` ファイルを指定します。

#### 2.2.4.3 ファイルの実行

`ensata` のウィンドウ上の「**連続フレーム**」ボタン (フレームと書かれている横にある2つの目のボタン) を押します。

#### 2.2.4.4 実行の停止

`ensata` のウィンドウ上の「**ストップ**」ボタン (手の平の絵が描かれているボタン) を押すと、実行が停止します。

## 2.3 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お使いになれます。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` をご参考になしてください。

なお、NITRO-System のビルドシステムは、NITRO-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、NITRO-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、NITRO-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、NITRO-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。

### 2.3.1 Makefile の記述

NITRO-System を用いた Makefile の記述やビルドの仕方は、NITRO-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。NITRO-SDK の Makefile との唯一の違いは、commondefs ファイルと modularules 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)。

- NITRO-SDK の Makefile でのインクルード  

```
include $(NITROSDK_ROOT)/build/buildtools/commondefs
include $(NITROSDK_ROOT)/build/buildtools/modularules
```
- NITRO-System の Makefile でのインクルード  

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

### 2.3.2 ビルドスイッチ

NITRO-System では、NITRO-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 NITRO-SDK のドキュメントをご覧ください。

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

### 2.3.3 ターゲット

NITRO-System ライブラリでは、NITRO-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。

ターゲットについての詳しい情報は、下記の NITRO-SDK のドキュメントをご覧ください。

[\\$NitroSDK/docs/SDKHowTo/HowToBuildSDKTree.rtf](#)

## 3 ソースツリー

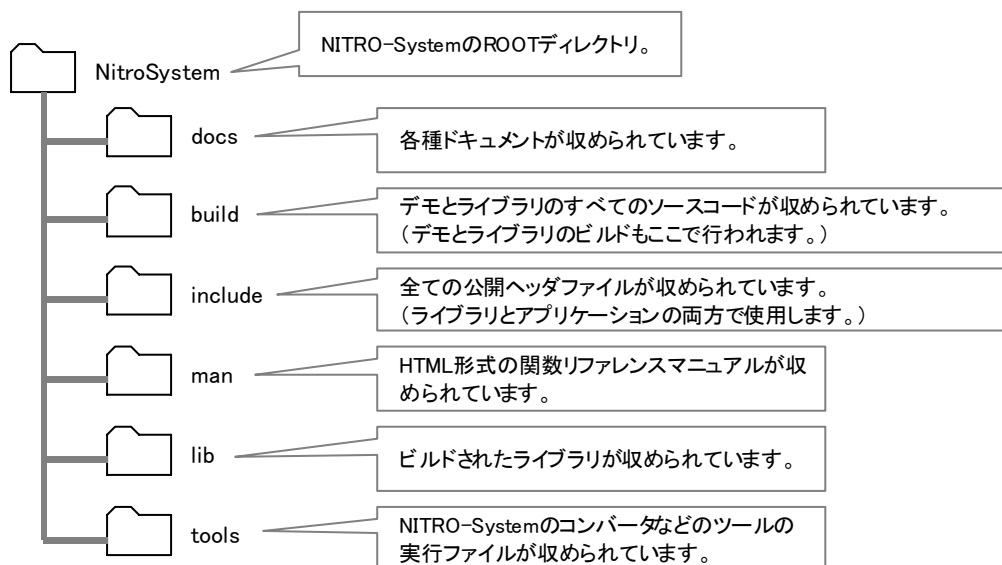


図 3-1 ソースツリー第1階層のディレクトリ

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要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

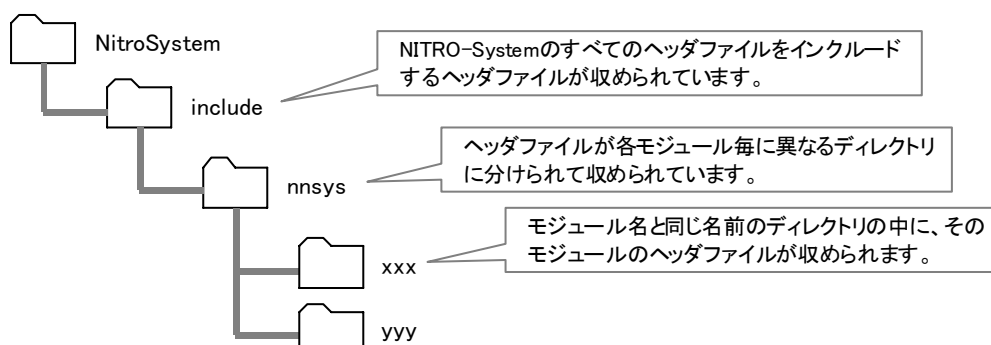


図 3-2 インクルードディレクトリ構造

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

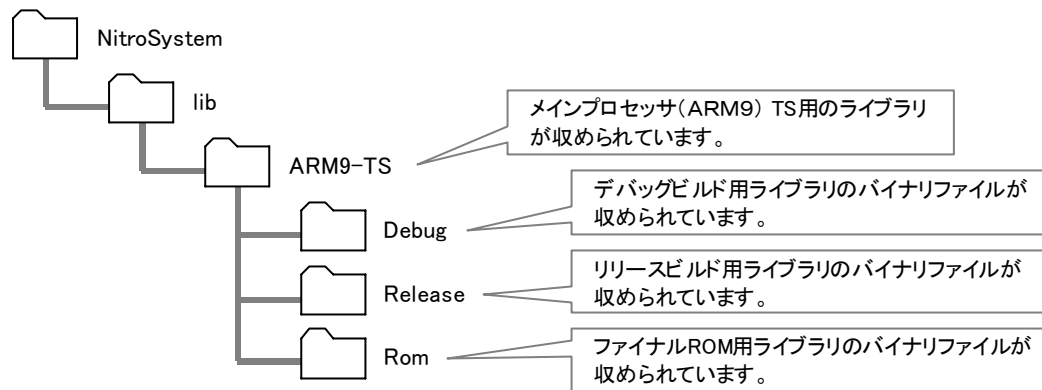


図 3-3 ライブラリディレクトリ構造

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

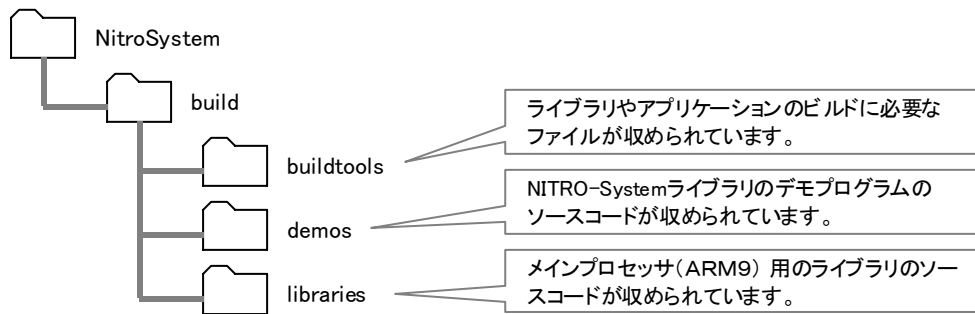


図 3-4 ビルドディレクトリ構造

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 `modulerrules` 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。

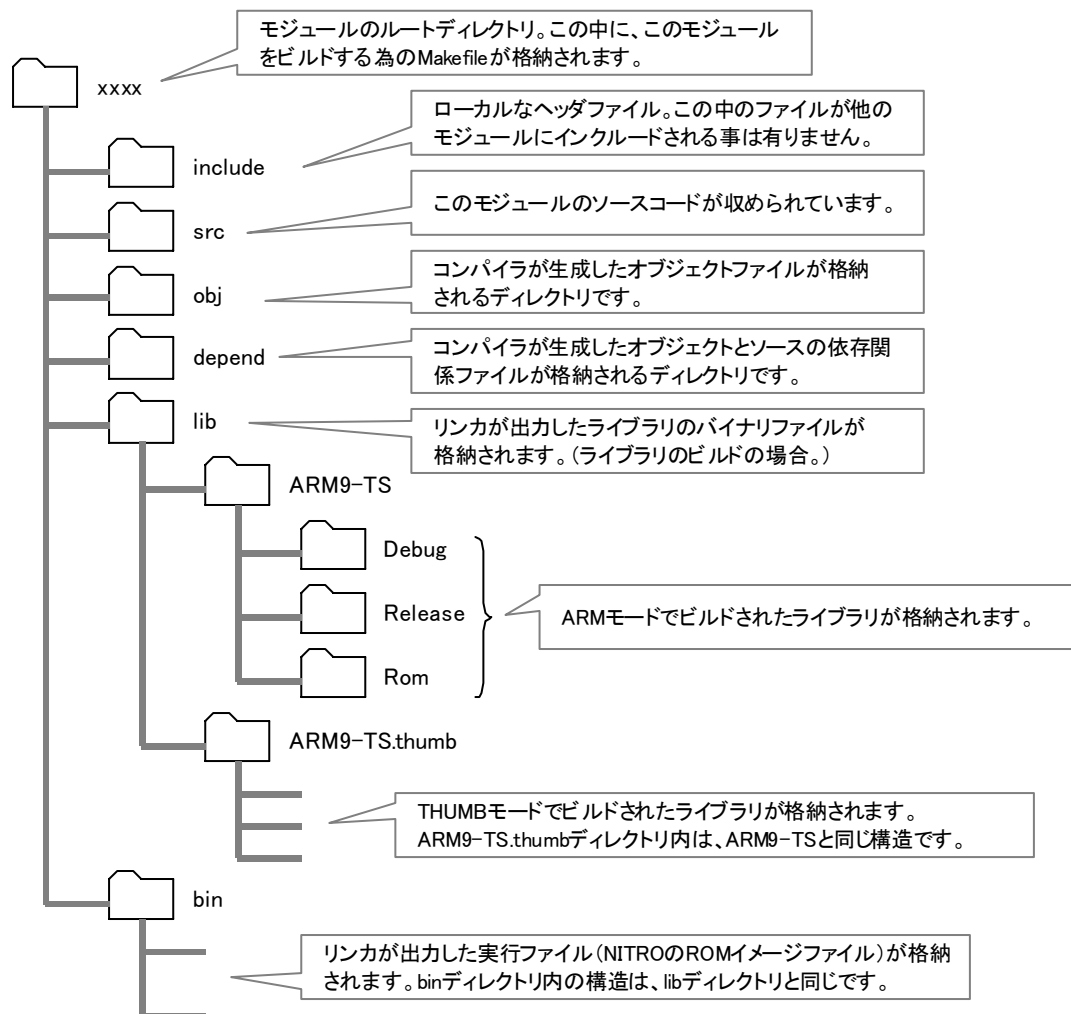


図 3-5 ライブラリとデモの基本ディレクトリ構造

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 NITRO-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、NITRO-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 NITRO-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、NITRO-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、NITRO-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、NITRO-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。

Microsoft、Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

CodeWarrior は Metrowerks Inc. の米国およびその他の国における登録商標または商標です。

その他、記載されている会社名、製品名等は、各社の登録商標または商標です。

© 2004-2006 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・複製・転写・頒布・貸与することを禁じます。