

Quick Start Guide

TWL-SDK

2008/11/04

任天堂株式会社

0. はじめに

本ドキュメントでは TWL-SDK のインストールと簡単なアプリケーションの作成方法を説明します。

以下のような順に説明します。

1. 開発ツールの用意
2. 環境変数の設定
3. TWL-SDK ツリーのビルド
4. サンプルを動かしてみる
5. 簡単なプログラムを書いてみる
6. ビルドスイッチ

1. 開発ツールの用意

現在 TWL-SDK では以下の Windows 環境でビルドできることを確認しています。

- Microsoft Windows XP Professional Service Pack 2

TWL-SDK 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が必要です。

- CodeWarrior for NINTENDO DSi
- Cygwin

また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。

- IS-TWL-DEBUGGER (software/hardware)
対応プラットフォーム: LIMITED ROM
HYBRID ROM (DSi 上で動作)
HYBRID ROM (DS/DS Lite 上で動作)
NITRO ROM
- IS-NITRO-DEBUGGER/IS-NITRO-EMULATOR
対応プラットフォーム: HYBRID ROM(DS/DS Lite 上で動作)
NITRO ROM

※ IS-NITRO-DEBUGGER のみがインストールされた環境でプログラムをビルドしたコードを IS-NITRO-DEBUGGER で実行する場合にはデバッガ上に文字列を表示させることが可能ですが、同コードを IS-TWL-DEBUGGER(software/hardware) で実行した場合には表示させることができません。IS-TWL-DEBUGGER(software/hardware) で実行した場合にも文字列を表示させたい場合には IS-TWL-DEBUGGER(software) がインストールされた環境でプログラムをビルドする必要があります。

CodeWarrior for NINTENDO DS、IS-TWL-DEBUGGER (software)、IS-NITRO-DEBUGGER に関しては NTSC-ONLINE から取得することができますが、その他の各ツールに関しては配布元にお問い合わせください。

Cygwin のインストールについては、TWL-SDK 展開後の \$TwlSDK/docs/SDKTools ディレクトリ内の CygwinPackageList.pdf をご覧ください。

インストールについての実作業については各ツールのドキュメントをご覧ください

2. 環境変数の設定

TWL-SDK では TWLSDK_ROOT, TWLSDK_PLATFORM の2つの環境変数を設定する必要があります。

環境変数 TWLSDK_ROOT には、展開されたディレクトリ TWL-SDK の絶対パスを設定します。何も指定されていない場合は C:¥TwlSDK が設定されます。以後このディレクトリのことを \$TwlSDK と表記します。

環境変数 TWLSDK_ROOT は TWL-SDK 展開後の \$TwlSDK/setup でも設定することができます。setup ファイルを使用した場合の設定方法は以下です。

```
% cd $TwlSDK
% source setup
```

環境変数 TWLSDK_PLATFORM には、ビルド対象とするプラットフォームを設定します。この環境変数を設定することによって、生成する必要のあるプラットフォーム用コードを決定します。
何も指定されていない場合は、ビルド時にエラーで停止します。
TWLSDK_PLATFORM 等のビルドオプションに関しては TWL-SDK 展開後の \$TwlSDK/docs/SDKRules/Rule-Defines.html を参照ください。

3. TWL-SDK ツリーのビルド

展開後の TWL-SDK にはあらかじめビルドされたライブラリやツールが含まれていますので、基本的には \$TwlSDK のトップからすべてをビルドする必要はありません。
例として、以下に 3D_Pol_LightColor サンプルデモをビルドする場合について示します。

Cygwin でシェル(bash など)を立ち上げ、\$TwlSDK/build/demos/gx/UnitTours/3D_Pol_LightColor に移動し

```
% make
```

と入力することでビルドが開始されます。

もし途中でエラーを出力して停止するようでしたら、ここまでの設定に間違いがある、あるいは本パッケージにバグがあることが考えられます。お手数ですが、ここまでの設定を見直された後で配布元へご連絡ください。

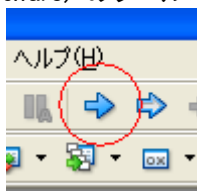
[2. 環境変数の設定] で環境変数 TWLSDK_PLATFORM に指定した値によって、どのプラットフォーム用コードを生成するのかが変わります。

TWLSDK_PLATFORM 以外のビルドオプションを指定しなかった場合、TWLSDK_PLATFORM が TWL の場合はデフォルトで ARM の Release ビルドの HYBRID ROM が生成され、TWLSDK_PLATFORM が NITRO の場合はデフォルトで ARM の Release ビルドの NITRO ROM が生成されます。
ROM の種類や生成方法に関しては TWL-SDK 展開後の
\$TwiSDK/docs/TechnicalNotes/AboutTwiApplication.pdf を参照ください。

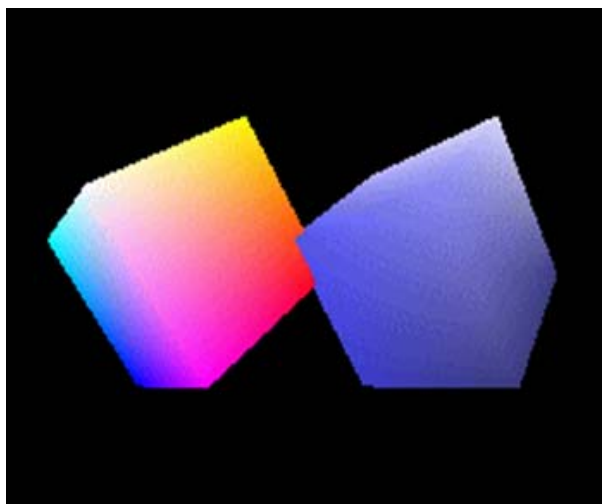
4. サンプルを動かしてみる

ビルドが正常に行なわれたかを確認するためにサンプルを動かしてみます。

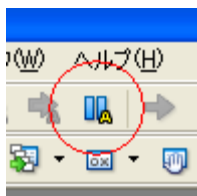
- 1) IS-TWL-DEBUGGER (software) を起動します。
(NITRO 用コードの場合は IS-NITRO-DEBUGGER)
- 2) [ファイル(F)] → [開く(O)] を選択します。
- 3) ダイアログで SRL ファイルを指定します。
ここでは先ほどのビルドで作成された下記のファイルを指定します。
\$TwiSDK/build/demos/gx/UnitTours/3D_Pol_LightColor/bin/ARM9-TS.HYB/Release/main.srl
(NITRO 用コードの場合は
\$TwiSDK/build/demos/gx/UnitTours/3D_Pol_LightColor/bin/ARM9-TS/Release/main.srl)
- 4) [デバッグ(D)] → [実行(R)] を選択、
もしくは IS-TWL-DEBUGGER (software) のツールバーの実行ボタン[下図 ○印] を押します。



- 5) IS-TWL-DEBUGGER (software) の上画面に以下のような立方体が表示されればOKです。
ちなみに右側の立方体の色は次々に切り替わります。



- 6) 停止ボタン[下図 ○印]を押すと bin ファイルの実行を停止します。



5. 簡単なプログラムを書いてみる

簡単なプログラムを書いてみましょう。

- 1) 適当な作業ディレクトリを作成し、以下のファイルをコピーします。

```
$TwiSDK/build/demos/template/src/main.c
$TwiSDK/build/demos/template/makefile
```

- 2) main.c を以下のように編集します。

16行目以降

```
#include <nitro.h>

void NitroMain(void)
{
    OS_Init( );
    OS_Printf( "Hello World of TWLid.¥n" );
    while (1){ }
}
```

← この行を追加

- 3) make コマンドを実行します。もし error や warning などの表示があったときは 2) での編集作業に誤りがないかを見直してください。

- 4) IS-TWL-DEBUGGER を起動し、前章と同様に[ファイル(F)] → [開く(O)] を選択し、作業ディレクトリの下
の bin/ARM9-TS.HYB/Release/main.srl を指定します。
(NITRO 用コードの場合は bin/ARM9-TS/Release/main.srl を指定)

- 5) 実行ボタンを押すとデバッグ出力ウィンドウに以下のような表示がでます。

Hello World of TWLid.

- 6) 停止ボタンを押すと bin ファイルの実行を停止します。

6. ビルドスイッチ

TWL-SDK には様々なビルドの設定があります。

デフォルトでは Release 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 Debug 版やRom 版のビルドを行なうことができます。これをビルドスイッチと呼びます。

ビルドスイッチについての詳しい情報は [\\$TwiSDK/docs/SDKRules/Rules-Defines.html](#) をご覧ください。