

CodeWarrior Development Studio for NINTENDO DSi V1.3 クイックスタート

システム要件

ハードウェア	Intel® Pentium® 1.4 GHz 互換のプロセッサ (またはそれ以上) の PC 512 MB メモリ (1 GB を推奨) CD-ROM ドライブ
オペレーティング システム	Microsoft® Windows XP SP2 または SP3 Windows Vista® (32 bit) SP1
ディスク容量	全体で 2 GB Windows システムドライブに 500 MB

本文書は、CodeWarrior for NINTENDO DSi V1.3 ソフトウェアをウィンドウズ PC にインストールする方法を説明しています。また、このソフトウェアを使用してニンテンドー DS またはニンテンドー DSi 用のプロジェクトを作成、ビルド、デバッグする方法を説明しています。

注 意 本ソフトウェアは、リリース前の製品名称を使用して作成されています。NITRO はニンテンドー DS 用の、TWL はニンテンドー DSi 用の名称です。

注 意 以下で説明している手順中、上級ユーザーは番号が付けられている手順をご参照ください。初心者の方にはアルファベットが付けられた、より詳細な説明が用意されています。

注 意 このクイック・スタート内の情報は最新でない可能性があります。最新の情報については、「Developer Notes」をご参照ください。

セクション A: ソフトウェアのインストール

IS デバッガ、Ensata エミュレータ、TWL SDK、および CodeWarrior ソフトウェアは、NTSC-ONLINE のウェブサイトから入手します。

<https://ntsc.nintendo.co.jp/>

各ソフトウェアを、以下の順序でインストールします：

1. **最新版の IS デバッガソフトウェアをインストールします。**
2. **最新版の TWL SDK をインストールします。**
3. **最新版の CodeWarrior Development Studio for NINTENDO DSi をインストールします。**
4. **ニンテンドー DS 開発用のオプションとして、最新版の Ensata エミュレータソフトウェアをインストールします。**

セクション B: IS デバッガのインストール

注 意 SDK と IDE は、DS と DSi の両方のビルドに使用できます。
IS デバッガは、対象とするプラットフォーム用の一つ、または両方をインストールする必要があります。

注 意 TWL SDK または IS デバッガがインストールされていなくても、CodeWarrior をインストールすることができます。CodeWarrior のインストール後に、TWL SDK または IS デバッガをインストールする場合には、バッチファイル {CW}\bin\setTWLSrcTree.bat を実行し、適切なソースツリーを作成する必要があります。詳細については、セクション D の、CodeWarrior のインストールを参照してください。

1. **最新版の IS デバッガソフトウェアをインストールします。**
 - a. ニンテンドー DSi 用のプロジェクトを開発する場合、IS-TWL-DEBUGGER ソフトウェアをインストールします。
 - b. ニンテンドー DS 用のプロジェクトを開発する場合、IS-NITRO-DEBUGGER ソフトウェアをインストールします。

注 意 インストーラが再起動を要求しなくても、IS デバッガソフトウェアパッケージのインストール後、必ずコンピュータを再起動してください。

セクション C: SDK のインストール

CodeWarrior ソフトウェアをインストールする前に、ニンテンドー DS およびニンテンドー DSi プラットフォーム用の SDK をインストールする必要があります。

1. SDK のインストール方法に従い、SDK をハードディスクの任意のフォルダ（例：C:\TwlSDK）へインストールします。

注 意 以前のリリースでは、CodeWarrior ツールのインストールフォルダを指す、以下の環境変数を作成する必要がありました。

- DS : CWFOLDER_NITRO
- DSi : CWFOLDER_TWL

本リリースの CodeWarrior インストーラでは、上記の環境変数が自動的に作成されます。ただし、以前のバージョンをインストールする場合は、従来通り環境変数を作成してください。

1. ウィンドウズの [コントロールパネル] を開く。
2. [システム] をダブルクリックする。
3. [詳細設定] タブをクリックする。
4. [環境変数] ボタンをクリックする。
5. [システム環境変数] 欄の [新規] ボタンをクリックする。
6. TWLSDK_ROOT 変数を、値 C:\TwlSDK で作成する

注 意 環境変数のより詳細な情報については、TWL SDK の docs フォルダ内にある、QuickStartForSDK を参照してください。

7. [OK] ボタンをクリックし、開いているダイアログやウィンドウを閉じてウィンドウズに戻る。

注 意 環境変数の作成後、コンピュータを再起動する必要があります。

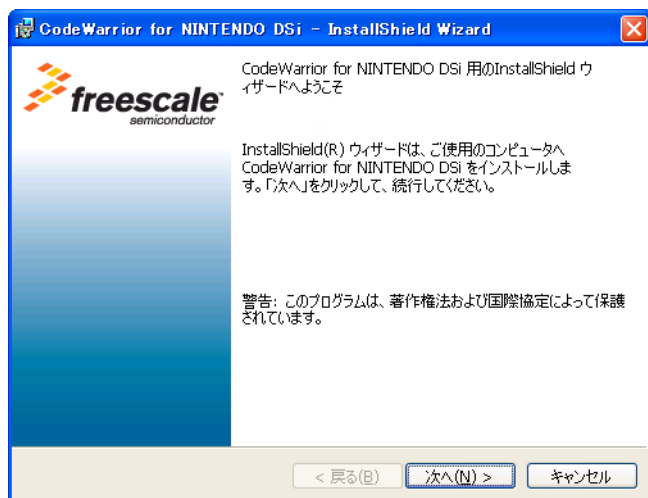
セクション D: CodeWarrior Development Studio のインストール

注 意 IS-NITRO-DEBUGGER または IS-TWL-DEBUGGER がインストールされていなくても、CodeWarrior をインストールすることができます。IS デバッガを後でインストールする場合には、ステップ 2 の手順に従ってください。

1. CodeWarrior ソフトウェアをインストールします。

- CodeWarrior Development Studio のインストーラを任天堂から入手します。この製品をインストールできるのは、任天堂が認可した開発者だけです。
- インストーラを起動します。インストールウィザードが実行されるので、ウィザードの手順に従ってインストールを進めます。

CodeWarrior インストーラ



- 「TWL SDK の場所」ページが表示されるまで、画面上の指示に従ってインストールを進めます。
- インストーラが TWL SDK のインストール先のフォルダを特定できない場合、[変更] ボタンをクリックし、手動でインストール先のフォルダを指定してください。
- [次へ] をクリックして、インストールを続けます。
- ウィザードの残りの指示に従って、インストール作業を進めます。

- g. ウィザードの指示に従って、コンピュータを再起動します。
- h. 再起動時に、インストーラは以下のパスを使用する「ソースツリー」を作成します。

- DS : TWLSDK_ROOT および IS_NITRO_DIR
- DSi : TWLSDK_ROOT および IS_TWL_DEBUGGER_DIR

2. CodeWarrior のインストール後に IS デバッガをインストールする場合には、以下の手順に従って CodeWarrior のインストールを完了してください。この手順により、ソースツリーが適切に作成されていることを確実にします。

- a. IS-NITRO-DEBUGGER または IS-TWL-DEBUGGER をインストールします。インストール時に、必要な環境変数が作成されます：
 - DS : IS_NITRO_DIR
 - DSi : IS_TWL_DEBUGGER_DIR
- b. CodeWarrior インストールフォルダ内の、以下のバッチファイルを実行します：
{CW}\bin\setTWLSrcTree.bat

セクション E: Ensata エミュレータのインストール

任天堂の Ensata エミュレータは、ニンテンドー DS の開発にだけ使用できる、ソフトウェアエミュレータです。

1. Ensata のインストール方法に従い、Ensata エミュレータをハードディスクの任意のフォルダ（例：C:\NitroSDK\ensata）へインストールします。
2. 以下のフォルダにある、CodeWarrior の Ensata デバッガ初期化ファイル `est_cw_debugger.ini` をテキストエディタで開きます。
{CodeWarrior}\bin\Plugins\Support\Nitro\IS
3. Ensata 実行ファイルのパスを指すように、変数を変更します。
例)
[control]
ensata path=C:\NitroSDK\ensata\Release\ensata.exe
4. `est_cw_debugger.ini` ファイルをディスクに保存します。

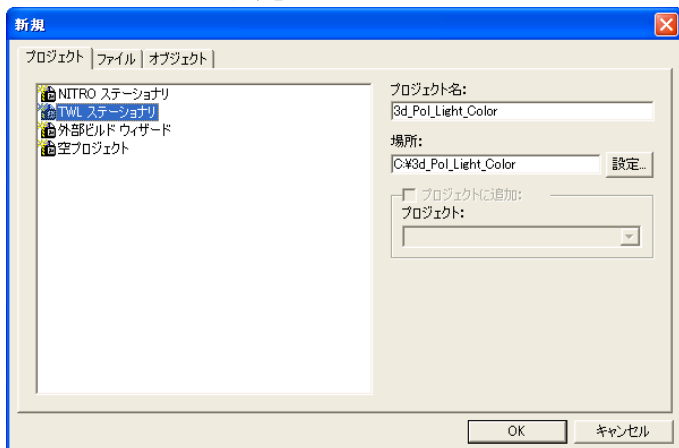
セクションF: プロジェクトの作成、ビルド、デバッグ

1. プロジェクトを作成します。

ウィンドウズのタスクバーから「スタート」→「すべてのプログラム」→「FreescaleCodeWarrior」→「CW for NINTENDO DSi V1.3」→「CodeWarrior IDE」の順に選択します。－ IDE が起動してメインウィンドウが表示されます。

- a. IDE のメニューバーから、「ファイル」→「新規」を選択します。
－ 「新規」ダイアログが表示されます。

「新規」ダイアログ



- b. 左側のペインで、対象とするプラットフォームに応じて、以下のどちらかを選択します：
－ DS : **NITRO ステーションナリ**
－ DSi : **TWL ステーションナリ**
- c. 「プロジェクト名」テキストボックスに、3d_Pol_Light_Color を指定します。
- d. 「場所」テキストボックスに、プロジェクトを作成するパスを指定します。

注 意 「場所」テキストボックスに指定したい場所が表示されない場合は、「設定」をクリックして「新規プロジェクト作成」ダイアログを表示し、場所を選択してください。

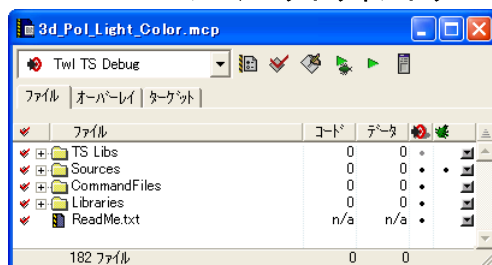
- e. 「OK」をクリックします。－「新規プロジェクト」ダイアログが表示されます。

「新規プロジェクト」ダイアログ



- f. 作成するプラットフォームに応じたステーションリを選択します。
例) 「SDK5.0」→「Hybrid Application」→「C」
- g. 「OK」をクリックします。－IDE がプロジェクトを作成します。－「3d_Pol_Light_Color.mcp」プロジェクトウィンドウが表示されます。

プロジェクトウィンドウ



2. 既存の makefile からプロジェクトを作成するには、ライブラリとソースのファイルを makefile で確認します。

注 意 プロジェクトのステーションリには予め標準の SDK ライブラリが指定されていますが、追加で必要なライブラリは自分で追加する必要があります。

- a. 以下のフォルダから「commondefs.gx.demolib」ファイルを開きます。
{TwlSDKFolder}\build\buildtools
- b. 次の行を見つめます。

LLIBRARIES += libDEMO\$(TWL_LIBSUFFIX).a

この行は、プロジェクトにライブラリファイルを追加する必要があることを示しています。(例: libDEMO.TWL.HYB.a)

- c. 以下のフォルダにある makefile を開きます。

```
{TwlSDKFolder}\build\demos\gx\UnitTours\  
3D_Pol_LightColor
```

- d. makefile 内で「SRCS = main.c」の行を探します。この行は、ソースファイル main.c をプロジェクトに追加する必要があることを示しています。

3. プロジェクトにソースファイルを追加します。

- a. エクスプローラで、以下のフォルダから main.c を探します。

```
{TwlSDKFolder}\build\demos\gx\UnitTours\  
3D_Pol_LightColor\src
```

- b. この main.c ファイルをコピーして、作成した 3d_Pol_Light_Color プロジェクトフォルダの main.c を置き換えます。

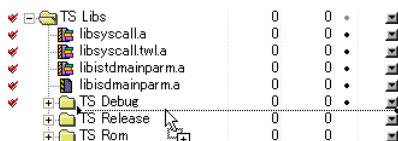
4. プロジェクトにライブラリファイルを追加します。

- a. {SubDir} は、Debug、Release または Rom のターゲットです。作成したプロジェクトの対象プラットフォームに応じて、使用するファイルを決定します。

プラット フォーム	ファイル
DS 専用	{TwlSDKFolder}\build\demos\gx\UnitTours\ DEMOLib \lib\ARM9-TS\{SubDir}\libDEMO.a
DSi 専用	{TwlSDKFolder}\build\demos\gx\UnitTours\ DEMOLib \lib\ARM9-TS.LTD\{SubDir}\ libDEMO.TWL.LTD.a
DS / DSi Hybrid	{TwlSDKFolder}\build\demos\gx\UnitTours\ DEMOLib \lib\ARM9-TS.HYB\{SubDir}\ libDEMO.TWL.HYB.a

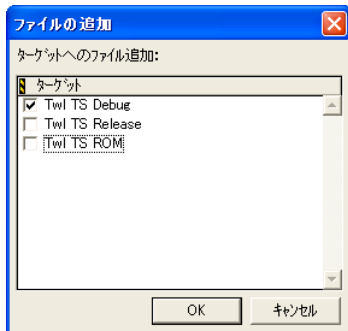
- b. ファイルをドラッグして、IDE のプロジェクトウィンドウの「TS Libs」→「TS {SubDir}」の下にドロップします。－「ファイルの追加」ダイアログが表示されます。

TS Debug ファイルグループにライブラリをドロップする



- c. 「ファイルの追加」ダイアログで、「Twl TS {SubDir}」チェックボックスだけを選択します。

「ファイルの追加」ダイアログ (DSi)

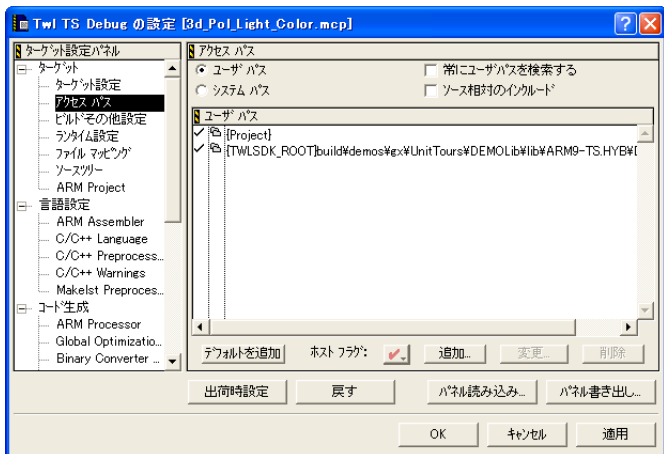


- d. 「OK」ボタンをクリックします：

ライブラリファイルが「Twl TS {SubDir}」ターゲットに追加されます。

- e. IDE のメニューバーから、「編集」→「Twl TS {SubDir} の設定」を選択し、ターゲット設定ウィンドウを開きます。
- f. ターゲット設定ウィンドウで、「ターゲット」→「アクセスパス」を選択します。

アクセスパスパネル

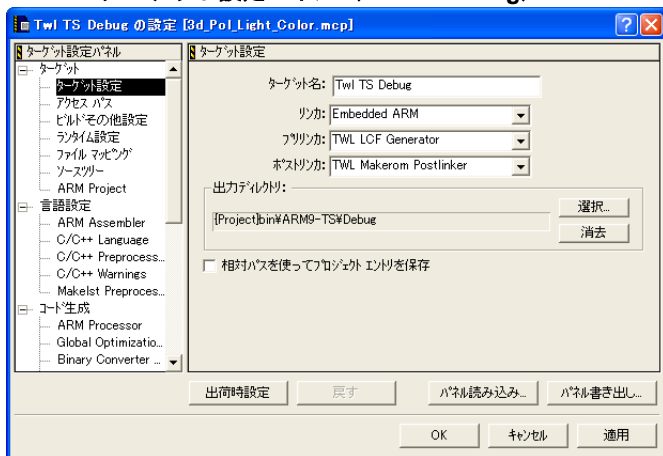


- g. 「追加」ボタンをクリックし、「フォルダの参照」ダイアログを開きます。
- h. 「パスの種類」に、TWLSDK_ROOT を設定します
- i. 以下のフォルダを選択します：
`{TWLSDK_ROOT}\build\demos\gx\UnitTours\DEMOLib\Include`
- j. 「OK」をクリックすると、パスが追加されます。

5. リンカコマンドファイルの設定を調整します。

- a. ターゲット設定ウィンドウで、「ターゲット」→「ターゲット設定」パネルを選択します。
- b. 「プリリンカ」に、以下のどちらかを選択します。
 - DS : **NITRO LCF Generator**
 - DSi : **TWL LCF Generator**

ターゲット設定パネル (Twi TS Debug)



c. 以下のどちらかを行います：

- － DS：「リンカ」→「Nitro LCF Prelinker」パネルを選択し、「Address」に 0x02000000 が設定されていることを確認します。
- － DSi：「リンカ」→「TWL LCF Prelinker」パネルを選択し、「Address」に 0x02004000 が設定されていることを確認します。

d. 「OK」ボタンをクリックし、ターゲット設定ウィンドウを閉じます。

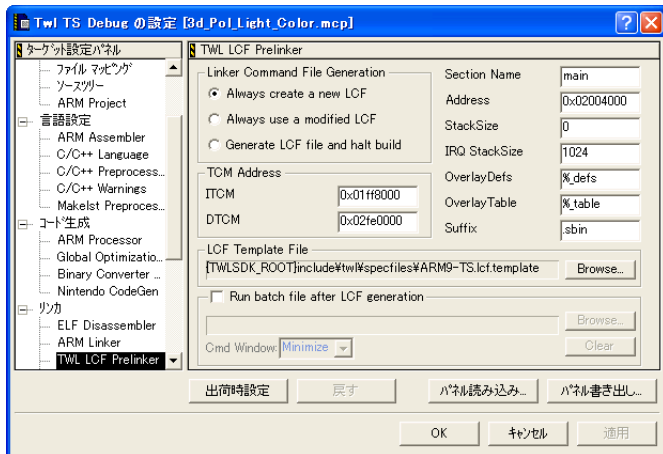
6. プロジェクトをビルドします。

a. IDE のメニューバーから、以下のどちらかを選択します：

- － DS：「プロジェクト」→「デフォルトターゲットを設定」→「Nitro TS Debug」
- － DSi：「プロジェクト」→「デフォルトターゲットを設定」→「TWL TS Debug」


b. IDE のメニューバーから、「プロジェクト」→「メイク」を選択します。
 ー IDE がファイルを更新し、コードをリンクしてアプリケーションを生成します。

TWL LCF Prelinker ターゲット設定パネル

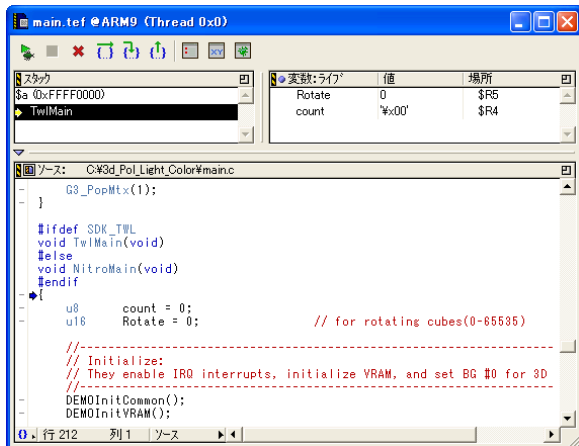



7. プロジェクトをデバッグします。

- IDE のメニューバーから、「プロジェクト」→「デバッグ」を選択します。ー IDE は、プログラムのアセンブル、コンパイル、リンクを実行し、デバッガウィンドウを表示します。

プログラムカウンタアイコン  は、カレントステートメント（次に行われるステートメント）を示します。

デバッガウィンドウ





- b. デバッガウィンドウの左端の列で、ブレークポイントを設定する行のダッシュ（-）をクリックします。- 行の横にブレークポイントアイコン  が表示されます。

ブレークポイントの設定

```

-  #ifdef SDK_TWL
-  void TwlMain(void)
-  #else
-  void NitroMain(void)
-  #endif
-  {
-  u8      count = 0;
-  u16     Rotate = 0;                                // for rotating cubes(0-85535)
-
-  //-----
-  // Initialize:
-  // They enable IRQ interrupts, initialize VRAM, and set BG #0 for 3D
-  //-----
-  DEMOInitCommon();
-  DEMOInitVRAM();
-  }
```

- c. 「実行」ボタン  をクリックします。- プロセッサは、ブレークポイントに達するまでの（ブレークポイントのステートメントを除く）全てのステートメントを実行し、ブレークポイントのステートメントで停止します。
- d. 「ステップオーバー」ボタン  を数回クリックして、ソースコードのステートメントをステップします。
- e. 再度「実行」ボタンをクリックし、プログラムの実行を再開します。
- 液晶画面にプログラムの出力が表示されます。
- f. IDE のメニューバーから、「デバッグ」→「終了」を選択します。
- デバッグセッションが終了します。開いている全てのウィンドウを閉じてかまいません。

おめでとうございます！

CodeWarrior ソフトウェアをインストールし、
CodeWarrior IDE を使用して、プロジェクトの作成、ビルド、
デバッグを行いました。

Freescale および Freescale のロゴは、Freescale Semiconductor, Inc. の商標です。CodeWarrior は、米国、その他の国における Freescale Semiconductor, Inc. の登録商標です。その他、記載されている製品名またはサービス名は該当各社の商標または登録商標です。
Copyright © 2005-2010 by Freescale Semiconductor, Inc. All rights reserved.

本書の情報は、Freescale Semiconductor 製品を使用されるシステムおよびソフトウェア開発者の支援のみを目的としています。したがって、本書の内容はその情報に基づく集積回路の設計 / 製造に関する明示的または暗黙の著作権上の権利を示すものではありません。

当社は、本書に記載した製品を予告なく変更する権利を有します。当社は、当社製品の特定の目的に対する適合性に関して保証または表明するものではなく、また製品または回路の適用または使用に起因するいかなる責務を負わず、特に間接的または偶発的な損害に関しては制限を設けずあらゆる責務を排除します。また、当社の特許権またはその他の権利に基づく法的な許諾を行うものではありません。Freescale Semiconductor のデータシート / 仕様に記載される「標準 (Typical)」パラメータは各用途において変化する場合があります、実際の性能は長期間で変動する可能性があります。「標準」パラメータを含むすべての動作パラメータは、利用者側で技術担当者が使用環境に応じて適切な値に設定することが求められます。当社の製品は、外科的に人体に移植することを意図したシステムの構成部品として、または、他の生命維持を意図した用途に、または、当社の製品の不具合により人体に危害を加えたり死に至らしめるかもしれない状況が発生するような用途に使用するために、設計、意図または認可されているものではありません。購入者が万一このような意図または認可されていない用途のために当社の製品を購入あるいは使用する場合、購入者は、当社およびその役員、従業員、子会社、関連会社、代理店に対し、直接または間接を問わず、当該使用に関連した傷害や死についてのすべての申し立て（たとえ、当社が部品の設計や製造において不注意であったという主張であったとしても）から生ずるすべての請求、費用、損害、および相当の弁護士費用を補償し、被害が及ばないものとするものとします。

連絡先

日本	フリースケール・セミコンダクタ・ジャパン株式会社 〒153-0064 東京都目黒区下目黒 1-8-1 アルコタワー 15 階
U.S.A	Freescale Semiconductor, Inc. 7700 West Parmer Lane Austin, TX 78729 U.S.A.

改訂：2010/3/19-JP2010/4/7

