

Release Notes

NITRO-SDK

2004/04/05

任天堂企画開発本部 環境制作グループ

Version: NitroSDK-1.0

本パッケージについて

本パッケージはNITRO上でのアプリケーション開発効率を高めるために用意されたライブラリ群です。ハードウェアレジスタを抽象化し、視認性の高いソースコードを作成するお手伝いをいたします。またメモリや割り込みなどのシステムリソース管理の標準的な機構を提供します。

パッケージに含まれるもの

- 環境制作グループ作成版NITRO-SDKライブラリ（グラフィクス・OSシステム etc）
- オンライン版関数リファレンスマニュアル
- NITRO機能別デモプログラム
- 開発ターゲットの切り替えを統合したmakeシステム

今回の変更点

今回のリリースでの変更点をご紹介します。過去の変更点については、オンラインリファレンスマニュアルを参照してください。[\\$NitroSDK/man/ja_JP/changelog.html](#) にあります。

1 全体的な変更点

1.1 NITRO-SDK への改称

irisSDK を NITRO ハードウェアに対応させたことに伴い、SDK の名称を NITRO-SDK に改めました。具体的には、ヘッダファイル名、ディレクトリ名、環境変数、コンパイルスイッチ、ドキュメントなどに変更があります。他にも、メイン関数が `NitroMain()` に変更されています。

1.2 ELF ファイルの拡張子の変更

ELF ファイルの拡張子を `.nef` に変更しました。他の NITRO 開発関連ツールも、随時 `.nef` に対応予定です。

1.3 サンプルコードのディレクトリの変更

サンプルコードのディレクトリを `$NitroSDK/build/tests` から `$NitroSDK/build/demos` に変更しました。また、過去のサンプルコードに含まれる `#ifdef SDK_AUTOTEST~#endif` は、SDK の内部テスト用のコードですので、サンプルコードをコピーしてお使いの場合は、該当箇所を削除しても構いません。

1.4 サンプルコードの追加

多くのグラフィックスサンプルコードが追加されています。

1.5 ドキュメントの追加

NITRO-SDK と ABGSDK の両方をインストールして共存させる方法についてのドキュメントを追加しました。

\$NitroSDK/docs/HowToJoinAGBDevEnv.txt にあります。

1.6 オンライン関数リファレンスマニュアルの修正

マニュアル左ウィンドウの3Dジオメトリ(G3*)がらジオメトリコマンドサイクルの一覧に移るためのリンクを追加しました。

また、MI_SetWRAMBank() を正しい関数名 MI_SetWramBank() に修正しました。これに伴い、HTML ファイル名も変更しています。

2 開発環境の変更点

2.1 make 単位での Thumb 版のコード生成方法の提供

makeコマンドの引数または環境変数で NITRO_CODEGEN=THUMB を定義することで、Thumb 版が生成されます。NITRO_CODEGEN=ARM または何も指定しない場合は、ARM版が生成されます。

2.2 関数単位での Thumb 版のコード生成方法の提供

ARM/thumb のコードを関数単位で切り替えるための設定を含むヘッダファイルを用意しました。

ARMコードを生成	\$NitroSDK/include/nitro/code32.h
Thumbコードを生成	\$NitroSDK/include/nitro/code16.h
初期設定に戻す	\$NitroSDK/include/nitro/codereset.h

なお、途中バージョンのPR2版までは codereset.h をインクルードしても常に thumb に切り替わってしまう不具合がありました。

2.3 C++のサポート

C++でのコード作成をサポートするために、以下の変更を行いました。

- NitroMain() を呼び出す前に NitroStartUp() 関数とスタティックコンストラクタを呼び出す処理を追加
- C++ で問題のあったヘッダファイルの修正
- C++ のサンプルコードの追加 (\$NitroSDK/build/demos/os/cplusplus-1)

NitroStartUp() 内でメモリ管理機構の初期化を行うことにより、スタティックコンストラクタでnew()関数を使用することができます。

これに合わせて、DEMOライブラリ (\$NitroSDK/build/demos/gx/UnitTours/DEMOLib) のヘッダファイルにも、Cのリンケージ指定である extern “C” { の記述を追加しました。ただし、このライブラリはあくまで UnitTours の実装を簡単にする目的で作られたライブラリであることに注意してください。

2.4 make関連ファイルの修正

build/buildtools/ 以下のファイルを修正することにより、以下の変更が加えられました。

- ソースファイルの依存情報ファイルを出力、参照するようになりました。
- `make -f` で正しく動作しない問題が修正されました。
- 大量のソースファイルを `SRCS` に指定したときに `make` が起動しない問題が修正されました。
- 出力コードを `ARM9/ARM7` で切り替えるためのマクロ変数 `CODEGEN_PROC` を `NITRO_PROC` に名称変更しました。

2.5 IDEのPostLinkerの追加

IDEのNITRO用のPostLinkerとして、`$NitroSDK/tools/elftobin/NitroPostLinker.bat` が用意されました。

2.6 elftobinで使用するARM7バイナリの変更

バイナリファイル作成時にデフォルトで結合する `ARM7` 用実行ファイルを新しく作成しました。これには `X`、`Y` ボタンの読み取り処理が含まれます。以前の `SDK` に含まれる `ARM7` 用実行ファイルと今回リリースされた `SDK` で作成された `ARM9` 用実行ファイルを結合した場合、`PAD_Read()` 関数が正常に動作しませんでしたので、注意してください。

ファイルの場所は、`$NitroSDK/build/components/idle/ARM7` 以下です。

2.7 IRQスタックサイズの設定方法の提供

LCFファイルに記述することで、IRQスタックのサイズを設定できるようにしました。

2.8 VRAMバンクの組み合わせ設定の追加

新しく、`80_EF`、`80_EG` のエントリを追加しました。VRAM をそれぞれ `EF`、`EG` の組み合わせで `BG` または `OBJ` に配置できます。

また、`SDK` に対応していない組み合わせで VRAM を配置したい場合のため、`SDK` ソース改変によるエントリの追加方法をドキュメントにまとめました。

`$NitroSDK/docs/SDKHowTo/HowToMakeMyVRamBankType.txt` にあります。

2.9 デバッガの例外ハンドラ呼び出しに関する変更

デバッガの例外ハンドラがある場合、ユーザの例外ハンドラを呼び出す前にデバッガの例外ハンドラを呼び出すように、仕様を変更しました。

2.10 MIファイル展開APIに対応する圧縮ツールの追加

データ圧縮ツール `ntrcomp` を追加しました。`$NitroSDK/build/tools/ntrcomp/` にあります。

3 ライブラリ (API) の変更点

3.1 X、Yボタンのサポート

`PAD_Read()` 関数で `X`、`Y` ボタンが読み取れるようになりました。ただし、ハードウェア仕様上 `X`、`Y` ボタンの入力により割り込みを発生させることはできません(`PAD_SetIrq()`参照)。

3.2 例外発生時のコンテキスト表示機構の追加

例外の発生時に、例外発生時点のコンテキストを表示する機能を追加しました。ただし、IRQ例外、FIQ例外、SWI 例外を除きます。また、ユーザのコールバックルーチンを呼び出すことも可能です。

3.3 PXIライブラリの追加

ARM9とARM7の通信を行うためのライブラリを追加しました。

3.4 GXS 関数の追加

サブ LCD に対応した API を用意しました。この API を使用したサンプルコードは、
\$NitroSDK/build/demos/gx/UnitToors 以下の Sub_ から始まるディレクトリにあります。

3.5 CPUメモリ関数の追加

メモリインターフェイスとしては MI_CpuClear() 等の DMA ではなく、CPU によるメモリ操作関数を用意しました。これに伴い、UTL ライブラリを廃止しました。

3.6 MI関数の修正

メモリインターフェイス関数 (MI_*)() に間して、以下の修正を加えました。

- MI_Dma*Async() で、コールバックに NULL が指定されていた場合、コールバックをセットせずに処理を進めるようにしました。バスのアクセス競合によるストールが発生しない場合、DMA コマンドの送信後、ウェイトせずに関数から戻ります。
- 新しく MI_DmaSend*() および MI_CpuSend*() を追加しました。

3.7 TCM領域にかかるDMAへの警告

ITCM および DTCM を含む領域を、DMA の転送元あるいは転送先に指定したとき、DEBUG ライブラリでは警告を出すようにしました。

3.8 起動時のメモリクリア処理の追加

crt0.c 内で DTCM をクリアするようにしました。これにより、起動時にスタック領域がクリアされるようになりました。また、BG/OBJ パレット、OAM もクリアします。

3.9 キャッシュ関連の関数の引数アドレスの丸め処理の追加

DC_StoreRange()、DC_FlushRange()、DC_TouchRange()、IC_InvalidateRange() で、開始アドレスを下位方向へ 32 バイト境界に丸めるように実装しました。

3.10 スレッド関数の変更

スレッドに関して、以下の変更が加えられました。

- OS_CreateThread() を、実行開始関数に渡す引数を指定できるように修正しました。
- スレッド優先度の範囲を 0~31 にしました。これに伴い、OS_InitThread() 呼び出し直後の、デフォルトのスレッド優先度を 16 に変更しました。
- スレッドの優先度を取得、設定する API (OS_SetThreadPriority() / OS_GetThreadPriority()) を追加しました。
- スレッドのスタック溢れをチェックする API (OS_CheckStack())を追加しました。
- 指定時間だけスレッドを休止させる API (OS_Sleep())を追加しました。
- スレッドシステムが初期化済みで使用可能であるかどうかを調べる API (OS_IsThreadAvailable())を追加しました。

3.11 OS_ReadMessage()の追加

メッセージキューの最初のメッセージを参照してコピーするだけの関数、OS_ReadMessage() を追加しました。OS_ReadMessage()は、OSReceiveMessage() とは異なり、メッセージを送ろうとしてブロックされているスレッドを復帰させません。また、メッセージキューの内容も変更しません。

3.12 タイマー関数の変更

タイマーに関して、以下の変更が加えられました。

- 16bit タイマー 1つを SDK 予約として割り当てることで 64bit 値を持つ時刻カウンタ(チックと呼びます)が使用できるようになります。この機構を使用する、しないは選択できます。なお、途中バージョンの PR1 版まではチックのシステムに不具合があり、ある桁以上にカウントアップしませんでした。
- チックの機能を有効にしている場合において、更にもう 1つ 16bit タイマーを SDK 予約として割り当てることで、指定時刻にコールバック関数を呼び出すアラーム機構を使用することができます。この機構を使用する、しないも選択できます。タイマー割り込みを多重化させることで、ハードウェアのタイマーの数よりも多くのコールバック関数を登録できます。なお、途中バージョンの PR2 までは周期アラームのハンドラ内で自身のアラームをキャンセルできませんでした。
- 秒、ミリ秒、マイクロ秒で指定された時間と、OS_Sleep や OS_SetAlarm で使用されるシステムクロックを基にしたチックカウント値との変換マクロを追加しました。

3.13 Vカウントアラームの実装

V カウント割り込みを多重化することで、複数のコールバック関数を登録できるようになりました。

3.14 OS_DisableInterrupts() 等におけるFIQの取り扱いの変更

OS_DisableInterrupts() で IRQ、FIQ の両方を停止していましたが、デバッグとの兼ね合いで IRQ のみを停止させるように変更しました。ただし、AD バスのロック/アンロック、コンテキストスイッチに関しては IRQ と FIQ の両方を停止させています。

3.15 VecFx16 型のサポート

Fx16 を要素とする 3D ベクトル型 VecFx16 をサポートしました。また、これに対応する算術関数 VEC_Fx16* を追加しました。

3.16 固定小数定義の追加

新しく 0.5 を表す小数定義を \$NitroSDK/include/nitro/fx/fx_const.h に追加しました。fx16 型、fx32 型、fx64 型それぞれについて、FX16_HALF、FX32_HALF、FX64_HALF として定義されています。

3.17 ARM7のレジスタ名の変更

reg_DISP_DISPSTAT レジスタおよび reg_DISP_VCOUNTER レジスタは所属カテゴリを GX に移し、それぞれ reg_GX_DISPSTAT および reg_GX_VCOUNTER に改名しました。

また、reg_OS_POWCNT は所属カテゴリを SND に移し、reg_SND_POWCNT に改名しました。

3.18 グラフィックス関数の変更

上記以外に、グラフィックス (GX/G2/G3/G3X) 関数の以下の変更が加えられました。

- 動的DL作成関数 G3*に、DL バッファオーバーフローチェックを追加しました。これは、DEBUG 版

でのみ有効となります。

- G2_SetBG*Control256x16Pltt() に、キャラクタベースブロックを指定するための引数を追加しました。
- 半透明テクスチャ A3I5 テクスチャに対応しました。

3.19 その他の変更

上記以外にも、以下の変更が加わっています。

- マクロの複文をなくしました。
- ARM / thumb 切り替えでの問題を回避するため、ヘッダファイルのインラインアセンブラを移動しました。
- 定数やアドレスを代入するアセンブラ表記において、ldconst、lda、ldr が使用されていましたが、すべて ldrに統一しました。

4 バグ修正

4.1 TEG ボードの VBLANK 中のメインバスへのアクセスに伴うハードバグの対策

TEG ボードでは、VBLANK 期間の初期にメインプロセッサのバスに特定パターンのデータが出力されたときに、ポリゴンの表示が乱れるというハードウェアバグがあります。GX_Init() でこのバグの対策を行いました。この対策では、Vカウントアラームを使用していますので、この対策を使用する場合は、V カウントアラームを停止させないようにしてください。

4.2 OS_StartTimer() のアサート判定の修正

DEBUG 版ライブラリにおいて、OS_StartTimer32() の id に OS_TIMER32_23 を設定した場合、または、OS_StartTimer48() の id に OS_TIMER48_123 を設定した場合にアサートが発生していたのを修正しました。

4.3 タイマー割り込みからの復帰時の不具合修正

OS_SetIrqFunction() でタイマー割り込みコールバックを設定し、タイマー割り込みを開始し、コールバックが呼び出されると、そこでタイマー割り込みを再設定してもコールバックから戻るときにクリアされていました。この問題は修正されました。

4.4 DTCMの位置変更のサポート

\$NitroSDK/include/nitro/hw/common/mmap_global.h の HW_DTCM の値を編集することで、DTCM の位置を変更できるようになりました。変更後は、ライブラリを作成しなおす必要があります。

4.5 例外ベクタの設定不具合の修正

デバッガが例外ベクタを使用している場合、NITRO-SDK での設定との共存のために例外ベクタをフックする処理に間違いがあり、正常に動作しませんでした。この問題は修正されました。

4.6 レジスタ表の修正

ARM9、ARM7のレジスタ表、\$NitroSDK/build/buildsetup/ioreg/io_reg_list*、\$NitroSDK/build/buildsetup/ioreg_sp/io_reg_list* が加筆修正され、ほぼ現仕様を網羅しました。主な追加項目は、EXI および PXI です。

4.7 IRQ チェックフラグセットに関する修正

IRQ 割り込みのチェックフラグのセットで、TIMER 割り込みの場合に誤ったビットにアクセスしてセットしていました。この問題は修正されました。

4.8 IRQ テーブルの修正

OS内部のIRQハンドラのテーブルに誤りがありました。この問題は修正されました。また、未使用のビットもテーブルに加えたため、ARM9で2つ、ARM7で3つの配列要素が増えました。

4.9 複数の V カウントアラームに関する修正

V カウントアラームに起動時間の近い複数の周期関数を登録した場合、最初の関数しか起動されない問題がありました。この問題は修正されました。この修正に伴い、起動時間に遅延許容カウントを指定できるようになりました。

4.10 Light パラメータの調整

Light 処理を行うサンプルコードについて、ライトカラーなどのパラメータの総和がかなり大きく適切ではありませんでした。これを調整しました。

また、DEMOLib のスペキュラライトに関するパラメータテーブルの定義を、コサインのn乗に近似するように修正しました。

4.11 グラフィックス関数の修正

上記以外に、グラフィックス (GX/G2/G3/G3X) 関数の以下の問題が修正されました。

- G3_Ortho() の実装に誤りがありました。
- G3_EndMakeDL() が正しいサイズを返さないことがありました。
- G3CS_LoadTexMtxEnv(), G3CS_LoadTexMtxTexCoord() で生成されるディスプレイリストに問題がありました。
- \$NitroSDK/build/library/gx/src/g3_util.c に不備があり、G3_Frustum の行列が正しく設定できませんでした。
- VRAM_A にテクスチャイメージのスロット 0 を確保して、GX_LoadTex() でイメージ転送を行うとエラーが発生し、止まってしまいました。
- GX_RegionCheck_OBJ で、OBJにVRAM_B を割り当てた状態で Texture ロードなどを実行した場合、ASSERT で停止していました。
- G3X_GetCommandFifoStatus からの返り値が正確ではありませんでした。
- G2_SetBG3ControlDCBump について、\$NitroSDK/include/nitro/gx/gx_bgcnt.h の ASSERT 判定に問題がありました。
- マクロの FX_MUL32x64C とインライン関数 FX_Mul32x64c の丸め処理が異なっていました。
- G3X_InitMtxStack による行列スタックの初期化の際、元の PROJECTION 行列のスタックレベルが0であった場合に、スタックアンダーフローを起こしていました。
- GX_HBlankOBJProc() および GXS_HBlankOBJProc() においてパラメータ proc の値と実際の動作が逆になっていました。
- G2_SetOBJAttr() および G2S_SetOBJAttr() において GX_OAM_MODE_BITMAPOBJ モードで GX_OAM_EFFECT_AFFINE を使った時に rsParam パラメータの指定が OAM に反映されませんでした。
- GX_DisableBankFor*() において内部動作に誤りがあり、正しく動作していませんでした。

