NITRO ROM ファイルシステム内部仕様書

2006/05/23

任天堂株式会社 開発技術部＆環境制作部

0. はじめに

NitroSDK では開発時に頻繁に発生する ROM 内のデータの更新作業を出来るだけ短時間で行なうための手法として簡易なファイルシステムを導入いたしました。このファイルシステムに関しての詳細は NitroROM ファイルシステム仕様書に記載されております。本ドキュメントは NitroROM ファイルシステム仕様書に記載されていない一般には公開しない情報をまとめた文書です。

1. NitroROM フォーマット

NITRO ROM ブロックの構造の非公開情報を示します。

a) ROM ヘッダ

ROM内登録データ

上記のアドレスは ROM アドレス(オフセット)を表わす

```
typedef struct
{
	//
	// 0x000 システム予約領域
	//
	u8		game_name[12];			// ソフトタイトル名
	u8		game_code[4];			// イニシャルコード
	u8		corp_code[2];			// メーカーコード
	u8		product_id;			// 本体コード
	u8		device_type;			// デバイスタイプ
	u8		device_size;			// デバイス容量

	u8		reserved_A[8];			// システム予約 A

	u8		:6;
	u8		for_korea:1;			// 韓国向け本体対応フラグ
```

```
    u8          for_china:1;                    // 中国向け本体対応フラグ
    u8          game_version;                   // ソフトバージョン

    //
    // 0x01F 諸フラグ
    //
    u8          comp_arm9_boot_area:1;          // ARM9 常駐モジュール圧縮フラグ
    u8          comp_arm7_boot_area:1;          // ARM7 常駐モジュール圧縮フラグ
    u8          inspect_card:1;                 // 検査カードフラグ
    u8          disable_clear_boot_pad:1;       // 非ブート領域クリア・ディセーブル
フラグ
    u8          :2;
    u8          warning_no_spec_rom_speed:1;    // ROMアクセス速度未設定フラグ
    u8          disable_detect_pullout:1;       // カード割り込みによる抜け検出禁
止フラグ

    //
    // 0x020 b) 常駐モジュール用パラメータ
    //
    //          ARM9
    void*       main_rom_offset;                // 転送元 ROM アドレス
    void*       main_entry_address;             // 実行開始アドレス(未実装)
    void*       main_ram_address;               // 転送先 RAM アドレス
    u32         main_size;                      // 転送サイズ

    //          ARM7
    void*       sub_rom_offset;                 // 転送元 ROM アドレス
    void*       sub_entry_address;              // 実行開始アドレス(未実装)
    void*       sub_ram_address;                // 転送先 RAM アドレス
    u32         sub_size;                       // 転送サイズ

    //
    // 0x040 c) ファイルネームテーブル用パラメータ
    //
    ROM_FNT*    fnt_offset;                     // 先頭 ROM アドレス
    u32         fnt_size;                       // テーブルサイズ

    //
    // 0x048 e) ファイルアロケーションテーブル用パラメータ
    //
    ROM_FAT*    fat_offset;                     // 先頭 ROM アドレス
    u32         fat_size;                       // テーブルサイズ

    //
    // 0x0050 d) オーバーレイヘッダテーブル用パラメータ
    //
    //          ARM9
```

```
        ROM_OVT*    main_ovt_offset;            // 先頭 ROM アドレス
        u32         main_ovt_size;              // テーブルサイズ

        //          ARM7
        ROM_OVT*    sub_ovt_offset;             // 先頭 ROM アドレス
        u32         sub_ovt_size;               // テーブルサイズ

        //
        // 0x0060 - 0x006f システム予約域
        //
        // マスク ROM コントロールパラメータ
        u8          rom_param_A[8];             // マスク ROM コントロールパラメータ A
        u8          reserved_B[6];              // システム予約 B
        u8          rom_param_B[2];             // マスク ROM コントロールパラメータ B

        //
        // 0x0070 - 0x0073 ARM9 Autoload 完了時呼び出し関数アドレス
        // 0x0074 - 0x0077 ARM7 Autoload 完了時呼び出し関数アドレス
        //
        //          - AUTOLOAD 完了時にブレークポイントなどの再設定を行なうためのブレークポイント
        //            を仕掛けるためのアドレス
        //          - このアドレスが呼び出されるときに R0 に以下のテーブルへのポインタが格納される。
        //
        //            static void *autoload_params[] =
        //            {
        //              (void*)SDK_AUTOLOAD_LIST,
        //              (void*)SDK_AUTOLOAD_LIST_END,
        //              (void*)SDK_AUTOLOAD_START,
        //              (void*)SDK_STATIC_BSS_START,
        //              (void*)SDK_STATIC_BSS_END,
        //              (void*)0,                   // Compressed Static End
        //              (void*)SDK_VERSION_ID,      // SDK version info
        //              (void*)SDK_NITROCODE_BE,    // Checker 1
        //              (void*)SDK_NITROCODE_LE,    // Checker 2
        //            };
        //
        //          - 2.0PR4 より前の SDK では、この値は 0 となっている。0 の場合はこの機能は
        //            サポートされていないので無視すること。
        //
        void*       main_autoload_done;         // ARM9 フックアドレス
        void*       sub_autoload_done;          // ARM7 フックアドレス

        //
```

```
        // 0x0078 - 0x007f システム予約域
        //
        // マスク ROM コントロールパラメータ
        u8          rom_param_C[8];                 // マスク ROM コントロールパラメータ C

        //
        // 0x0080 - 0x0083 ROM 実効サイズ（ 現状では認証コードアドレスに等しい ）
        // 0x0084 - 0x0087 HEADER サイズ
        //
        union
        {
            u32     rom_valid_size;                 // ROM 実効サイズ
            u32     mb_sign_offset;                 // 認証コード ROM アドレス
        };
        u32         rom_header_size;                // ROM ヘッダサイズ

        //
        // 0x0088 - 0x008b ARM9 モジュールパラメータアドレス
        // 0x008c - 0x008f ARM7 モジュールパラメータアドレス
        //
        //          -領域 0x0070-0x0077 の説明における autoload_params の ROM 内のオフ
        //          セット値
        //          -この値は compstatic によって static 領域を圧縮した後のサイズを
        //          autoload_params[5] の位置に書き込むために使用されます
        //          -現状 ARM9 側しか static 領域の圧縮を行なわないので ARM7 側は 0 となっている
        //
        u32         main_module_param;              // ARM9 モジュールパラメータアドレス
        u32         sub_module_param;               // ARM7 モジュールパラメータアドレス

        //          その他
        u8          reserved_C[0xF0];               // システム予約 C
        u8          reserved_D[4*1024-0x180];       // システム予約 D
        u8          reserved_E[12*1024];            // システム予約 E

} ROM_Header;       // 16KB
```

【特定仕向地対応】

for_korea ： 韓国向け対応フラグ（公開）
　韓国向けNITRO本体へ対応したアプリケーションであることを示します。
　注意点については下記の for_china と同様です。追加した言語コードは(7)です。

for_china : 中国向け対応フラグ（公開）
　中国向けNITRO本体へ対応したアプリケーションであることを示します。
　このフラグがセットされている場合にはオーナー情報の言語コードが中国語になっていることを取得することができます。セットされていない場合には中国語の代わりに英語の言語コードが取得されます。
　これは追加した言語コード(6)を渡すと不具合が発生する既存のアプリケーションが存在することへの対処です。

【諸フラグ】

comp_arm9_boot_area : ARM9常駐ジュール圧縮フラグ
comp_arm7_boot_area : ARM7常駐ジュール圧縮フラグ
　これらのフラグは現在使用されていません。

inspect_card : 検査カードフラグ
　検査カードであることをIPL2が識別するためのフラグです。
　このフラグがセットされているとブートメニューを省略してアプリケーションを起動します。
　量産本体の検査カードやProDGデバッガのROMエミュレータ機能で使用されています。

disable_clear_boot_pad : 非ブート領域クリア・ディセーブルフラグ
　通常はIPL2がメインメモリの非ブート領域をクリアします。しかし、デバッガ動作時に本フラグがセットされている場合は、このメモリクリアを行いません。これによってデバッグ時のアプリケーションの起動時間が若干短くなります。IS-NITRO-DEBUGGERでは「ツール」->「オプション」->「ハードウェアリセット時のメモリ初期化を省略する」にて有効にすることができます。

warning_no_spec_rom_speed : ROMアクセス速度未設定フラグ（公開）
　ビルド時に RSF へ RomSppedType が正常に指定されていない場合にセットされ、デバッガはこのフラグを検出すると警告を行います。マスター提出時には明示的に RomSppedType を指定しなければなりません。

disable_detect_pullout : カード割り込みによる抜け検出禁止フラグ
　カード割り込みを使用したデバイスを搭載している場合、デバイスからの割り込み要求信号によってカード抜けが発生したものと誤認識する可能性があります。この状況を回避するためのフラグです。このようなカードではスクランブル機能を利用し、ROM-IDが以前に読み込んだものと同じ値が返ってくるかどうかを確認することによって抜け検出を行うことができます。

【ROMコントロールパラメータ】

現在、下記の値が設定可能な値となっています。

	［マスクROM］	［ワンタイムROM］	
0x60 :	0x00586000	0x00416657	（ゲームコマンドパラメータ）
0x64 :	0x001808F8	0x081808F8	（セキュアコマンドパラメータ）
0x6E :	0x051E	0x0D7E	（セキュアコマンドソフトウェアレイテンシ）

```
#define ROM_SCRAMBLE_MASK    0x00406000     // スクランブル
#define ROM_LATENCY1_MASK    0x00001FFF     // レイテンシ1（×ROMクロック周期）
#define ROM_LATENCY2_MASK    0x003F0000     // レイテンシ2（×ROMクロック周期）
#define ROM_CLOCK_LEN_MASK   0x08000000     // ROMクロック周期（5 or 8 システムサイクル）

#define ROM_GAME_OP_MASK     0x087F7FFF     // ゲームコマンドパラメータ有効ビット
#define ROM_SECURE_OP_MASK   0x083F1FFF     // セキュアコマンドパラメータ有効ビット
```

スクランブルについて

アプリケーションや IPL2 が使用するゲームコマンドでは 0x00406000 のビットを設定することによって、カードバス上のデータをスクランブルする機能が有効となります。無効にしてもマスクROMからの応答はスクランブルされたままとなるため、実機上では整合性が取れなくなります。逆にデバッガハードウェアにはスクランブル機能が無いため、メインメモリのシステム領域へコピーされたゲームコマンドパラメータの該当ビットはIPL1によってクリアされ、スクランブルが無効な状態でROM領域へアクセスすることになります。このデバッガ上でのスクランブルビットのクリアはクローンブートの認証が通らなくなる原因となるため、c関数でクローンブートの転送イメージにはスクランブルビットが強制設定され、ROM上と同一イメージへ復元されます。

レイテンシについて

通常はROMに対してコマンドを発行すると、レイテンシ1とレイテンシ2の合計期間を待った後、ROM-IDやROM上のデータが返ってきます。しかし、ROMのメモリセルへアクセスしないROM-IDを取得するためにレイテンシ1の最大期間まで待つのは非効率であり、かつカードの抜け検出を行う場合にはARM7が長期間カードバスを占有し続けてしまうという問題もあるため、CARDi_ReadRomIDCore 関数ではレイテンシ2のみを適用してアクセスしています。

【認証コードROMアドレス】
マルチブートにて転送されるプログラムはこのアドレスへ認証コードが格納されることになります。クローンブートもこれに該当します。通常時は単にROMイメージの実効サイズを示しています。

【ROMヘッダサイズ】
ROMヘッダバイナリのサイズを示します。この値はROMヘッダテンプレートのサイズで決まります。
通常は 0x4000 バイトですが、マルチブート子機バイナリでは0x160バイトのROMヘッダテンプレートを使用することによって、ROM容量を少し節約することができます。ただし、この方法で生成したバイナリは通常のアプリケーションとしてカードバスからブートすることはできなくなります。

06/08/18 韓国向け対応フラグの説明が追加されました。
05/04/04 特定仕向地対応/諸フラグ/ROMコントロールパラメータ/認証コードROMアドレス/ROMヘッダサイズの説明が追加されました。

b） 常駐モジュール（MainP/SubP）

特記事項無し

c) ファイル名テーブル

特記事項無し

d) オーバーレイヘッダテーブル

特記事項無し

e) ファイルアロケーションテーブル

特記事項無し

f) ファイルイメージ

特記事項無し

g) マルチブート認証コード

マルチブート認証コードの構造は下記のようになっています。認証サーバー発行ファイル内に含まれる認証コードが attachsign を使用することで ROMイメージへ格納されます。格納場所は cヘッダの mb_sign_offset (rom_valid_size) で示される位置になります。

```
//
// 認証コード
//
typedef struct
{
      u8          id[2];                        // ID = {'a','c'}
      u16         version;                      // 現在のバージョンは 1
      u8          digest[128];                  // ダイジェスト
      u32         serial_no;                    // シリアル番号
} MBSignCode;     // 136B

//
// 認証サーバー発行ファイル
//
typedef struct
{
      MBSignCode  code;                         // 認証コード

      u8          game_name[12];                // ソフトタイトル名
      u8          game_code[4];                 // イニシャルコード
      u8          corp_code[2];                 // メーカーコード
      u8          product_id;                   // 本体コード
```

```
} MBSignFile;      // 155B
```

05/04/04 マルチブート認証コードの項目が追加されました。

2.　NitroROM 作成パス

a) NitroROM 定義ファイル .nsp

特記事項無し

b) NitroROM オブジェクトファイル .nlf

特記事項無し

3 オーバーレイ処理

特記事項無し