

TwISDK

# NAND アプリ開発マニュアル

2009-11-17

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 .....	5
2	用語集 .....	5
3	NANDアプリとSDK .....	6
3.1	NANDアプリ .....	6
3.2	SDK .....	7
4	NANDアプリの製作 .....	8
4.1	プログラムの作成 .....	8
4.2	NAライブラリを含めてリンク .....	8
4.3	RSFの作成 .....	8
4.4	SRLの作成 .....	9
4.5	tadの作成 .....	9
4.6	TWL-SDKのビルドシステムを使用する場合 .....	9
5	インポートと実行 .....	11
5.1	インポートに関する注意 .....	11
5.1.1	イニシャルコード .....	11
5.1.2	バージョン .....	11
5.1.3	セーブデータ領域とサブバナー .....	11
5.2	TwlNmenuを使用する場合 .....	11
5.3	IS-TWL-DEBUGGERを使用する場合 .....	12
6	デバッグ .....	12
6.1	通常のデバッグ .....	12
6.2	簡易的なデバッグ .....	12
7	セーブデータ .....	12
7.1	セーブデータ領域の使用 .....	13
7.2	セーブデータ領域のサイズ .....	13
7.2.1	サイズの指定 .....	13
7.2.2	利用可能なサイズ .....	13
7.2.3	ルートディレクトリに作成可能なファイル数 .....	13
7.2.4	クラスタサイズ .....	14
7.3	SDカードからの書き戻し .....	14
8	バージョン .....	14
8.1	バージョンの形式と指定 .....	14
8.2	バージョンアップ .....	14
8.3	バージョンダウン .....	15

9	SDカードへのコピー .....	15
9.1	コピー .....	15
9.2	コピーの書き戻し .....	15
9.3	書き戻し先の制限 .....	16
10	NANDアプリで利用できる追加の機能.....	16
10.1	サブバナー .....	16
10.2	本体内蔵フォント.....	16
10.3	他のNANDアプリのセーブデータへのアクセス .....	17
10.4	アプリのリセット .....	17
10.5	写真データベース.....	17

## コード

---

コード 4-1	Makefileの例 .....	10
---------	------------------	----

## 改訂履歴

改訂日	改訂内容
2009-11-17	7.1 セーブデータやサブバナーは tad をインポートした際に作成されることを追記
2009-05-22	4.3 サブバナーを使用した場合アプリのサイズが 16KB 増加することを追記 10.5 TCL ライブラリの説明を追加
2009-01-20	9 開発機において NAND アプリのコピー／書き戻しができないことを追加
2008-12-17	6.2 簡易的なデバッグについて注意を追加
2008-11-06	関数リファレンスへのリンクを追加。 DS メニューという語を DSi メニューに修正。 バージョンアップ時の変更禁止プロパティについての記述を追加。
2008-10-23	11 章を削除。
2008-09-16	初版。

# 1 はじめに

このドキュメントは NAND アプリの開発に必要な技術情報を提供します。

このドキュメントではカードアプリとの比較を中心に解説を行うため、カードアプリを作成するのに必要な知識を持っている方を対象としています。

## 2 用語集

本ドキュメントで使用される用語を簡単に解説します。これらの用法は一般的な意味とは異なる場合があります。

- 本体保存メモリ  
TWL 本体に内蔵されている NAND フラッシュメモリ。
- カードアプリ  
ゲームカードに格納され、実行される TWL アプリケーション。実行時にはゲームカードからデータを読み込む。
- NAND アプリ  
本体保存メモリに格納され、実行される TWL アプリケーション。実行時には本体保存メモリからデータを読み込む。
- SRL  
TWL アプリケーションの実行ファイル形式。またそれを格納したファイル、およびそのファイルにつけられる拡張子。
- ROM  
SRL と同義。ただし SRL よりデータとしての面が強調される。
- RSF (Rom Spec File)  
SRL を作成する時に使用する各種パラメータを記述したファイル。
- リマスターバージョン  
RSF に記述され SRL 内部に格納される SRL のバージョン番号。
- 会社コード  
パブリッシャ毎に割り当てられる ASCII 2 文字からなる識別子。
- イニシャルコード  
各 TWL アプリケーションに割り当てられる一意の識別子。ASCII 4 文字からなる。
- インポート  
NAND アプリを本体保存メモリに格納し、起動可能な状態にすること。
- tad  
NAND アプリを本体保存メモリにインポートする時に使用するファイル形式。内部に SRL を含む。
- DSi メニュー  
TWL を起動したときに表示される画面。起動するアプリケーションを選択するために用いられる。
- アーカイブ  
TWL-SDK の FS ライブラリが操作の対象とする個々のファイルシステム。Windows PC でいう「ドライブ」に相当する。

- ROM アーカイブ  
TWL-SDK に標準で用意されている ROM 内ファイルシステムを表すアーカイブ。デフォルトで FS ライブラリにロードされる。
- バナー  
DSi メニューでアプリケーションの一覧を表示する時に使われる、アプリケーションを識別するためのアイコンやアプリケーション名などをまとめたデータ。
- サブバナー  
バナーのサブセットであり、バナーに含まれる画像データを置き換える目的で使用されるデータ。またはそれを使用したバナー画像置き換えの仕組み。
- セーブデータ  
アプリケーションがアプリケーション終了後も継続して保持するデータ。
- セーブデータ領域  
NAND アプリのセーブデータを格納するために本体保存メモリに確保される領域。
- public セーブデータ  
セーブデータ領域の一つで、ユーザが SD カードへコピーすることができる領域。
- private セーブデータ  
セーブデータ領域の一つで、ユーザが SD カードへコピーすることができない領域。
- リファレンス  
TWL-SDK の man ディレクトリ以下にある API およびツールの説明文書のこと。

## 3 NANDアプリとSDK

### 3.1 NANDアプリ

NAND アプリは TWL 本体内蔵の本体保存メモリにインストールすることができるアプリケーションのことです。

カードアプリと比較して技術的な観点から主に以下のような違いがあります。

- 本体保存メモリから起動  
NAND アプリの起動には DS カードは不要です。代わりに本体保存メモリにインポートする必要があります。
- ROM のアクセス速度  
ROM が本体保存メモリ内に存在するため FS ライブラリでの ROM アーカイブに対するアクセス速度が異なります。またアクセス頻度に制限があります。
- セーブデータの保存場所  
本体保存メモリに保存されます。詳しくは「7 セーブデータ」を参照してください。
- SD カードへのコピー  
ユーザがSDカードへアプリとセーブデータをコピーすることができます。詳しくは「9 SDカードへのコピー」を参照してください。

## 3.2 SDK

SDK のパッケージに含まれるものでカードアプリには不要で NAND アプリには必要なものは以下になります。

### ライブラリ

- `include/twl/na.h`  
NA ライブラリのヘッダです。NA ライブラリには NAND アプリで使える追加の API が含まれています。また SDK 内部で使用する NAND アプリに必須の関数も含まれているため NAND アプリは必ず NA ライブラリをリンクする必要があります。
- `include/twl/specfiles/ROM-TS.nand.rsf`  
NAND アプリで使える追加の項目を含んだ RSF ファイルのサンプルです。
- `lib/ARM9-TS/*/libna.TWL*.a`  
NA ライブラリの .a ファイルです。

### ツール

- `tools/bin/maketad.exe`  
SRL から tad に変換する Windows ツールです。NAND アプリを本体保存メモリにインポートするには tad に変換する必要があります。
- `bin/ARM9-TS/Rom/TwlNmenu.srl`  
SD カード上の tad ファイルから TWL 開発機へ NAND アプリをインポートするための TWL ツールです。
- `bin/ARM9-TS/Rom/NandFiler.srl`  
NAND アプリのセーブデータ領域内を参照することができる TWL ツールです。
- `bin/ARM9-TS/Rom/TwlNmenu.tad`, `bin/ARM9-TS/Rom/NandFiler.tad`  
TwlNmenu および NandFiler を NAND アプリとして作成したものです。

### ドキュメント

- `docs/README/NandAppManual.pdf`  
このドキュメントです。
- `man/ja_JP/na/list_na.html`  
NA ライブラリのリファレンスです。
- `man/ja_JP/demos/nandApp/nandAppdemos.html`  
NAND アプリサンプルデモのリファレンスです。
- `man/ja_JP/romfiles/TwlNmenu.html`  
TwlNmenu のリファレンスです。
- `man/ja_JP/romfiles/NandFiler.html`  
NandFiler のリファレンスです。
- `man/ja_JP/tools/maketad.html`  
maketad のリファレンスです。
- `man/ja_JP/tools/SaveDataSize.html`  
セーブデータ領域のサイズとして使用できる値の一覧です。

### サンプルデモ

- build/demos.TWL/nandApp

NAND アプリのサンプルデモです。

## 4 NANDアプリの製作

NAND アプリを TWL 上で実行するためには **tad** を作成する必要があります。**tad** を作成するまでの大まかな流れは以下のようになります。

1. カードアプリと同様にプログラムを作成する
2. NA ライブラリを含めてリンクする
3. アプリ固有の RSF を作成する
4. アプリ固有の RSF を用いて SRL を作成する
5. SRL から tad を作成する

各ステップについて以下で説明します。

### 4.1 プログラムの作成

基本的にはカードアプリと同様に作成できます。ただし、以下の点が異なるため注意する必要があります。

- FS ライブラリでの ROM アーカイブに対するアクセス速度が異なり、アクセス頻度に制限がある。
- セーブデータは **dataPub**、**dataPrv** アーカイブに保存する。「7 セーブデータ」を参照してください。
- セーブデータはそのままにアプリケーション本体がアップデートされる場合がある。「8 バージョン」を参照してください。
- ユーザにより **dataPrv** アーカイブがクリアされる可能性がある。「9 SDカードへのコピー」を参照してください。

### 4.2 NAライブラリを含めてリンク

NAND アプリではプログラムのリンク時に NA ライブラリ (**libna.TWL\*.a**) をリンクする必要があります。NA ライブラリに含まれる API の使用は任意ですが、使用しない場合であっても NAND アプリの動作に必要な関数が NA ライブラリに含まれているため、必ず NA ライブラリをリンクする必要があります。

### 4.3 RSFの作成

SRL を作成する時には RSF (Rom Spec File) を用意する必要があります。TWL-SDK では NAND アプリに関するパラメータを指定する **AppendProperty** セクションが RSF に追加されています。NAND アプリを作成する場合には **AppendProperty** セクションに適切なパラメータを記述した RSF を使用して SRL を作成する必要があります。適切なパラメータが設定されていない場合、**tad** の作成時に **maketad.exe** によりエラーが報告されます。各 NAND アプリに応じた適切な RSF を作成してください。なお、**include/twl/specfiles/ROM-TS\_nand.rsf** に NAND アプリで設定可能な項目を含んだ RSF のサンプルがあります。

**AppendProperty** セクションの各設定項目について以下で簡単に説明します。詳しい説明や **AppendProperty** セクション以外の RSF の仕様については **makerom.TWL** のリファレンスを参照してください。



- Media

ROM の格納メディアを指定します。デフォルトは「GameCard」です。NAND アプリでは「NAND」を指定しなければなりません。

- InitialCode

イニシャルコードを指定します。アプリケーション毎に割り当てられたイニシャルコードを ASCII 4 文字で指定してください。本体保存メモリにインポートされた NAND アプリはイニシャルコードで区別されるため、イニシャルコードが同じ NAND アプリが既にインポートされている場合のインポートは上書きインポートとなることに注意してください。

- PublicSaveDataSize

publicセーブデータ領域のサイズを指定します。このセーブデータ領域はSDカードにコピーできる領域となります。セーブデータ領域についての詳細は「7 セーブデータ」を参照してください。

指定できる値については関数リファレンスの[「ツール」 → 「ROMイメージ関連」 → 「セーブデータサイズ一覧」](#)を参照してください。

- PrivateSaveDataSize

privateセーブデータ領域のサイズを指定します。このセーブデータ領域はSDカードにコピーできない領域となります。セーブデータ領域についての詳細は「7 セーブデータ」を参照してください。

指定できる値は PublicSaveDataSize と同様です。

- SubBannerFile

サブバナーを使用するかどうかを指定します。デフォルトは「FALSE」です。サブバナーを使用する場合は「TRUE」を、使用しない場合は「FALSE」を設定してください。サブバナーについては「10.1 サブバナー」を参照してください。

サブバナーを使用した場合は、NANDアプリ全体のサイズが 16KB増加します。NANDアプリ全体のサイズの計算方法の詳細は関数リファレンスの[「ビルド済みプログラム」 → 「NANDアプリ関連」 → 「TwlNmenu.srl」](#)を参照してください。

## 4.4 SRLの作成

SRLの作成はカードアプリと同様です。ただしSRLの作成に使用するRSFは「4.3 RSFの作成」で述べている適切なRSFを使用しなければならないことに注意してください。

## 4.5 tadの作成

SRLからtadを作成するにはSRLファイルのパスをコマンドライン引数としてmaketad.exeを実行してください。デフォルトではRSFのTitleNameプロパティの値に拡張子.tadを付加したファイル名で出力されますが、-oオプションで出力ファイル名を指定することもできます。詳しくは[maketadのリファレンス](#)を参照してください。

例:

```
maketad.exe MyNandApp.srl -o MyNandApp.tad
```

## 4.6 TWL-SDKのビルドシステムを使用する場合

TWL-SDK の make ビルドシステムには SRL 作成のためのルールがあらかじめ用意されています。また NAND アプリを作成するためのルールも用意されており Makefile 中で TWL\_NANDAPP 変数に TRUE を指定することで、こ

のルールを有効にできます。この NAND アプリを作成するためのルールには `tad` を作成するルールと NA ライブラリをリンク対象に加える設定が含まれるため、これらを使用することで簡単に `tad` を作成することができます。

Twl-SDK のビルドシステムを使用して NAND アプリの作成を行う場合、カードアプリ用 Makefile の下記 3 箇所を変更します。

- `TWL_NANDAPP` に `TRUE` を指定します。
- `ROM_SPEC` に NAND アプリ用の `RSF` へのパスを指定します。
- `TARGET_BIN` に `main.tad` のような拡張子が `.tad` となったファイル名を指定します。

これで `make` コマンドを実行することで `tad` の作成まで行われます。Makefile の具体例は NAND アプリのサンプルデモを参照してください。

#### コード 4-1      Makefile の例

```
～（略）～
TARGET_PLATFORM := TWL
TWL_ARCHGEN     := LIMITED
TWL_NANDAPP     =  TRUE

SRCS            =  main.c
TARGET_BIN      =  MyNandApp.tad

ROM_SPEC        =  MyNandApp.rsf

include $(TWLSDK_ROOT)/build/buildtools/commondefs

do-build:      $(TARGETS)

include $(TWLSDK_ROOT)/build/buildtools/modulerrules
～（略）～
```

## 5 インポートと実行

NAND アプリを実行するためには NAND アプリを TWL の本体保存メモリにインポートする必要があります。NAND アプリのインポートには TwlNmenu もしくは IS-TWL-DEBUGGER を使用します。

### 5.1 インポートに関する注意

#### 5.1.1 イニシャルコード

本体保存メモリにインポートされた NAND アプリはイニシャルコードで識別されます。そのため NAND アプリのインポート時にイニシャルコードが同じ NAND アプリがインポートされている場合は上書きインポートとなります。

#### 5.1.2 バージョン

NAND アプリにはそれぞれにバージョンがあります。上書きインポートを行う場合は、インポートする側のバージョンが既にインポートされている側のバージョンと同じか、より大きなバージョンでなければなりません。バージョンダウンとなる場合はインポートに失敗します。

バージョンダウンを行う必要がある場合はインポートされている NAND アプリを削除してからインポートしなおしてください。NAND アプリの削除は TwlNmenu で行うことができます。

NAND アプリのバージョンについては「8 バージョン」を参照してください。

#### 5.1.3 セーブデータ領域とサブパナー

上書きインポートが行われる場合、通常はセーブデータ及びサブパナーの内容は維持されます。ただし、インポートする側と既にインポートされている側とでセーブデータ領域のサイズ及びサブパナーの使用/不使用の設定がどれか一つでも異なる場合、全てのセーブデータ領域とサブパナーの内容がクリアされます。

### 5.2 TwlNmenuを使用する場合

tad をインポートして実行するまでの手順は以下のようになります。

1. tad ファイルを SD カードに保存し TWL 開発機の SD カードスロットにセットします。
2. TwlSDK/bin/ARM9-TS/Rom/TwlNmenu.srl を開発機で実行します。
3. 開発機上の TwlNmenu を操作して tad をインポートします。
  - (i) 開発機の十字ボタン左右で SD モードに切り替えます。
  - (ii) 対象の tad ファイルを選択し A ボタンを押します。十字ボタンの上下でディレクトリ内のファイル/ディレクトリの選択の変更が、ディレクトリを選択している状態で A ボタンを押すと選択されているディレクトリへ移動ができます。
  - (iii) インポートの確認画面が表示されますので A ボタンを押します。
  - (iv) インポートが終了するまで待ちます。

4. 開発機上の TwlNmenu を操作して NAND アプリを実行します。
  - (i) 開発機の十字ボタン左右で NAND モードに切り替えます。
  - (ii) NAND モードではインストールされている NAND アプリのイニシャルコード一覧が表示されますので十字ボタン上下で起動したい NAND アプリを選択し **start** ボタンを押します。
  - (iii) NAND アプリが起動します。

### 5.3 IS-TWL-DEBUGGERを使用する場合

IS-TWL-DEBUGGER ソフトウェアのウインドウに **tad** をドラッグアンドドロップするか、「ファイル」メニューの「開く」から **tad** を指定して「OK」ボタンを押すと **tad** のインポートが行われます。その後カードアプリと同様に NAND アプリをデバッグ実行できます。

## 6 デバッグ

デバッグには IS-TWL-DEBUGGER を使用します。

### 6.1 通常のデバッグ

通常のデバッグは「5.3 IS-TWL-DEBUGGERを使用する場合」の方法で行います。この方法でのデバッグを強く推奨します。

### 6.2 簡易的なデバッグ

デバッグ対象の NAND アプリを一度インポートした後は SRL または TLF を使用した簡易的なデバッグを行うことができます。これはカードアプリと同様に SRL または TLF を IS-TWL-DEBUGGER ソフトウェアから開くことで行います。

この方法では通常のデバッグと異なり以下の注意点があります。

- ROM アーカイブが本体保存メモリへのアクセスではなく DS カード相当へのアクセスになります。  
この場合カードアプリとして起動されたものと同じ状態になり、**NAND アプリをインポートして実行した場合と比較して ROM アーカイブへのアクセス速度が半減します。一方コンポーネントの負荷が軽くなるため、無線やマイク、サウンドの動作が異なる可能性があります。**
- セーブデータやサブパナーは **tad** をインポートした際に作成されますので、必ず一度 NAND アプリをインポートしてください。またセーブデータ領域のサイズやサブパナーの使用/不使用の設定を変更した場合には再度インポートを行う必要があります。

**開発の最終段階では、必ず「6.1 通常のデバッグ」の方法で NAND アプリをインポートして動作確認をしてください。**

## 7 セーブデータ

NAND アプリではカードアプリと異なりカード上のバックアップデバイスを使用することができません。そのため、この代

わりとして NAND アプリごとに本体保存メモリ上にセーブデータ領域が確保されます。

## 7.1 セーブデータ領域の使用

NAND アプリのセーブデータ領域は独立したファイルシステムとなっており、FS ライブラリのアーカイブの一つとしてロードされます。NAND アプリで `FS_Init` 関数を呼び出すと内部で「dataPub」と「dataPrv」の 2 つのアーカイブがロードされます。これらのアーカイブ名を付加したパスで FS ライブラリを使用することでセーブデータ領域にファイル/ディレクトリを作成したりファイルの読み書きを行ったりすることができます。

「dataPub」アーカイブは後述する NAND アプリの SD カードへのコピー時に一緒に SD カードへコピーされるセーブデータ領域です。コピーされても問題ない通常のセーブデータはこの領域に保存します。この領域のことを「public セーブデータ」と呼びます。

「dataPrv」アーカイブはコピーされることがないセーブデータ領域です。コピーされては困るセーブデータはこの領域に保存します。この領域のことを「private セーブデータ」と呼びます。

セーブデータやサブバナーは `tad` をインポートした際に作成されますので、必ず一度 NAND アプリをインポートしてください。またセーブデータ領域のサイズやサブバナーの使用/不使用の設定を変更した場合には再度インポートを行う必要があります。インポートせずに `SRL` または `TLF` を実行した場合には、これらのアーカイブのファイル操作に失敗します。

## 7.2 セーブデータ領域のサイズ

### 7.2.1 サイズの指定

セーブデータ領域のサイズはNANDアプリごとに選択し、RSFで指定する必要があります。セーブデータ領域サイズの指定については「4.3 RSFの作成」を参照してください。

デフォルトではセーブデータ領域のサイズは 0 に設定されています。セーブデータ領域のサイズが 0 の場合、それぞれに対応するアーカイブはロードされません。

### 7.2.2 利用可能なサイズ

セーブデータ領域内にはファイルシステムの管理領域などが含まれるため、アプリケーションで実際に使用できる領域は指定する値より小さな値になります。セーブデータ領域サイズごとの実際に利用可能なサイズの値については関数リファレンスの「[ツール](#)」→「[ROMイメージ関連](#)」→「[セーブデータサイズ一覧](#)」を参照してください。

セーブデータ領域内では利用可能なサイズを超えない限り自由にファイル/ディレクトリを作成することができます。サイズを超えたファイル/ディレクトリの作成は FS API がエラーを返します。

利用可能な最大サイズおよび現在の残り容量は[FS\\_GetArchiveResource](#)関数で取得することができます。

### 7.2.3 ルートディレクトリに作成可能なファイル数

セーブデータ領域で作成できるファイル数に制限はありません。ただしセーブデータ領域のルートディレクトリに限っては作成できるファイルとディレクトリの数に上限があります。ルートディレクトリに作成できるファイル/ディレクトリ数はそのファイル名/ディレクトリ名の長さにより変化します。

ルートディレクトリで作成可能なファイル数の上限はセーブデータ領域サイズによって異なります。値の一覧については関数リファレンスの[「ツール」](#) → [「ROMイメージ関連」](#) → [「セーブデータサイズ一覧」](#)を参照してください。

### 7.2.4 クラスタサイズ

セーブデータ領域でファイルを作成する場合、N byte のファイルが消費する領域は一般に N byte ではなく N を特定の値の倍数に切り上げた値となります。この特定の値をクラスタサイズと呼びます。たとえばクラスタサイズが 512 byte の場合には 1 byte のファイルも 512 byte のファイルも 512 byte の領域を消費します。同様に 513 byte でも 1024 byte でも 1024 byte の領域を消費します。

クラスタサイズはセーブデータ領域サイズによって異なります。値の一覧については関数リファレンスの[「ツール」](#) → [「ROMイメージ関連」](#) → [「セーブデータサイズ一覧」](#)を参照してください。

## 7.3 SDカードからの書き戻し

SD カードへのコピーでは private セーブデータは含まれないため、SD カードにコピーされた NAND アプリを書き戻すときには private セーブデータはクリアされます。このため、アプリケーションは常に private セーブデータのみがクリアされる状況を想定する必要があります。

SDカードへのコピーに関する詳細は「9 SDカードへのコピー」を参照してください。

# 8 バージョン

NAND アプリにはバージョンがあります。

## 8.1 バージョンの形式と指定

NAND アプリのバージョンは 0～255 までのメジャーバージョンと、同じく 0～255 までのマイナーバージョンをピリオドで連結した形で表現されます。例えば、メジャーバージョンが 2 でマイナーバージョンが 1 の場合であればバージョンは「2.1」と表現されます。

NAND アプリのメジャーバージョンは SRL のリマスターバージョンがそのまま使用されます。SRL のリマスターバージョンが 1 であれば、NAND アプリのメジャーバージョンも 1 となります。NAND アプリのリリース後、SRL のバージョンアップを行うごとに 1 ずつ増加していくことになります。

NAND アプリのマイナーバージョンは予約されており、現在は常に 0 です。

## 8.2 バージョンアップ

カードアプリと異なり NAND アプリはユーザの手元で遊ばれているソフトウェアをバージョンアップすることができます。バージョンアップが行われる場合はセーブデータが維持されるため、新しいバージョンの NAND アプリは過去の全てのバージョンのセーブデータを適切に扱えるよう実装する必要があります。セーブデータのフォーマットを恒久的に変更しないか、あるいは事前にバージョン番号を埋め込んでおきプログラムで変更に対応できるようにしてください。

なお、バージョンアップ前後で RSF の下記のプロパティを変更することを禁止します。

- TitleName

- MakerCode
- CardRegion
- InitialCode
- PublicSaveDataSize
- PrivateSaveDataSize
- SubBannerFile

## 8.3 バージョンダウン

ユーザはあらかじめ古いバージョンの NAND アプリを SD カードにコピーしておくことで、いつでも古いバージョンの NAND アプリを書き戻すことができます。

NAND アプリを SD カードへコピーする時にはセーブデータも一緒にコピーされ、本体保存メモリに書き戻す場合にもセーブデータを一緒に書き戻します。そのため、NAND アプリ本体は古いのにセーブデータは新しいという状況は発生しません。ただし、SD カードから本体保存メモリに書き戻すときに **public** セーブデータは SD カードから書き戻されますが、**private** セーブデータはクリアされるという点に注意する必要があります。

# 9 SDカードへのコピー

TWL システムの本体設定内にある「ソフト管理」画面では、ユーザに NAND アプリを SD カードにコピーする機能が提供されます。ここでは SD カードへのコピーの仕様を情報として掲載します。

但し **SystemUpdater** でアップデートされた各種開発機向けシステムメニューでは、「ソフト管理」画面において以下の NAND アプリのコピー／書き戻しはできませんので、ご注意ください。

## 9.1 コピー

NAND アプリを SD カードにコピーする際には NAND アプリ本体と同時に **public** セーブデータもコピーされます。一方、**private** セーブデータはコピーされません。NAND アプリ本体と **public** セーブデータは単一のファイルにまとめられて SD カードにコピーされます。

SD カードにコピーされたデータには暗号化と改竄防止処理が施されます。

## 9.2 コピーの書き戻し

SD カード上の NAND アプリのコピーを TWL 本体に書き戻す場合には NAND アプリ本体と共に **public** セーブデータも書き戻されます。TWL 本体側にある **public** セーブデータは常に SD カード側のもので上書きされます。これは、場合によってはユーザによるセーブデータの巻き戻しが可能なことを意味します。

セーブデータの巻き戻しを防ぎたい場合は、**private** セーブデータを用いることができます。しかしながら、**private** セーブデータは SD カードにはコピーされないため、ユーザが本体保存メモリへの書き戻しを行った場合に **private** セーブデータが空の状態となります。つまり、アプリケーションは常に **private** セーブデータのみがクリアされている状態で起動される可能性があることを想定する必要があります。**public** セーブデータは維持されているが、**private** セーブデータのみがクリアされているような状況で異常な動作をしないようにしなければなりません。

これらの特性を踏まえ、アプリが保存すべき情報のうちどれを **public** セーブデータに配置して、どれを **private** セーブデータに配置するかを決定してください。上述の通り、**private** セーブデータは簡単に失われる可能性があるため、必要以上に **private** セーブデータに情報を詰め込み過ぎないように注意してください。

### 9.3 書き戻し先の制限

TWL では、SD カードからのコピーの書き戻しでは元のコピーを作成した TWL 本体にのみ書き戻せるように制限されます。コピーを作成した TWL 本体とは異なる本体には書き戻しを行うことはできません。よってセーブデータを他の本体にコピーすることはできません。

この制限はシステムによって実装されていますので、アプリケーションの側で本体を特定するような情報（MAC アドレスなど）をセーブデータに埋め込んで本体が一致しているかどうかの判定を行う必要はありません。また、本体修理時にモジュール交換をした場合などに問題となるため、そのような実装は禁止します。

## 10 NANDアプリで使用できる追加の機能

NAND アプリではカードアプリで使用できないいくつかの追加の機能を使用することができます。ここではその機能を簡単に紹介します。

### 10.1 サブバナー

NAND アプリもカードアプリと同様にバナーを持ちます。本体保存メモリにインストールされた NAND アプリは DSi メニュー上でバナーが列挙され、ユーザはこれを選択することで NAND アプリを起動することになります。

カードアプリでは SRL の作成時に指定したバナーが常に使用され、バナー画像を途中で変更することができませんでした。NAND アプリではサブバナーと呼ばれる代替のバナーが使用でき、この仕組みによりバナー画像を変更することができます。

サブバナーは NAND アプリから書き換えることができるため、たとえばゲームの進行に応じてバナー画像が変化するということを実現できます。

詳細については関数リファレンスの [「NA」 → 「概要」 → 「サブバナー」](#) を参照してください。サンプルデモ SubBanner がサブバナーのデモです。

### 10.2 本体内蔵フォント

TWL の本体保存メモリには大中小 3 種類のフォントが内蔵されています。NAND アプリは任意でこの内蔵フォントを使用することができます。

本体内蔵フォントは TWL-System の NFTR 形式で格納されており下記文字セットをサポートしています。

- ASCII
- ISO 8859-1
- ISO 8859-7
- CP 932
- CP 1252
- CP 1253



- JIS X0201
- JIS X0208
- DS 外字
- Wii 外字

詳細については関数リファレンスの[「NA」→「概要」→「内蔵フォント」](#)を参照してください。サンプルデモ `sharedFont` が本体内蔵フォントを読み込むデモです。

## 10.3 他のNANDアプリのセーブデータへのアクセス

---

NAND アプリでは会社コードが同一の他の NAND アプリのセーブデータを読み書きすることができます。他の NAND アプリのセーブデータ領域を FS ライブラリのアーカイブとしてロードすることができ、これにより FS ライブラリで操作できるようになります。

詳細については関数リファレンスの[「NA」→「概要」→「アーカイブ」](#)を参照してください。サンプルデモ `other_backup` が他の NAND アプリのセーブデータ (backup デモのセーブデータ) を読み書きするデモです。

## 10.4 アプリのリセット

---

NAND アプリでは技術的な制約により `OS_ResetSystem` 関数によるリセットが利用できません。この代わりとして `OS_RebootSystem` 関数が用意されています。また、`OS_JumpToSystemMenu` 関数を使用することでアプリケーションの処理を終了して DSi メニューへ戻ることができます。

詳細については関数リファレンスの[「OS」→「概要」→「リセット」](#)を参照してください。

## 10.5 写真データベース

---

TWL には写真データベースと呼ばれる NAND アプリ間で JPEG 画像を共有する仕組みが用意されています。共有された JPEG 画像にアクセスするためには DSi 写真データベースライブラリ (TCL) ライブラリを使用します。TCL ライブラリを使うと、DSi カメラ等によって NAND 上に保存される JPEG 画像の読み込みや、DSi カメラが読み込めるパスへの JPEG 画像の書き出しを行うことができます。

詳細については関数リファレンスの[「TCL」→「概要」→「TCLについて」](#)を参照してください。

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。

© 2009 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。