

E-Manual Display (M2d) Library Programmer's Guide

2010/04/13

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	5
2	Development Environment	6
2.1	Required Development Environment.....	6
2.2	File Organization.....	6
2.2.1	Library File.....	6
2.2.2	Header File	6
2.2.3	E-Manual Binary Files	6
2.2.4	Resource Files for the Library	6
3	Incorporating the Library.....	7
3.1	OS Initialization	7
3.2	Preprocessing Required Before Displaying E-Manuals.....	7
3.2.1	Touch Panel.....	7
3.2.2	Sounds	8
3.3	Initializing the M2d Library	8
3.4	Displaying the E-Manual	10
3.5	Closing the Viewer	10
3.6	Recovery Processing Required After Closing the E-Manual Display	10
3.7	Multiple Language Support.....	11
3.7.1	Determining the Language to Be Displayed.....	11
3.7.2	Explicitly Specifying the Language.....	11
4	Notes.....	13
4.1	Archive Identifiers	13
4.2	IRQ Interrupt Handler.....	13
4.3	Automatic Power OFF.....	13

Code

Code 3-1 OS Library Initialization	7
Code 3-2 Preprocessing Before Displaying E-Manuals	7
Code 3-3 Initializing the M2d Library Using File Paths.....	8
Code 3-4 Initializing the M2d Library Using Data in Memory	9
Code 3-5 Displaying the E-Manual	10
Code 3-6 Closing the Viewer	10
Code 3-7 Recovery Processing After E-Manual Display	10
Code 3-8 Explicitly Specifying a Language	11

Revision History

Revision Date	Description
2010/04/13	Revised information on sound volumes in section 3.2.2 Sounds.
2009/09/07	Revised section 3.2.2 Sounds.
2009/08/06	Revised section 2.1 Required Development Environment.
2009/05/20	Added a description of the number of channels required for sounds.
2009/02/23	Changed the version numbers of TWL-SDK and TWL-System.
2008/12/22	Added a description of how to determine the language to use when the e-manual binary file includes manuals in multiple languages. Added precautions (about archive identifiers, the interrupt handler, and automatic power OFF). Deleted a list of markets from section 2.2.4 Resource Files for the Library.
2008/11/19	Revised descriptions due to changes in the initialization function's arguments (font array).
2008/11/06	Added a description of required preprocessing.
2008/10/29	Added a description of required recovery processing and cautions.
2008/10/15	Added a description.
2008/10/06	Initial version.

1 Introduction

The M2d library is used to display e-manuals. This document describes the requirements for incorporating the M2d library into applications.

For a detailed explanation of each function, see the function reference.

2 Development Environment

2.1 Required Development Environment

The M2d library requires the following development environment. See “Software Requirements” in `docs/ReleaseNotes.html` for details on the versions.

- TWL-SDK
- TWL-System

2.2 File Organization

2.2.1 Library File

The library files (`libntmvm2d.*a`) are in the `lib/ARM9-TS` directory.

2.2.2 Header File

Include the header files for the required functions with the following directive.

```
#include <ntmv/m2d.h>
```

2.2.3 E-Manual Binary Files

Binary files for e-manuals are generated by saving a ManualEditor project as a BLZ file. For more information, see the ManualEditor documentation.

2.2.4 Resource Files for the Library

The M2d library requires a resource file to display the user interface. This file (`m2dres_narc.blz`) is located in the `data` directory.

3 Incorporating the Library

This section describes the procedure for incorporating the M2d library into an application.

3.1 OS Initialization

Initialize the OS library. Call the `FS_Init` function to use file functions with the M2d library.

Code 3-1 OS Library Initialization

```
void
NitroMain
{
    OS_Init();

    // Enable V-Blank interrupts
    OS_SetIrqFunction(OS_IE_V_BLANK, VBlankIntr);
    OS_EnableIrqMask(OS_IE_V_BLANK);
    GX_VBlankIntr(TRUE);

    // Permission for FIFO interrupt for communication with ARM7
    OS_EnableIrqMask(OS_IE_SPFIFO_RECV);

    OS_EnableIrq();

    FS_Init(FS_DMA_NOT_USE); // File system initialization
    ...
}
```

3.2 Preprocessing Required Before Displaying E-Manuals

3.2.1 Touch Panel

The M2d library uses a touch panel autosampling feature. If your application also uses touch panel autosampling, stop that feature before using the M2d library.

Code 3-2 Preprocessing Before Displaying E-Manuals

```
// Stop touch panel autosampling
TP_RequestAutoSamplingStop();
```

3.2.2 Sounds

The M2d library uses TWL-System NITRO Composer (Snd) internally. Be sure the caller stops all sound playback and capture before displaying the e-manual. The master volume, player 0, and player 1 volumes must be set to maximum (default). These volumes are not changed by the M2d library.

Also, if you allocate channels using the following functions, sounds in the e-manual may not play back in some cases.

Deallocate all sound channels before displaying the e-manual. The e-manual requires four channels. The exact channel numbers that need to be opened are not specified.

- SND_LockChannel
- NNS_SndLockChannel
- NNS_SndArcStrmAllocChannel
- NNS_SndStrmAllocChannel
- NNS_SndWaveOutAllocChannel

3.3 Initializing the M2d Library

Initialize the M2d library by passing `NNSFndAllocator` from TWL-System, the binary data for the e-manual, the resource data for M2d, and a pointer array to `NNSG2dFont` structures (from TWL-System) to the initialization function. There are two methods for specifying e-manual binary data and M2d resource data. The first method is to specify the path to the ROM file as a string; the second is to specify the address of the data extracted to memory in advance. The initialization functions are `NTMV_M2dInitForFilePath` and `NTMV_M2dInitForMem`, respectively. The following example describes the use of the `NTMV_M2dInitForFilePath` function to specify the file path.

Code 3-3 Initializing the M2d Library Using File Paths

```
NNSFndAllocator    gAllocator;          // Allocate memory

. . .

// Pointer array to NNSG2dFont structures configured with the console's internal
// font data used to display the e-manual
NNSG2dFont* fonts[3];
...

// Structure that stores the information used by the M2d library
NTMVM2dManualViewerInfo manViewInfo;

// Initialization
NTMV_M2dInitForFilePath(
    &manViewInfo,
```



```
&gAllocator,  
"/data/manpages_narc.blz",    // E-manual binary files  
"/data/m2dres_narc.blz",      // Resource files  
fonts);
```

The following example describes how to use the `NTMV_M2dInitForMem` function to specify the data extracted to memory. The e-manual binary data and M2d library resource file are compressed with the TWL-SDK `compBLZ` tool. The data in memory that will be passed to `NTMV_M2dInitForMem` must first be extracted with the `MI_SecureUncompressBLZ` function in the TWL-SDK. The M2d library resource file must be 32-byte-aligned.

Code 3-4 Initializing the M2d Library Using Data in Memory

```
// Initialization  
  
FSFile manFile;  
FSFile resFile;  
  
// Loading e-manual binary files  
FS_InitFile(&manFile);  
FS_OpenFile(&manFile, "/data/manpages_narc.blz");  
...  
//      Extracting data compressed with compBLZ  
MI_SecureUncompressBLZ(manData, manCompBytes, manExtractBytes);  
...  
// Load M2d library resource files  
FS_InitFile(&resFile);  
FS_OpenFile(&resFile, "/data/m2dres_narc.blz");  
...  
//      Extracting data compressed with compBLZ.  
//      The memory indicated by resData must use 32-byte alignment.  
MI_SecureUncompressBLZ(resData, resCompBytes, resExtractBytes);  
...  
  
NTMV_M2dInitForMem(  
    &manViewInfo,  
    &gAllocator,  
    manData, // E-manual binary data  
    resData, // Resource data  
    fonts);
```

3.4 Displaying the E-Manual

To display the e-manual, call `NTMV_M2dRun`. Once this function is called, control does not return until the user presses the **Close Manual** button to stop viewing the e-manual.

Code 3-5 Displaying the E-Manual

```
// Runs the manual viewer
NTMV_M2dRun(&manViewInfo);
```

3.5 Closing the Viewer

The following example closes the M2d library.

Code 3-6 Closing the Viewer

```
// Closes the e-manual viewer
NTMV_M2dFinalize(&manViewInfo, &gAllocator);
```

3.6 Recovery Processing Required After Closing the E-Manual Display

After the e-manual viewer is closed, the M2d library calls the following TWL-SDK functions and changes the settings. This enables reconfiguration of the display.

- Graphics
- Touch panel

Note: The M2d library uses the TWL-System NITRO composer (Snd) for sound. When NITRO composer (Snd) is used, reconfigure Snd by calling the `NNS_SndArcSetCurrent` and `NNS_SndArcPlayerSetup` functions.

Code 3-7 Recovery Processing After E-Manual Display

```
// Reconfiguring graphics hardware
GX_Init();
...
// Reconfiguring the touch panel
TP_RequestAutoSamplingStart(vcount, frequency, samplingBufs, bufSize);
. . .
// Reconfiguring sound
NNS_SndArcSetCurrent(gameSoundArc);
NNS_SndArcPlayerSetup(gameSoundHeap);
```

3.7 Multiple Language Support

Multiple languages can be included in the e-manual binary file. This section describes how the language used to display the e-manual is determined. It also describes how to explicitly specify the language to be displayed.

3.7.1 Determining the Language to Be Displayed

If an e-manual in the language set in the Language option in the System Settings is included, the e-manual is displayed in that language. If an e-manual in that language is not available, the first language selected as an output target when outputting projects to binary files with TWL ManualEditor is used to display the e-manual.

For example, if English and French have been selected for binary output, the e-manual language is determined as follows based on the language set in the System Settings.

- English if English is set in the System Settings.
- French if French is set in the System Settings.
- English if Spanish is set in the System Settings (the language that comes first is selected because Spanish is not selected)

3.7.2 Explicitly Specifying the Language

TWL ManualEditor can be used to create an e-manual in a language not defined in the System Settings such as Dutch. If e-manuals for multiple languages are included in the e-manual binary file, and if these include e-manuals in languages not defined for in the System Settings, the language used to display the e-manual cannot be determined using the System Settings alone. In such cases, the application can select the appropriate language and specify the language to be displayed to the M2d initialization function. The M2d library initialization function whose name ends in “Ex” can be used to explicitly specify the language to be displayed for the e-manual.

Code 3-8 Explicitly Specifying a Language

```
// Select a language
u16 lang;
switch ( selectLang )
{
case LANG_EN: lang = 'en'; break; // English
case LANG_NL: lang = 'nl'; break; // Dutch
...
}

// Initialize
NTMV_M2dInitForFilePathEx(
    &manViewInfo,
    &gAllocator,
```

```
"/data/manpages_narc.blz", // E-manual binary file
"/data/m2dres_narc.blz",   // Resource file
fonts,                     // Font
lang);                     // Language to be displayed
```

4 Notes

4.1 Archive Identifiers

The E-Manual Display library mounts the e-manual binary file and resource file for the library to the file system. The following identifiers are used at this time.

- E-manual binary file: M2M
- Resource file for the library: M2U

4.2 IRQ Interrupt Handler

The E-Manual Display library does not change IRQ interrupt settings. Therefore, if an interrupt handler has been registered using `SetIrqFunction` of the TWL-SDK, the interrupt handler is called each time an interrupt is generated even while the e-manual is being displayed (even while the `NTMV_M2dRun` function is being called).

4.3 Automatic Power OFF

If `FALSE` has been set by the TWL-SDK's power control function, `PM_SetAutoExit`, `TRUE` is set temporarily by the initialization function for the E-Manual Display library to support the automatic power OFF process. The original setting is then restored by the shutdown process function.

If `PM_SetAutoExit` is `TRUE` (existing state), the E-Manual Display library does nothing.

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2007-2010 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.