SONY

Programmer's Companion for Sony CLIÉ™ Handheld

CLIÉ Software Development Kit Release 2.0

CLIÉ

Trademark Ownership Information

CLIÉ, Memory Stick and Jog Dial are registered trademarks of Sony Corporation.
Palm Computing, Graffiti, HotSync are registered trademarks of Palm, Inc. and subsidiary companies of Palm in the United States and other countries.
Palm OS is a registered trademark of Palm, Inc. in the United States.
Windows is a registered trademark of Microsoft Corporation in the United States and other countries.
All other trademarks are property of their respective owners.

Notes

Table of Contents

Introduction

Purpose of this manual

This manual describes the essential information on the software development of the CLIÉ™. It enables users to utilize the original features of the CLIÉ™ Handheld and to promote software development.
In addition, it is recommended to read the Palm OS Programmer's Companion and Palm OS SDK Reference provided by Palm, Inc.

How to read this manual

This manual provides a guideline that is newly adopted function of the CLIÉ™ Handheld on the Palm platform and the reference information for development.
The list below shows the new features and the pages to refer to for more information or details.

PEG-N700C, N710C

Original feature	Pages to refer
Jog Dial	Chapter 1, "Features." Chapter 2, "Jog Dial™ Navigator."
JogAssist	Chapter 1, "Features." Chapter 2, "Jog Dial™ Navigator." Chapter 3, "JogAssist."
Memory Stick file system	Chapter 1, "Features." Chapter 1, "Notification." Chapter 6, "Memory Stick® File System."
Hold	Chapter 1, "Features." Chapter 1, "Notification." Chapter 5, "Hold."
High resolution	Chapter 1, "Features." Chapter 7, "High Resolution : Sony HR Library."

Memory Stick audio	Chapter 1, “Features.” Chapter 1, “Notification.” Chapter 8, “Memory Stick® Audio : Sony Msa Library.”
Audio remote control	Chapter 1, “Features.” Chapter 1, “Notification.” Chapter 9, “Audio remote control : Sony Rmc Library.”

PEG-S300, S500C

Original feature	Pages to refer
Jog Dial	Chapter 1, “Features.” Chapter 2, “Jog Dial™ Navigator.”(Haven't responded to Back key.)
Memory Stick file system	Chapter 1, “Features.” Chapter 1, “Notification.” Chapter 6, “Memory Stick® File System.”

Audio Adapter(PEGA-SA500)

Original feature	Pages to refer
Memory Stick audio	Chapter 1, “Features.” Chapter 1, “Notification.” Chapter 7, “Memory Stick® Audio : Sony Msa Library.”

CLIÉ™ SDK Components

Directory components

The CLIÉ™ SDK Release2.0 is composed of the following directories

```
Sony SDK Support\
 └Rel2.0\
    ├Documentation\   An explanation of the CLIÉ™ SDK.
    └Incs\            Root directory of the header file of the CLIÉ™ SDK
       ├System\       Stores the system related header file of the CLIÉ™
       │              SDK
       └Libraries\    Stores the library related header file of the CLIÉ™
                      SDK
```

Header file

These are the header files stored in Inc directory.

Incs Directory

`SonyCLIE.h`	All the header files are integrated in this file. Including this automatically includes the rest.

System Directory

`SonySystemPublic.h`	The system related header files are integrated in this file.
`ExpansionMgr.h`	For Expansion Manager that controls a media in extension slot.
`VFSMgr.h`	For VFS Manager used to operate VFAT file system in MemoryStick media.
`SonyErrorBase.h`	Error codes unique to CLIÉ™ Handheld are defined.
`SonyHwrOEMIDs.h`	Constants unique to CLIÉ™ Handheld are defined.
`SonyKeyMgr.h`	For key events unique to CLIÉ™ Handheld and Key Manager.
`SonyChars.h`	Jog Dial-related constants are defined.
`SonyJogAssist.h`	Constants for JogAssist function are defined.
`SonySystemResources.h`	System resource of CLIÉ™ Handheld is defined.
`SonySystemFtr.h`	Features unique to CLIÉ™ Handheld are defined.
`SonyNotify.h`	For Notification Manager that notifies status change in CLIÉ™ Handheld.

Library Directory

`SonyLibPublic.h`	The library related header files are integrated in this file.
`SonyHRLib.h`	For High-resolution library.
`SonyMsaLib.h`	For Memory Stick library.
`SonyRmcLib.h`	For audio remote control library.

Software Development Environment

Software development should be made on WindowsPC. These are the required development tools.

CodeWarrior for Palm Release 6

Development tool for applications that run on C/C++ -supported Palm OS devices. This contains Integrated Development Environment (IDE) and all the tools required to develop Palm OS applications. CodeWarrior for Palm Computing platform is the recommended

development environment for CLIÉ™ applications. For more information, visit the Web site of Metrowerks.co. at <http://www.metrowerks.com/>.

Palm OS SDK 3.5

CLIÉ™ SDK is for proprietary features of the CLIÉ™ Handheld. For Palm OS basic development information including Palm OS SDK, visit the Palm OS platform Web site at <http://www.palmos.com/>.

Palm OS Emulator

Palm OS emulator (POSE) is software that emulates PalmOS platform devices including the CLIÉ™ Handheld. This emulates Palm OS environment by using ROM image. Your application can be tested with added functions such as error checking and debugging before performing validation on real machine. The emulator and ROM image of CLIÉ™ Handheld are available at the CLIÉ™ Developer Web site at <http://www.us.sonypdadev.com/>.

Installing CLIÉ™ SDK

Copying SDK

Copy directory structure under `Sony SDK Support` to CodeWarrior directory (example: `C:\Program Files\Metrowerks\CodeWarror for Palm OS R6`).

Adding an access path

To add a path to allow access to CLIÉ™ SDK header files using CodeWarrior for Palm Release6.0:

1. Open a project. From [Edit] menu, select [Starter Settings].
2. In the <Starter Settings> dialogbox, select “Access Paths” under “Target” on <Target Settings Panels>. Then, select “System Paths" on <Access Paths>. Click [Add] button.
3. In the “Please Select an Access Path” dialogbox, select “Compiler Relative” from <Path Type> list. Next, select “Sony SDK Support” under CodeWarrior directory and click [OK] button.
4. Check that “{Compiler}Sony SDK Support” is added to <System Paths>. Click [Save] button. Click  at upper right corner to quit.

Adding a header file

To add CLIÉ™ SDK header files to a source file, type in “SonyCLIE.h” as below.

```
#include <PalmOS.h>
#include <SonyCLIE.h>
#include "StarterRsc.h"
```

History

Version 1.0	- available on PEG-N700C [2001/4/9]
Version 1.1	- available on PEG-N710C [2001/6/6]
Version 1.2	- available on System Update Program for PEG-N700C [2001/6/18]
Version 1.3	- available on Audio Adapter(PEGA-SA500) [2001/9/18]
Version 1.3a	- corrected example at “Library loading” on page 103 [2001/10/16]

Part I: System Function

1

Palm OS® System Features

Features

This section describes the features that indicate the system status in CLIÉ™ Handheld. For more details on a feature, see the relevant Palm OS documents.

Feature Creator

To access the features unique to CLIÉ™ Handheld, use `sonySysFtrCreator` as a feature creator. For a `creator` argument of `FtrGet()` and `FtrSet()`API, specify `sonySysFtrCreator` and for `featureNum` argument, specify a value described in "Feature number".

Feature number

This section provides the descriptions of the feature numbers defined in CLIÉ™ Handheld.
Note that previous models do not offer these features, so an application should not determine that a device is NOT a CLIÉ™ Handheld even if the feature is NOT present. (However, if any of the features exists, a device can be regarded as CLIÉ™ Handheld.)

sonySysFtrNumSysInfoP

This gets a pointer to the structure, `SonySysFtrSysInfoType`, where system information such as usable functions and current hardware status is stored.
As for `featureNum` argument of `FtrGet()`, specify `sonySysFtrNumSysInfoP`.
The pointer will not be changed by reset.
An application should not write in the location shown by the pointer.

SonySysFtrSysInfoType structure

```
typedef struct S_SonySysFtrSysInfo {
  UInt16 revision;
  UInt16 rsv16_00;
  UInt32 extn;     /* loaded extension */
  UInt32 libr;     /* loaded libr */
```

```
    UInt32 rsv32_00;
    UInt32 rsv32_01;

    void *rsvP;
    UInt32 status;     /* current system status */
    UInt32 msStatus;   /* current MemoryStick status */
    UInt32 rsv32_10;

    UInt16 msSlotNum;  /* number of slot of MemoryStick */
    UInt16 jogType;
    UInt16 rmcType;
} SonySysFtrSysInfoType;
```

Field Descriptions

`revision` — Revision number of `SonySysFtrSysInfoType`. The number increases by one every time a new member is added. The number is 1 at default.

`rsv16_00` — Reserved. Not usable.

`extn` — Bit field that indicates the loaded and working extension. When a particular extension is working, the corresponding bit will be set (1).

There are four bits:

`sonySysFtrSysInfoExtnJog`

Jog (and also Back button, if available) is usable.

`sonySysFtrSysInfoExtnRmc`

Remote control is usable.

`sonySysFtrSysInfoExtnHold`

Hold function is usable.

`sonySysFtrSysInfoExtnJogAst`

JogAssist is usable.

For the specification of each extension, see the corresponding document.

`libr` — Bit field that indicates a loaded and usable library. When particular library is already loaded by a system, the corresponding bit will be set (1). Every library works properly only on a device that supports a corresponding function, and with a non-supporting device, a bit field will usually not be set even if a library is saved in a device using HotSync technology. However, your application should not rely on this setting to determine whether a device supports a particular function.

Here are the bits:

`sonySysFtrSysInfoLibrHR`

Sony HR Library is usable.

`sonySysFtrSysInfoLibrMsa`

Sony Msa Library is usable.

`sonySysFtrSysInfoLibrRmc`

Sony Rmc Library is usable.

For the specification of each Library, see the corresponding document.

`rsv32_00` Reserved. Not usable.

`rsv32_01` Reserved. Not usable.

`rsvP` Reserved. Not usable.

`status` The bit field that indicates the system status which changes dynamically.

There are two bit fields:

`sonySysFtrSysInfoStatusHP`

Headphones are connected.

`sonySysFtrSysInfoStatusHoldOn`

Hold feature is ON.

`msStatus` The bit field that shows the status of Memory Stick.
There are four bit fields
Note that ExpansionMgr/VFSMgr might not recognize the setting. For example, API of VFSMgr might fail to access MS even though the set bit indicates MS is inserted; this is due to the specifications of PalmOS.

`sonySysFtrSysInfoMsStatus1MS`

Memory Stick media is inserted in slot 1.
Regardless of the state of the other bits, this will be set when Memory Stick media is inserted in the slot. This means the other bits are not necessarily set just because this bit is set.

`sonySysFtrSysInfoMsStatus1StrgMS`

Physically formatted Memory Stick media storage type is inserted in slot 1. Note that this does not ensure the validity of logical format (and correct mounting of VFS.)

`sonySysFtrSysInfoMsStatus1MGMS`

Memory Stick media that supports MG(MagicGate™) is inserted in slot 1.

Note that the setting of this bit has nothing to do with MG authentication or status of `sonySysFtrSysInfoMsStatus1StrgMS` bit.

`sonySysFtrSysInfoMsStatus1WP`

Write-protected Memory Stick media is inserted in slot 1.
Note that the setting of this bit has nothing to do with physical formatting or MG authentication.

You can use either `msStatus` or Expansion Manager/VFS Manager. However, feature has these advantages:

- No need to use VFS Manager APIs and Notification

- Able to get the information that cannot be obtained by VFS Manager APIs.

- Able to get accurate information of SlotDriver level (PalmOS 3.5 can fail to issue a notification, in that case VFS Manager may not be able to detect the insertion of a card).

`rsv32_10` — Reserved. Not usable.

`msSlotNum` — Number of Memory Stick slots

`jogType` — Type of Jog Dial (including Back button) feature incorporated to a device.
The values are as follows:

`sonySysFtrSysInfoJogTypeNone`

Jog Dial navigator is not incorporated.

`sonySysFtrSysInfoJogType1`

2D type(Up/Down and Push)

`sonySysFtrSysInfoJogType2`

2D type with Back key

`rmcType` — Type of remote control incorporated into a device.
The values are as follows:

`sonySysFtrSysInfoRmcTypeNone`

Remote control is not incorporated.

`sonySysFtrSysInfoRmcType1`

AD conversion type with 6 buttons.

`sonySysFtrSysInfoRmcType2`

Audio Adapter type.

sonySysFtrNumStringInfoP

This gets a pointer to the structure, `SonySysFtrStringInfoType`, where a character string that represents system property is stored.
As a `featureNum`, argument of `FtrGet()`, specify `sonySysFtrNumStringInfoP`.
An application should not write in the location shown by the pointer.
Every character string has fixed length; If charcter string is shorter than the specified length, it will be ended with Null(`0x00`). In some cases, only null may be put in.

SonySysFtrStringInfoType structure

```
typedef struct S_SonySysFtrStringInfo {
  Char maker[16];    /*   0/0x0000: ex. "Sony Corp." */
  Char model[16];    /*  16/0x0010: ex. "PEG-S300" */
  Char ship[16];     /*  32/0x0020: ex. "Japan" */
  Char os[32];       /*  48/0x0030: ex. "Palm OS 3.5" */
  Char cpu[32];      /*  80/0x0050: ex. "Motorola..." */
  Char comment[128]; /* 112/0x0070: ex. "Personal..." */
  UInt16 code;       /* 240/0x00F0: code for comment2 */
  Char comment2[254];/* 242/0x00F2: ex. "SonyCLIE..." */
                     /* 496/0x01F0: */
} SonySysFtrStringInfoType;
```

Field Descriptions

Values in parentheses indicate character string length (in bytes):

`maker[16]` manufacturer

`model[16]` Model No.

`ship[16]` Addressee

`os[32]` OS name

`cpu[32]` CPU name

`comment[128]` `Comments in` ASCII

`code` Character code of comment2.

Here are the codes:

`sonySysFtrStingInfoCodeASCII`
: ASCII

`sonySysFtrStingInfoCode8859`
: Modified 8859-1

`sonySysFtrStingInfoCodeMSJIS`
: MS-JIS

`comment2[254]` Comment written in the set code.

sonySysFtrNumJogAstMaskP

Return the address to specify a pointer of Mask data that controls the JogAssist function.
The address doesn't be changed after reset.
See "JogAssist Mask Pointer" for more information.

sonySysFtrNumJogAstMOCardNoP

Return the address for a the card number of application to specify Mask data that controls the JogAssist function.
The address doesn't need to be changed after reset.
See "JogAssist Mask Owner" for more information.

sonySysFtrNumJogAstMODbIDP

Return the address for the database ID of application to specify Mask data that controls JogAssist function.
The address doesn't need to be changed after reset.
See "JogAssist Mask Owner" for more information.

Notification

On the CLIÉ™ Handheld, original Notifications are issued other than those of issued by PalmOS. This section explains original Notifications.
See Palm OS document for details on Notification.

Event

The following are explanations of event constant specified as available notification on the CLIÉ™ Handheld.
The application shouldn't determine the device is CLIÉ™ Handheld, based on the fact that these events are received.

`sonySysNotifyMsaStatusChangeEvent`
: issued when replaying mode of Memory Stick audio is changed.
Defined using SonyNotify.h
For receiving, ensure broadcaster field of `SysNotifyParamType` is `sonySysNotifyBroadcasterCode`.
For details, see "Memory Stick® Audio : Sony Msa Library".

`sonySysNotifyMsaEnforceOpenEvent`
: issued when Sony Msa Library is requested to suspend.
Defined using SonyNotify.h.
For receiving, ensure broadcaster field of `SysNotifyParamType` is `sonySysNotifyBroadcasterCode`.
For details, see "Memory Stick® Audio : Sony Msa Library".

`sonySysNotifyHoldStatusChangeEvent`
: issued when Hold condition is changed.
Defined using SonyNotify.h.
For receiving, ensure broadcaster field of `SysNotifyParamType` is `sonySysNotifyBroadcasterCode`.
For details, see "Hold".

Broadcaster

`sonySysNotifyBroadcasterCode` is used as broadcastser, on `sonySysNotifyMsastatusChangeEvent`, `SonySysNotifyMsaEnforceOpenEvent`, and `SonySysNotifyHoldStatusChangeEvent`.
`sysFileCExpansionMgr` is used as broadcaster on `sysNotifyCardInsertedEvent` and `sysNotifyCardRmovedEvent`.
`SysFileCVFSMgr` is used as broadcaster on `SysNotifyVolumeMountedEvent` and `sysNotifyVolumeUnmountedEvent`.
There is no argument to specify broadcaster when registering notification handler so that different broadcasters may broadcast same event. Thus, for notification handler, it's preferable to confirm the broadcaster before transaction as the code indicated below.

The example of `SonySysNotifyHoldStatusChangeEvent`

```
static Err PrvHoldNotificationHandler(SysNotifyParamType
*notifyParamsP)
{
  if (notifyParamsP->broadcaster !=
   sonySysNotifyBroadcasterCode)
     return errNone;
  if (((SonySysNotifyHoldStatusChangeDetailsP)
   (notifyParamsP->notifyDetailsP))->holdOn) {
     /* Hold is about to be ON */
  } else {
     /* Hold is about to be OFF */
  }
  ...
```

Device Detection

How to distinguish the CLIÉ™ Handheld

To distinguish the CLIÉ™ Handheld, use the feature number provided by Palm OS by comparing the value with the one defined with `SonyHwrOEMIDs.h`. Specify `sysFtrCreator for creator` parameters of `FtrGet()`.
The chart indicates the relation between feature numbers and specified values of the CLIÉ™ Handheld that have been released. Each constant is defined with `SonyHwrOEMIDs.h`

Model	sysFtrNumOEMCompanyID	sysFtrNumOEMHALID	sysFtrNumOEMDeviceID
PEG-S300	sonyHwrOEMCompanyID_Sony	sonyHwrOEMHALID_S300	sonyHwrOEMDeviceID_S300
PEG-S500C	sonyHwrOEMCompanyID_Sony	sonyHwrOEMHALID_S500C	sonyHwrOEMDeviceID_S500C
PEG-N700C PEG-N710C	sonyHwrOEMCompanyID_Sony	sonyHwrOEMHALID_N700C	sonyHwrOEMDeviceID_N700C

Below are example codes to distinguish the CLIÉ™ Handheld in practice.

```
#include <SonyCLIE.h>
...
UInt32 val;
if(!FtrGet(sysFtrCreator, sysFtrNumOEMCompanyID, &val)) {
   if (val == sonyHwrOEMCompanyID_Sony) {
        /* device might be CLIE */
   } else {
        /* device might not be CLIE */
   }
} else {
   /* something wrong ... */
}
```

Availability of functions

To determine whether a device provides a particular function unique to CLIÉ™ Handheld, you set Feature. Here is an example code that determines whether the Hold function is available or not.

```
#include <SonyCLIE.h>
...
SonySysFtrSysInfoP infoP;
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumSysInfoP,
(UInt32 *)&infoP)) {
```

```
   if (infoP && (infoP->extn & sonySysFtrSysInfoExtnHold)) {
      /* Hold function is available */
   } else {
      /* Hold is NOT available */
   }
} else {
   /* something wrong, maybe not CLIE */
}
```

For other functions, it's possible to detect the availability. See each explanation for details.

Availability of library

A particular library can be used only in a device that supports the corresponding function, which means the system that runs on that device automatically loads the library.
To determine whether a library is loaded, you check on Feature.
Here is an example code that determines whether the Audio remote control function is available or not.

```
#include <SonyCLIE.h>
...
SonySysFtrSysInfoP infoP;
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumSysInfoP,
(UInt32 *)&infoP)) {
   if (infoP && (infoP->libr & sonySysFtrSysInfoLibrRmc)) {
      /* 'Sony Rmc Library' has been loaded */
   } else {
      /* Rmc is not available */
   }
} else {
   /* something wrong, maybe not CLIE */
}
```

If this bit is not specified, the loaded library may not work properly.
Even though its specified, the library is likely to be unloaded.

2

Jog Dial™ Navigator

The Jog Dial navigator is an original feature of the CLIÉ™. Here, we describes the jog events which occur when operations are performed using the Jog Dial navigator.

Jog Event

Virtual key

When a certain operation is performed using the Jog Dial navigator, `keyDownEvent` will be issued. At this moment, `data` field of `eventType` is `_KeyDownEventType;` the value of the pressed key is stored in `chr` field; `commandKeyMask` bit is set in `modifiers` field.
These are the cords set in chr field.
For more information about `keyDownEvent` or events in general, refer to Palm OS documentation.

`vchrJogUp`	Issued when Jog Dial navigator is rotated clockwise. One event is generated on each Jog Dial click with the minimum event interval of 6 SystemTicks.
`vchrJogDown`	Issued when Jog Dial navigator is rotated counter-clockwise. One event is generated on each Jog Dial click with the minimum event interval of 6 SystemTicks.
`vchrJogPush`	Issued when Jog Dial button is pressed. This will not be issued when Jog Dial navigator is pressed continuously or rotated while being pressed.

NOTE: In SonyChars.h (previous version), this event was defined as `vchrJogPress`. This code is still usable but we strongly recommned to use `vchrJogPush.`

`vchrJogPushRepeat`	Issued when Jog Dial is pressed continuously. `autoRepeatKeyMask` in `modifiers` field will be automatically set. This event will not be issued when Jog Dial navigator is pushed and rotated at the same time.

NOTE: In SonyChars.h (previous version), this event was defined as `vchrJogPressRepeat.` This code is still usable but we strongly recommned to use `vchrJogPushRepeat.`

`vchrJogRelease`	Issued when Jog Dial navigator is released.
`vchrJogPushedUp`	Issued when Jog Dial navigator is pushed in and rotated clockwise. One event is generated on each JogDial click with the minimum event interval of 6 SystemTicks.

NOTE: In SonyChars.h (previous version), this event was defined as `vchrJogPageUp`. This code is still usable but we strongly recommned to use `vchrJogPushedUp`.

`vchrJogPushedDown`	Issued when Jog Dial navigator is pushed in and rotated counter-clockwise. One event is generated on each Jog Dial click with the minimum event interval of 6 SystemTicks.

NOTE: In SonyChars.h (previous version), this event was defined as `vchrJogPageDown`. This code is still usable but we strongly recommned to use `vchrJogPushedDown`.

`vchrJogBack`	Issued when Back Button is pressed. When Jog Dial navigator is pressed continuously, `autoRepeatKeyMask` in `modifiers` field will be set and this functions as repeat key. (This will not issued in PEG-S300)

NOTE: Note that this event key is made for the system and not for an application. In case of use, the processing should conform to the guideline to keep user interface consistent.
The code might be processed by the system extension so your application should not assume this event will be issued.

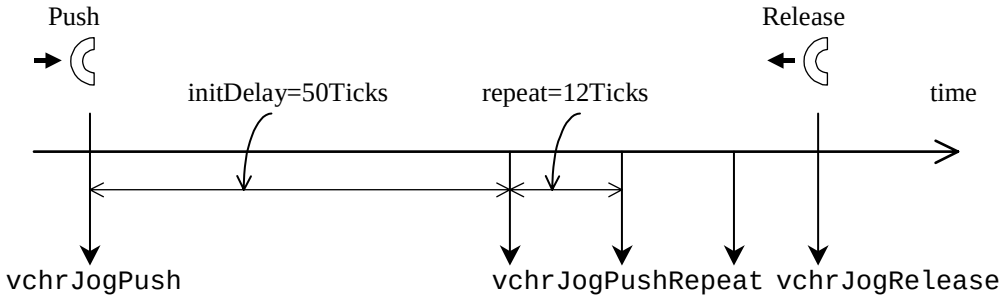
Current Palm OS cannot issue key event when key queue is full. For example, there can be a case that `vchrJogRelease` is not issued even though `vchrJogUp` has. So, the processing of an user command should always come before acceptance of a certain event.

Event interval

Now, we will show you how the issued events are related to one another.
In the description, Tick (Ticks) denotes system tick and this is counted as 10msec in the

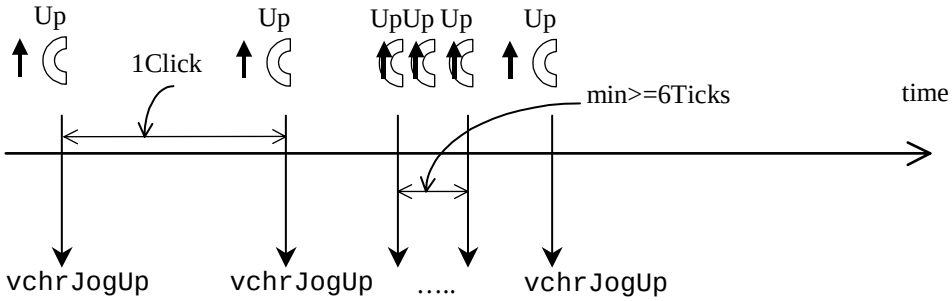
current Palm OS. For more about the system tick, refer to Palm OS documentation. Note that the interval control has an error of +/- 1 tick.

1. Push/PushRepeat/Release



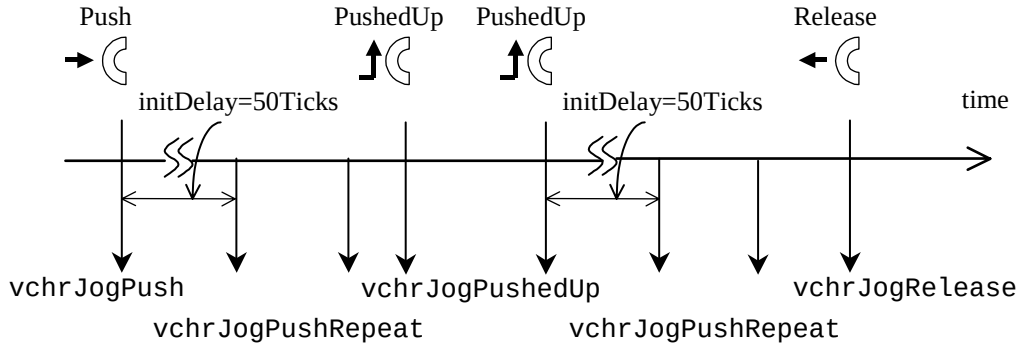
If the Jog Dial navigator has kept pressed down, the first `vchrJogPushRepeat` is generated 50 ticks later, then every 12 ticks, `vchrJogPushRepeat` is generated.

2. Up(Down)



`vchrJogUp` is generated whenever rotating the Jog Dial navigator one time. If rotating is fast, (under 6 ticks in between one click) that the event couldn't be generated.

3. Push/PushedUp(PushedDown)/PushRepeat/Release



If user rotates the Jog Dial navigator while its button is being pressed, `vchrJogPushRepeat` isn't generated[1]. If rotating stops while the button is being pressed, `vchrJogPushRepeat` starts to be regenerated after the button is pressed and kept still during the initial delay, initDelay.

Event processing

Example codes of jog event are given below.

```
#include <SonyCLIE.h>
...
Boolean JogHandleEvent (EventPtr eventP) {
Boolean handled = false;
if (eventP->eType == keyDownEvent) {
   if (EvtKeydownIsVirtual(eventP)) {
      if (eventP->data.keyDown.chr == vchrJogUp) {
         /* do 'Up' */
      } else if (eventP->data.keyDown.chr == vchrJogDown) {
         /* do 'Down' */
      } else {
         ...
      }
   }
}
```

1. On some divice, vchrJogPushRepeat is issued.

Note

Determining If Function Is Available

The following steps determine whether it's the right device equipping with the Jog Dial navigator to issue the key down event that responds to each operation.

1. Is it CLIÉ™?

   If you find the CLIÉ™ as the way shown in "How to distinguish the CLIÉ™ Handheld", `keyDownEvent` that responds to Jog Dial operations is issued[1]. Yet, not specified whether `vchrJogBack` event is issued.

2. Has the jogType of SysInfo feature been set?

   Obtain the feature shown in "Feature number" to determine the types of Jog Dial navigator. If the information is obtained and that value isn't `sonySysFtrSysInfoJogTypeNone`, `keyDownEvent` that responds to the Jog Dial is issued.

   If the value is `sonySysFtrSysInfoJogType2, the event responds to` back key is issued.

   If no feature is obtained, determine whether it's the CLIÉ™ or not by step 1.

1. There is no guarantee every CLIÉ™ is equipped with Jog dial even in the future.

3

JogAssist

Some models offer JogAssist functionality. This functionality enables the use of the Jog Dial™ navigator in applications that do not support the Jog Dial control is running. With applications that properly support the Jog Dial navigator, JogAssist automatically suspends itself from processing jog events. By minimizing the number of Jog-related tasks to be handled explicitly by the application, this function is not only useful to the user but also to the application developer.
Note that specifications are subject to change without notification.

JogAssist processing

JogAssist is designed to process unmasked jog events instead of an application and to increase user-friendliness. How JogAssist processes each jog event is described below.

vchrJogBack Assist

`vchrJogBack` is generated when the Back key is pressed. Normally, this event is processed by a system utility such as JogAssist. This allows the user to perform operations such as returning to the previous screen or cancelling an operation in any application.

NOTE: To keep user interfaces consistant, applications should not mask the Back key. If the Back key is masked, the application is responsible for providing Back key functionality equivalent to that of JogAssist.

A) No pop-up list, cursor, menu or list displayed

- Response

  Button Control is pressed. / System returns to the Home screen.

- Handling

  One of the usable and visible Button Controls in the current form is selected. The Button will be selected in the order of priorities shown below. If there is more than one button with the same priority level, the one with the smaller numerical index value will be selected. If these buttons do not exist, the application will quit to return to the Home screen

(High priority) - Cancel, Previous
- No, Close
- Done

(Low priority) - Yes, OK

NOTE: For JogAssist to utilize this event, applications should have buttons with the above labels in every form.

B) Pop-up list displayed

- Response

  Pop-up list is closed.

- Handling

  The displayed pop-up list is closed. The current item will be the one selected.

C) Cursor displayed.

- Response

  Cursor disappears.

- Handling

  The displayed cursor will disappear if the back button is pressed for less than one second.

D) Menu displayed.

- Response

  Menu disappears.

- Handling

  The menu closes.

E) List displayed.

- Response

  Goes back to the previously selected item in the list.

- Handling

  After moving the selection cursor by rotating the Jog Dial navigator, the selection returns to the previously selected item if the back button is pressed before pressing the Jog Dial navigator.

Keeping the Back button pressed provides the functionality described below. Note that the response and handling may change depending on the user settings in the Jog Preferences panel.

Check box for power off is checked in Jog panel

F) Back key pressed longer than 1 second.

- Response

  Shut off the power.

- Handling

  When Back key is pressed for longer thatn 1 second, the system turns the power off. When the key is released in less than 1 second, normal Back key processing is performed.

Check box for displaying cursor and menu is checked in Jog panel

G) No pop-up list, cursor or menu is displayed, and the Back button is pressed for more than 1 and less than 2 seconds.

- Response

  Cursor displays.

- Handling

  Cursor appears when back key is pressed longer than 1 second.

H) No pop-up list, cursor, or menu is displayed and the Back button is pressed for more than 2 second.
Or, with the cursor displayed, the Back button is pressed for more than 1 second.

- Response

  Menu displays.

- Handling

  The menu appears when the Back button is pressed for more than 2 seconds. After the first second, the cursor will be displayed temporarily but will disappear before the menu appears.
  If the cursor is already displayed, the menu appears when the back button is pressed for more than only 1 second.

I) With the “i” icon displayed, the back button is pressed for more than 2 seconds.

- Response

  Starts online help

- Handling

  In a modal form with the “i” icon, online help appears when the Back button is pressed for more than 2 seconds or for more than 1 second if the cursor is displayed.

NOTE: To keep the user interface consistent, an application should not have an interface which requires continuous pressing of Back Button.

vchrJogUp/Down Assist

`vchrJogUp` and `vchrJogDown` are generated when the Jog Dial navigator is rotated up or down. Being a frequently used event, this is generally used to move the selection cursor or to scroll text. Every application might have a slightly different user interface.

JogAssist is made to provide an independent and general user interface, so the use of this event is not limited to linguistic meaning of Up/Down.

A) No pop-up list, cursor, menu, or list displayed

- Response

  Moves the scroll car up/down in a scroll bar or performs an operation equivalent to pushing the up/down scroll buttons.

- Handling

  A scroll bar that is usable and visible in the current Form will be selected and its scroll car moves in response to the rotating the Jog Dial navigator up or down. When a scroll bar is not present, the Jog Dial navigator will act the same as pushing the up/down scroll buttons. If there is more than one scroll bar in a Form, the one with the younger index will be selected.

NOTE: To utilize this event, an application should not have more than one scroll bar in a form.

B) Pop-up list displayed

- Response

  Selection marker moves.

- Handling

  Changes the selected item in a pop-up list.

  vchrJogUp causes the selection (highlight) to move to one item up. vchrJogDown causes the selection to move to one item down.

C) Brightness/Contrast control form displayed

- Response

  Brightness control bar moves.

- Handling

  When the brightness/contrast control dialog box is displayed, this processing precedes A) and B). `vchrJogUp` causes the bar to move to the right (brightness/contrast increases); `vchrJogDown` causes it to move to the left (brightness/contrast decreases).
  In actual processing, the `chr` field of the `keyDown` event is replaced with `pageUpChr` for `vchrJogUp` and with `pageDownChr` for `vchrJogDown`.

D) Cursor displayed.

- Response

  Cursor moves.

- Handling

  Moves the cursor if displayed. Selectable objects are buttons, checkboxes, popup triggers, push buttons, selector triggers, and repeating buttons.

E) Menu displayed.

- Response

  The selection cursor in the menu is moved.

- Handling

  Moves the selection cursor in the menu if the menu is displayed.

F) List displayed.

- Response

  Moves the selection cursor in the list.

- Handling

  Moves the highlighted part in the list. Note that the highlighted item is not selected until the Jog Dial navigator is pressed and released.

vchrJogPushedUp/PushedDown Assist

The events of Jog Dial being pushed up or down. These events are less used as compared to `vchrJogUp/Down` and their use might greatly differ depending on each application's needs.
Regarding these as complementary event of `vchrJogUp/Down`, their working is similar to that of `vchrJogUp/Down`.

A) No pop-up list displayed

- Response

  Moves the scrollCar up or down (by one page at a time) in a ScrollBar.

- Handling

  See `vchrJogUp/Down`.

  The scroll car moves to the previous or to the next page corresponding to the direction of the jog rotation. The size of a "page" is defined by the pageSize in a ScrollBar object.

NOTE: To utilize this event, an application should not have more than one scroll bar in a form.

B) Pop-up list displayed

- Response

  No response.

- Handling

  A nilEvent is generated so that this event will not be passed to the system event handler to close the pop-up list.

C) Brightness control form displayed

- Response

  Brightness control bar moves.

- Handling

  See `vchrJogUp/Down`.

vchrJogPush/PushRepeat/Release Assist

`vchrJogPush`, `vchrJogPushRepeat`, and `vchrJogRelease` events are all related to the Jog Dial being pushed down. They are generally used to execute commands, so their uses differ depending on each application's needs. JogAssist must offer the user interface not depending on an application. For this reason, it is used only to select a particular item in the list. Note that the selection is not set until the release of a pushed Jog Dial navigator.

A) No pop-up list, cursor, menu, or list displayed.

- Response

  No response.

- Handling

  No processing will be made. The jog event will be passed to the system event handler.

B) Pop-up list displayed.

- Response

  Sets the selected list item

- Handling

  `vchrJogRelease` (Jog Dial navigator is released) sets the selected list item (current item) and closes the popup list
  Replaces with a nilEvent so the pop-up list will not disappear by passing `vchrJogPush` and `vchrJogPushRepeat` events.

C) Cursor displayed.

- Response

  Sets the selected cursor item.

- Handling

  When the cursor is displayed, `vchrJogRelease` sets the selected cursor item and the cursor disappears.

D) Menu displayed.

- Response

  Sets the selected menu item.

- Handling

  `vchrJogRelease` selects the highlighted menu item and closes the menu.

E) List displayed.

- Response

  Sets the selected list item.

- Handling

  `vchrJogReleease` sets the selected list item.

JogAssist Mask Specification

It is possibile for users to specify different behavior from those which are defined by the application: In applications designed to handle Jog Dial navigator events explicitly, JogAssist functionality may interfere with its Jog Dial behavior and may cause undesirable results.
To cope with these issues, there is a system to restrict JogAssist functionality temporarily on the CLIÉ™.

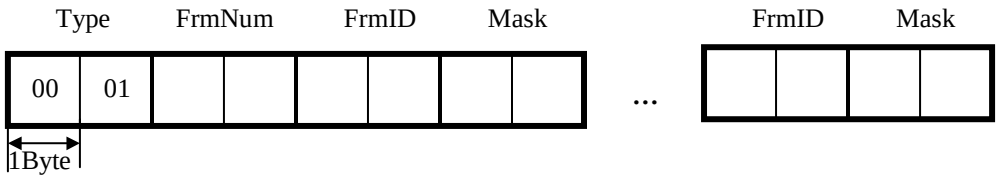
JogAssist Mask Data

To disable the JogAssist function, the application must specify Mask data.

Below is the format of the currently defined Mask data.

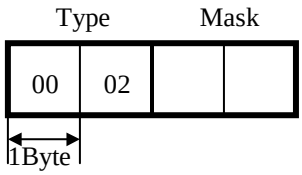
- Type 1

  It specifies the masks for each form in the application. (Forms that are not specified in the mask will have full JogAssist functionality available.)



- Type 2

  Specifies effective masks for all forms in the application, including system forms such as alert or help.

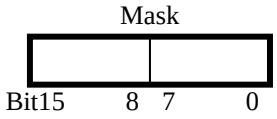


Each field should follow the states below.

- Describe the numeric value with binary BigEndian.
- Specify the mask type in the Type field. These values are defined in `SonyJogAssist.h`

- Mask field is a bitmask that specifies the events to mask.

  1 means masked (JogAssist function is disabled), 0 means unmasked (JogAssist function is enabled). However, whether JogAssist function actually works in unmasking depends on the specification of the extension software which functions then. It is not guaranteed that any JogAssist function works.
  The Reserved bits must be set to zero. These bits are likely to be defined by Sony in the future.
  See, `SonyJogAssist.h` for the actual definition of each bit.



Bit0: `vchrJogUp`
Bit1: `vchrJogDown`
Bit2: `vchrJogPushedUp`
Bit3: `vchrJogPushedDown`
Bit4: `vchrJogPush`
Bit5: `vchrJogRelease`
Bit6: `vchrJogPushRepeat`
Bit7: `vchrJogBack`
Bit8〜15: `Reserved`

Example:

`0x0070 ->`	Mask `vchrJogPush/vchrJogRelease/ vchrPushRepeat`
`0x0000 ->`	Unmask all events. (Same as not specifying mask data.)

- In the `FrmNum` field, specify the number of forms for which to set the mask.
- In the `FrmID` field, specify the form ID of the form for which to set the mask. (Note that the form ID must be used, not resource ID, although the two usually have the same value.)

The following is an example of Mask data in hexadecimal format.

- `0x0001000203E80003044C0018`

  Type 1, the two Forms that use masks have form IDs 1000 and 1100. The specified masks: Form 1000 masks `vchrJogUp/vchrJogDown`. Form 1100 masks `vchrJogPush/vchrJogRelease`.

JogAssist Mask Pointer

JogAssist requires a JogAssist mask pointer to the top address of the Mask data. The application must specify the JogAssist mask pointer in a system-defined address.
The address where the mask pointer will be set can be obtained by using `FtrGet()` with `sonySysFtrNumJogAstMaskP` as the feature number, as demonstrated below:

```
#include <SonyCLIE.h>
...
```

```
UInt16 **maskPP;
UInt16 mask[MASK_DATA_LENGTH];
...
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumJogAstMaskP,
(UInt32 *)&maskPP)) {
   /* Mask can be set */
   *maskPP = mask;
} else {
   /* something wrong ... */
}
```

After a system reset, the contents of the specified address is set to NULL. This address itself will not be changed after the system reset.
The pointers stored in features are shared among all applications and Extensions. Thus, it is highly recommended that all applications and extension software (which has an original event loop.) use the procedure below to set the JogAssist mask pointer properly when activating and finishing. It is recommended to follow these procedures even when a JogAssist mask is unnecessary.

- When activating, save the old mask pointer, and when finishing, restore it.
- Before sub-launching other applications, set the mask pointer to NULL, and then reset it to the original value afterward.

JogAssist Mask Owner

Palm OS sometimes can activate other applications or forms on its own independently of the current application. If mask data is specified for the current application, it still is valid unless the sub-launched application specifies its own mask. This may cause the sub-launched application to not respond to Jog Dial navigator events, which may be inconvenient for the user. To avoid this the card number and local ID of the application can be used to set mask data for only the specified application (mask owner). The address specifying this data can be obtained as a Feature, similar to the one used to store the mask pointer.
The code below demonstrates how to set the JogAssist mask owner.

```
#include <SonyCLIE.h>
...
UInt16 cardNo, *ftrCardNoP;
LocalID dbID, *ftrDbIDP;
...
SysCurAppDatabase(&cardNo, &dbID);
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumJogAstMOCardNoP,
(UInt32 *)&ftrCardNoP)
&& !FtrGet(sonySysFtrCreator, sonySysFtrNumJogAstMODbIDP,
(UInt32 *)&ftrDbIDP)) {
   /* Mask can be set */
   *ftrCardNoP = cardNo;
```

```
   *ftrDbIDP = dbID;
} else {
   /* something wrong ... */
}
```

If the local ID of the mask owner is NULL, JogAssist will not be able to determine which application is the mask owner, and thus the current mask will be valid for all applications. So we encourage users to set the value for the mask-owner on the applications that Palm OS can sub-launch other applications. (However, it is only necessary for such applications that Palm OS adds the original item in the menu by itself and sub-launch applications as Address book.) When done, restoring the original data also is recommended.

Support to JogAssist mask system

JogAssist loaded on the CLIÉ™ works by utilizing the JogAssist mask system. It is recommended that other kinds of jog utility softwarealso employ this mask value.
Note that the mask does not affect JogAssist functionality when a pop-up list is displayed. This is because the event loop in the system is waiting for the event while the popup list is displayed so that the application can't process it even though the mask is specified.

Notes

Determining If JogAssist Is Available

To determine if JogAssist is available on a device, you can obtain a `SonySysFtrSysInfoType` structure by using `sonySysFtrNumSysInfoP` as the feature number and then checking the `sonySysFtrSysInfoExtnJogAst` bit in the `extn` field.

Preferences

JogAssist functionality can be set via the “Jog” panel in the Preferences. Changing the preferences can be performed only by using the preferences panel not by using the program.
The current “Jog” panel has the items below[1]. These settings are retained after a soft reset.

- [Power On with BACK button] check box

  Check to allow the device to be powered on by pushing the BACK button. This does not depend on the state of the [Use JogAssist] check box.

[1] The panel setting items and the values are subject to change in the future.

- [Use JogAssist] check box

  Check to enable JogAssist. If you intend to use a software functionally equivalent to JogAssist, enabling JogAssist may interfere with it. In this case, uncheck this box.

- [Select Applications] button

  Used to set particular applications for which JogAssist should be disabled. Tap this to display the [Select Applications] dialog box. This is valid only when [Use JogAssist] is ON.

- Select Additional Menu] button

  Used to add new system items to the first menu in an application. Tap this to display the [Additional menu] setting screen. This is valid only when [Use JogAssist] is ON.

- [Control Power] or [Power Off] check box

  Check to allow the device to be powered off by pushing the BACK button for more than 1 second. This is valid only when [Use JogAssist] is ON; OFF when [Display Cursor/Menu] is ON.

- [Display Cursor/Menu] check box

  Check to allow the cursor or the menu to be displayed by pushing the BACK button for more than 1 second.
  When enabled:

  – Pushing the BACK button down for more than 1 second displays the cursor. If the cursor already is displayed, the menu appears.

  – Pushing the BACK button down for more than 2 seconds displays the menu. After the first second, the cursor temporarily appears before the menu is shown.

  – This is valid only when [Use JogAssist] is ON; OFF when [Power Off] is ON.

Mask Setting

- Note that the JogAssist specification is subject to change. Thus, applications that depend on specific Jog Dial navigator behavior should not depend on JogAssist and should process jog events explicitly using an appropriate mask.
- If no masking is required, set the mask pointer or the mask owner to NULL to indicate that your application is not masked.

4

Audio Remote Control

Some CLIÉ™ models allow you to use the audio remote control as an external input terminal. This chapter describes the events which will be issued whenever an operation is performed by using the audio remote control. Library is also provided to allow more sophisticated use. For more information about the library, see "Audio remote control : Sony Rmc Library".

Remote Control Event

Virtual Key

If you perform a specific operation using the audio remote control supplied with CLIÉ™, the corresponding virtual key, `keyDownEvent` is issued.
See PalmOS documentation about `keyDownEvent` or events in general.
`Data` field of the `eventType` in the `keyDownEvent` is `_KeyDownEventType` and the value to indicate the kinds of operation is stored in its `chr` field. In the `modifiers` field, `commandKeyMask` bit is set.

The codes specified in the `chr` field are given in the following.

`vchrRmcKeyPush`	issued when any keys of remote control is pressed. `autoRepeatKeyMask` in the `modifiers` field is set and issued while the key continues to be pressed `keyCode` field determines what key is pressed.
`vchrRmcKeyRelease`	issued when key pressing of remote control is stopped. `keyCode` filed is unsettled.

`vchrRmcKeyRelease` isn't always issued corresponding to `vchrRmcKeyPush`. Because PalmOS event queue may overflow. Thus, an application waiting for only `vchrRmcKeyRelease` should not be developed.
A/D value, a physical interface with audio remote control is stored in the key Code. The value is generated when a button is pressed and has a few ranges. In audio remote control with 6 buttons loaded on the CLIÉ™, there is a relation in respose between values and buttons as below. If two buttons are pressed at a time, A/D value like validating higher priority (play side) buttons will be returned.

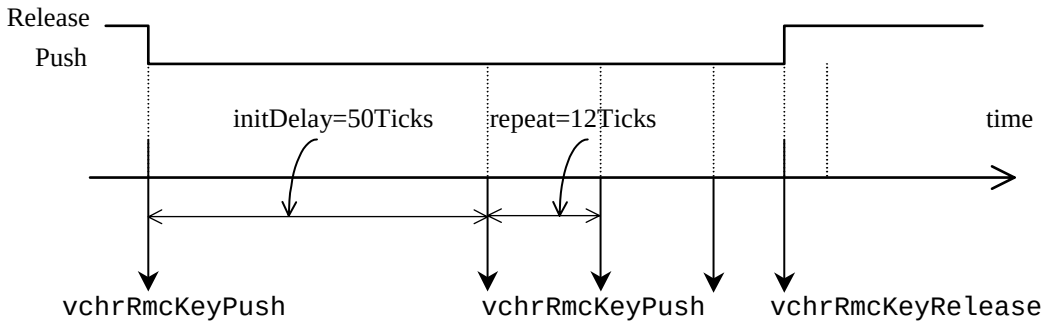
Button	keyCode(A/D value)	
	Min	Max
Play	3235	3372
FR Play	3030	3167
FF Play	2430	2566
Stop	1938	2048
Volume Down	1802	1911
Volume Up	1665	1761

Event intervals

How the events are associated with one another will be explained.
Tick (Ticks) represents system tick and it equals 10msec according to latest PalmOS. For more information about system tick, refer to PalmOS document.

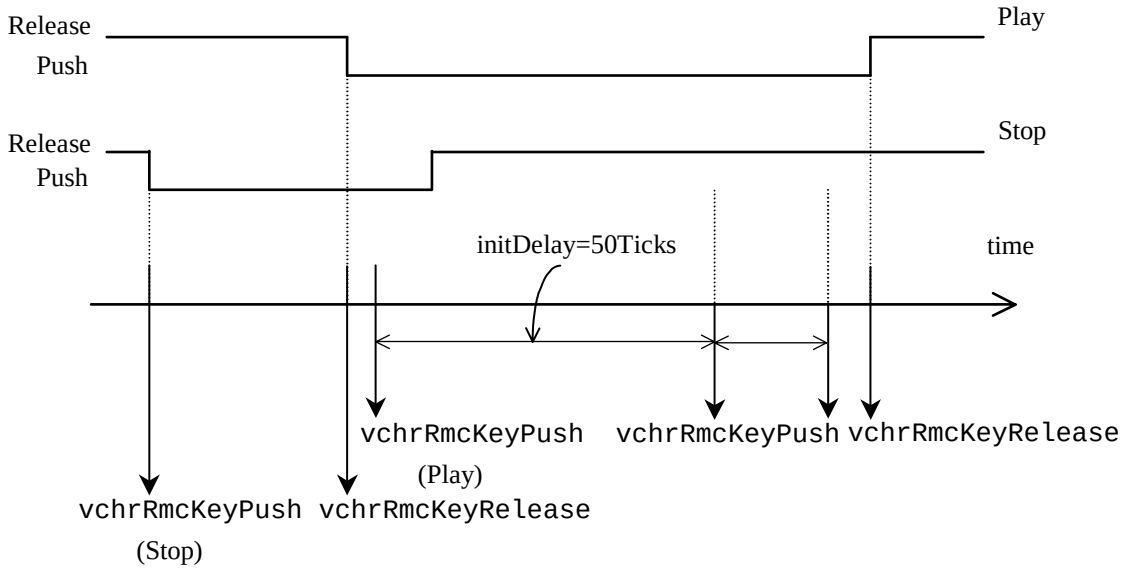
Every interval has an error of +/- 1 tick.

1. Push/Release



As a button is pushed, `vchrRmcKeyPush` occurs. If it is kept pushed in, `vchrRmcKeyPush` will occur again after 50 ticks. After this, `vchrRmcKeyPush` will occur every 12 ticks.
`vchrRmcKeyRelease` occurs as the button is released.

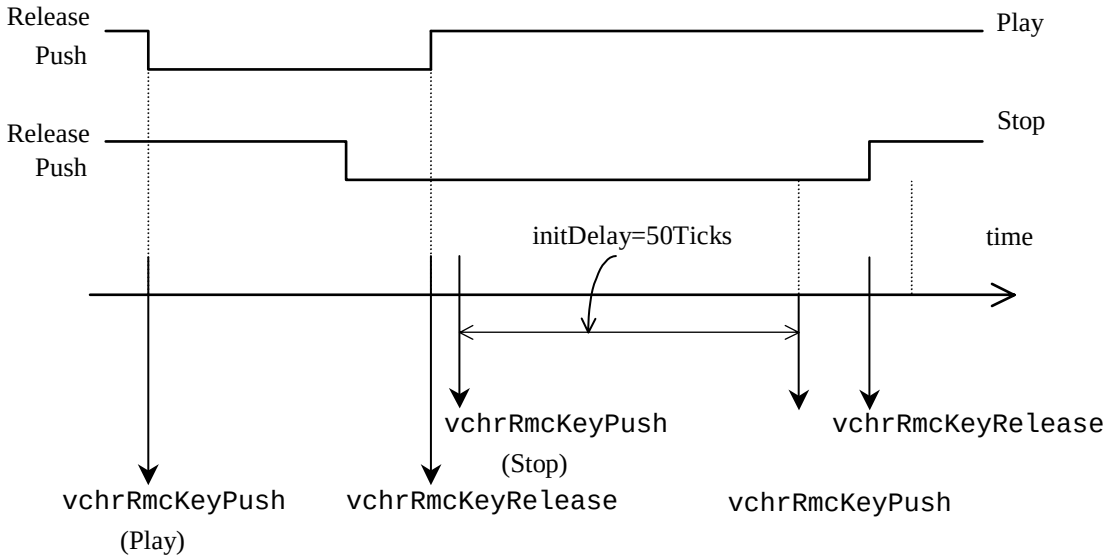
2. Push of two buttons overlapped (When pushing the button with a high priority later.)[1]



If a button with a higher priority (button A) is pushed while another button with a lower priority (button B) is pushed, the system determines button B is already released at this moment.

1. After the two buttons whose A/D value difference is under 50 are pressed, if the different button is pressed around, `vchrRmcKeyRelease` event isn't issued during the time for the current driver's restriction. This specification is subject to change.

3. Push of two buttons overlapped (When pushing the button with a low priority later.)



If a button with a lower priority (button A) is pushed while another button with a higher priority (button B) is pushed, the system ignores button A.
However, if button A is still pushed when button B is released, the system will respond to it.

Event processing

If you process remote control event, consider some ranges of A/D value stored in `keyCode` field.
By using a macro written on the header file(`GetRmcKey()`), easy mapping to 6 buttons is available. Sample codes are given below.

```
#include <SonyCLIE.h>
...
static Boolean MainFormHandleEvent(EventPtr eventP)
switch (eventP->eType) {
case keyDownEvent:
  switch (eventP->data.keyDown.chr) {
  case vchrRmcKeyPush:
    switch (GetRmcKey(eventP->data.keyDown.keyCode)) {
    case rmcKeyPlay:
      /* Play key has been pushed */
      break;
    case rmcKeyFrPlay:
```

```
        /* FR_Play key has been pushed */
        break;
        ...
      default:
        break;
      }
      break;
    case vchrRmcKeyRelease:
      /* remocon key has just been released */
      break;
    default:
      break;
    }
    break;
    ...
}
```

Notes

Determining If Audio Remote Control Is Available

To determine if a device supports audio remote control and issues corresponding `keyDownEvent`, check the setting of `sonySysFtrSysInfoExtnRmc` bit in `extn` field of `SonySysFtrSysInfoType` structure obtained by using `sonySysFtrNumSysInfoP` as a feature number.

Auto-On

When a button on the remote control is pressed while the power is off, the key event of `poweredOnKeyMask` set to modifiers field will be generated powering a device on. However, the screen will not light up and auto-off timer will not be reset[1].
So, if your application needs to turn on the power of both a device and its screen at the same time, call `EvtResetAutoOffTimer()` API; or if you want to turn the power on only when a particur button on the remote control is pressed, call `EvtResetAutoOffTimer()` API as needed.

Application of Remote Control Interface

A driver loaded into the CLIÉ™ doesn't assume an attached audio remote control alone. So the A/D values obtained from physical interface through remote control aren't converted to the fixed variables such as play and stop. They are stored in the event as is.

1. Some devices may turn on the screen as the remote-control button is pressed, however, that will be modified soon following the spec of this manual.

That’s why, other remote controls, even though not provided by Sony, are able to connect with the CLIÉ™ (only if they meet the reqirements of hardware.).
The applicable possibility extends wider like games, if a remote control to generate A/D value segmented into narrower range is developed and an application to interpret those values directly is provided to the user. However, the A/D values must be output as the table shown Virtual Key to be compatible with the application that assumes the standard loaded audio remote control.

5

Hold

Hold provides functions of key lock and LCD-off.
If the power is on, Hold function helps to conserve battery power because the LCD turns off and protects malfunctions by inadvertent key-pressing. If the power is off, the Hold function specifies the key lock alone.
This section explains the specifications of the Hold function.

Hold User Interface

Turn on and off

Slide the tab up and down, where located on the left side[1] of the CLIÉ™. Upward slide turns on and downward turns off. This could perform at any time you like, regardless of the power mode and any performing applications. When sliding upward, it activates after the message of Hold on the display. On the other hand, when sliding downward, it's released without any message.
Below are the cases that the Hold doesn't work shortly after upward tab slide. It will work after these procedures:

- After Memory Stick media insertion during file system recognition.
- Formatting the Memory Stick media.
- Some object, like button, is being tapped on the Graffiti area or in the display by stylus.
- The system is in progress, showing message like "Please wait for a while".

Hold on spec

Below is a chart to show the performance of each function. (O is valid, X is invalid, - is originally invalid, regardless of the Hold mode.)

1. It may change with devices.

Power	LCD	Button				Pen	Audio Remo-con	LED
		Power	Appl	Jog	Cradle			
ON	X	X	X	X	O	X	O	O
OFF	-	X	X	-	O	-	O	-

If you perform Hold on, while the power is on, the performing applications keep working even though invisible on the LCD display. In general, it's not necessary for the application to check whether a device is Hold on.
The Hold is fully independent of the auto shut-off. Thus, the power automatically shuts off regardless of the Hold mode (if the auto shut-off has been set before).

Application Interface

Getting current Hold status

Basically, the Hold function has only to do with an user interface. However, some interfaces are available for application to let it obtain relevant information. One use of this is plot suspension. By this, power consumption reduces and the response rate of remote control speeds up when Hold is enabled.
Here is the code that gets current Hold status from Feature. The value changes in real time.

```
#include <SonyCLIE.h>
...
SonySysFtrSysInfoP infoP;
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumSysInfoP,
(UInt32 *)&infoP)) {
  if (infoP) {
    if (infoP->status & sonySysFtrSysInfoStatusHoldOn)) {
      /* Hold is ON (active) */
    } else {
      /* Hold is OFF (not active) */
    }
  } else {
    /* something wrong, maybe not CLIE */
  }
}
```

Obtained pointer remains unchanged unless a device is reset.
An application should not write in the area indicated by the pointer.

Receiving change in Hold status

Everytime Hold is turned ON or OFF, `sonySysNotifyHoldStatusChangeEvent` Notification is issued. `holdOn` field tells whether Hold is ON or Off: If true, it is active; if false, it is not.
Lock field indicates the locked feature when Hold is ON (Note that the present sytem setsKey (`sonySysNotifyHoldLockKey`), Pen (`sonySysNotifyHoldLockPen`), and Screen (LCD) (`sonySysNotifyHoldLockScreen`) to 1 (Lock).
These codes register and process received Notification, respectively.

Registering received Notification

```
#include <SonyCLIE.h>
UInt16 cardNo;
LocalID dbID;
DmSearchStateType state;
DmGetNextDatabaseByTypeCreator(true, &state,
  myType, myCreator, true, &cardNo, &dbID);
SysNotifyRegister(cardNo, dbID,
  sonySysNotifyHoldStatusChangeEvent,
  PrvHoldNotificationHandler, sysNotifyNormalPriority,
  (void *)anyP);
```

Processing received Notification

```
static Err PrvHoldNotificationHandler(SysNotifyParamType
*notifyParamsP)
{
  if (notifyParamsP->broadcaster !=
  sonySysNotifyBroadcasterCode)
    return errNone;
  if (((SonySysNotifyHoldStatusChangeDetailsP)
   (notifyParamsP->notifyDetailsP))->holdOn) {
    /* Hold is about to be ON */
  } else {
    /* Hold is about to be OFF */
  }
  ...
```

Notification will be issued immediately after the Hold switch is turned on or off. So, an application receives ON Notification after enabled Hold is known to the user (that is, when “Hold” is displayed on LCD).
Notification is issued only when CLIÉ™ is powered. This means the switching during the power-off will not affect Hold status. Thus, ON and OFF Notifications are not necessarily issued in pairs.

Note

Determining If Function Is Available

To defermine if the system offers the hold function, see “Availability of functions”.

6

Memory Stick® File System[1]

On the CLIÉ™, Memory Stick is available as an expansion memory slot.
An application is accessible to the file of Memory Stick media using provided API.

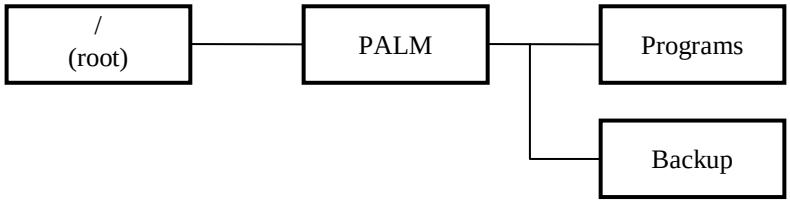
File System Format

Logical Format

On the Memory Stick media, logical format is MS-DOS compatible format defined on the Memory stick format specification. However, it's recognized as virtual file system (VFS) for the application.

Directory structure

There is a definition of file name and storing position in the memory stick file system so that users can handle the Memory Stick media and store the file easily.
The directory structure is defined below.



Basically, only pre-assigned directories should be placed under the ROOT directory. The files should not be placed under the ROOT directory immediately since they have a specific one to be stored according to the format.

1. It is equivalent to Expansion Manager and VFS Manager supported by Palm OS 4.0, and is compatible with these.

The use of each directory

/(Root) The volume of file system is mounted.

PALM This directory is exclusive for the use of Palm OS and applications.

In general, the application shouldn't place files immediately under this directory.

Programs This Directory is for the exclusive use of applications.
Normally, it creates subdirectory for each application to store those respective application files.

Backup This directory is exclusive for the use of SYSTEM BACKUP and RESTORE of PalmOS®.
The application access to this directory is forbidden.

Name Specification

Pass name Specify absolute path name to access a file in Memory Stick media. "." (current directory) and ".." (parent directory) cannot be used.
To delimit each directory, '/' is used.
The path name must be terminated by NULL.
The length of the path name should be within 255 characters including NULL and the directory's delimiters.

File name CP932 character set can be used, but excludes those indicated below.

- ASCII characters forbidden to use
  `"  *  /  :  <  >  ?  \  |`

The file name string must be within 255-byte characters. (excludes NULL)

Volume name It has the same restriction as file name.

Volume and Slot

Memory Stick file system can be accessible by being fixed into the system (Being mounted). The entire mounted file system is called **Volume** that is given a volume reference number. An application calls API by specifying the volume number.
The hole to insert the Memory Stick media on the device calls **Slot** and the physical medium attachment to insert the slot calls **Card**. Slot has a **Slot reference number** to identify the card existence easily. However, the application doesn't need to be aware of the slot condition.

File System Notification

Event

Notification event is issued when inserting and removing the Memory Stick media.

sysNotifyCardInsertedEvent

Issued when the Memory Stick is inserted to the CLIÉ™.
Memory Stick slot reference number can be obtained from `notifyDetailsP` which is a parameter of `SysNotifyParamType` argument passed in to Notification Handler.

sysNotifyCardRemovedEvent

Issued when Memory Stick is removed from the CLIÉ™.
The Memory Stick slot reference number can be obtained from `notifyDetailsP` which is a parameter of `SysNotifyParamType` argument passed in to Notification Handler.

sysNotifyVolumeMountedEvent

Issued when Memory Stick file system is mounted by the system correctly.
The `NotifyDetailsP` parameter of `SysNotifyParamType` argument passed in to Notification Handler function can be casted to `VFSAnyMountParamTypePtr`. Thus, VolRefNum and mountclass('libs') of the mounted volume can be passed to the Notification Handler.

sysNotifyVolumeUnmountedEvent

Issued when Memory Stick file system is unmounted from the system.
The `NotifyDetailsP` parameter of `SysNotifyParamType` argument passed in to Notification Handler function can be casted to `VFSAnyMountParamTypePtr`. Thus, VolRefNum and mountclass('libs') of the mounted volume can be passed to the Notification Handler.

The sequence of event issuing

The explanations are given below for the sequence of issuing each notification when the volume is mounted and unmounted.

Memory Stick Media Insertion

1. When a Memory Stick media is inserted into the handheld expansion slot, the system instructs Expansion Manager that a card has been inserted through a slot driver.
2. Expansion Manager broadcasts `sysNotifyCardInsertedEvent` through Notification Manager.
3. Each Notification Handler for `sysNotifyCardInsertedEvent` is called.

4. Expansion Manager receives `sysNotifyCardInsertedEvent` in the lowest priority and precedes mounting at step 5 and below unless `expHandledVolume` in the `sysNotifiyParamType`. `handled` field is 1.
5. Expansion Manager checks the Memory Stick media whether it's a right storage card through a slot driver.
6. If it's determined as a right one, Expansion Manager mounts the Memory Stick File system to VFS Manager.
7. If mounting succeeds, VFSMgr broadcasts `sysNotifyVolumeMountedEvent` through Notification Manager.
   If it fails, `VFSVolumeFormat` is called. It gives a dialog to ask user whether to format the memory stick or not.
   If the user chooses to format then the formatting is successfully complete, after `sysNotifyVolumeMountedEvent` will be broadcasted again through Notification Manager.
   Memory Stick file system isn't mounted if user cancel the format procedure.
8. The notification handler for `sysNotifyVolumeMountedEvent` is called.

Memory Stick Media Removal

1. When the Memory Stick media is removed from the handheld expansion slot, the system instructs Expansion Manager that a card has been removed.
2. Expansion Manager issues `sysNotifyCardRemovedEvent` through Notification Manager.
3. Expansion Manager receives `sysNotifyCardRemovedEvent` in the highest priority. If the Memory Stick file system is mounted, unmounting is proceeded at step 7 and below. If not mounted, the procedure below will be executed.
4. Expansion Manager precedes unmounting with VFS Manager.
5. If unmounting succeeds, system issues `sysNotifyVolumeUnmountedEvent` though Notification Manager.
6. Notification handler is called for `sysNotifyVolumeUnmountedEvent`.
7. Notification handler is called for `sysNotifyCardRemovedEvent`.

handled Field

`SysNotifyParamType.handled` field is defined by Boolean. However, four notifications connected to Memory Stick file system, are handled as bit field for the cooperative work with expansion manager, associating system and application.

The following Bit fields are defined now.

- `expHandledVolume`

  Use in case of `sysNotifyCardInsertedEvent` and `sysNotifyCardRemovedEvent`.
  If specified `(1)`, Expansion Manager doesn't call VFS Manager for the procedure of mount and unmount of the file system.
  For instance, Specify it if you would like to mount the file system that's not supported by OS.

- `vfsHandledUIAppSwitch`.

  Use in case of `sysNotifyVolumeMountedEvent` and `sysNotifyVolumeUnmountedEvent`.
  If specified, system, system with the function of activating application, and associating application won't switch over to application. For instance, specify this bit when you would like to activate the application automatically, which reads the user's original activating script and stores it in the assigned directory.

Handling Instructions for Notification

- If a file is formatted during the mounting of the file system, `sysNotifyVolumeUnmountedEvent and sysNotifyVolumeMountedEvent` will occur to cancel the mounting.
- Notification will not be sent in some cases. Here are some of those cases:
  - Logical file format in the MS is not right.
  - The CLIÉ™ needs battery charging
  - Memory capacity is not enough to handle mount processing.
- To register for notification using `SysNotifyRegister()` API, we recommend to set the priority as below:
  - If you want to receive it only when an applicaiton is normally started (started using `sysAppLaunchCmdNormalLaunch`) set `sysNotifyNormalPriority.`
  - If you want to receive it also for background processing, set a value larger than `sysNotifyNormalPriority.`
- If the Memory Stick media is inserted or removed when the power is off, the notification will be issued but the screen (LCD) will not light up. If you want to explicitly notify the user the change of plot, let the power on by using `EvtResetAutoOffTimer()` API.
- If you want to run another application in the reception handlers, `SysNotifyVolumeMountedEvent or sysNotifyVolumeUnmountedEvent`, `SysUIAppSwitch()` should not be called directly, or the succeeding handlers may not be executed.
  In that case, `vfsHandledUIAppSwitch` is set in the reception handler. Then, user-defined notification will be issued by using `SysNotifyBroadcastDeferred()` to go through handler processing. Lastly, call `SysUIAppSwitch()` API in the reception handler for the user-defined notification.
  Notification Handler issued using `SysNotifyBroadcastDeferred()` will be executed, but those issued by `SysUIAppSwitch()` will not.

File System API

Both VFS (Virtual File System) manager and Expansion Manager provide Memory Stick File System API.

Data Structure

FileInfoType

```
typedef struct FileInfoTag{
  UInt32  attributes;
  Char    *nameP;
  UInt16  nameBufLen;
} FileInfoType, *FileInfoPtr;
```

Field descriptions

`attributes`	Attributes of file: including read-only, system file, directory, and archive.
`nameP`	Pointer to the buffer that receives a name of file or directory as `VFSDirEntryEnumerate()` is executed.
`nameBufLen`	Buffer size of nameP(Number of bytes).

VFSAnyMountParamType

```
typedef struct VFSAnyMountParamTag {
  UInt16  volRefNum;
  UInt16  reserved;
  UInt32  mountClass;
} VFSAnyMountParamType;
```

Field descriptions

`volRefNum`	Volume reference number.
`reserved`	Reserved.
`mountClass`	Mount class. Indicates a class of file system.

VFSSlotMountParamType

```
typedef struct VFSSlotMountParamTag {
  VFSAnyMountParamType  vfsMountParam;
  UInt16  slotLibRefNum;
  UInt16  slotRefNum;
} VFSSlotMountParamType;
```

Field descriptions

`vfsMountParam`	`VFSAnyMountParamType` (See the descriptions given above.)
`slotLibRefNum`	Slot library reference number
`slotRefNum`	Slot reference number

VolumeInfoType

```
typedef struct VolumeInfoTag{
  UInt32  attributes;
```

```
  UInt32  fsType;
  UInt32  fsCreator;
  UInt32  mountClass;
  UInt16  slotLibRefNum;
  UInt16  slotRefNum;
  UInt32  mediaType;
  UInt32  reserved;
} VolumeInfoType, *VolumeInfoPtr;
```

Field descriptions

`attributes`	Volume attributes: read-only, hidden.
`fsType`	A type of file system (ex. FAT file system).
`fsCreator`	Creator ID of file system library.
`mountClass`	Mount class.
`slotLibRefNum`	Reference number of slot library.
`slotRefNum`	Slot reference number.
`mediaType`	Media type. Indicates type of a card (ex. Memory Stick)
`reserved`	Reserved.

ExpCardInfoType

```
typedef struct ExpCardInfoTag {
  UInt32  capabilityFlags;
  Char    manufacturerStr[expCardInfoStringMaxLen+1];
  Char    productStr[expCardInfoStringMaxLen+1];
  Char    deviceClassStr[expCardInfoStringMaxLen+1];
  Char    deviceUniqueIDStr[expCardInfoStringMaxLen+1];
}ExpCardInfoType, *ExpCardInfoPtr;
```

Field descriptions

`capabirityFlags`	Flag of card information. Indicates free space available, reading and writing capabilities.
`manufactureStr`	Name of manufacturer.
`productStr`	Name of product.
`deviceClassStr`	Classification of product.
`deviceUniqueIDStr`	Unique ID of product.

Constants

Error codes of Expansion Manager

`expErrUnsupportedOperation`
: Unsupported or undefined opcode and/or creator.

`expErrNotEnoughPower`
: The required power is not available.

`expErrCardNotPresent`
: No Memory Stick media is present.

`expErrInvalidSlotRefNumber`
: Slot reference number is bad.

`expErrSlotDeallocated`
: Slot reference number is within valid range, but has been deallocated.

`expErrCardNoSectorReadWrite`
: The Memory Stick media does not support the SlotDriver block read/write API.

`expErrCardReadOnly`
: The Memory Stick media does support R/W API but the card is read only.

`expErrCardBadSector`
: The Memory Stick media does support R/W API but the sector is bad.

`expErrCardProtectedSector`
: The Memory Stick media does support R/W API but the sector is copyright protected.

`expErrNotOpen` Memory Stick File System library or Memory Stick slot driver library has not been opened.

`expErrStillOpen` Memory Stick File System library or Memory Stick slot driver library is still open.

`expErrUnimplemented`
: This API is unimplemented.

`expErrEnumerationEmpty`
: No values remaining to enumerate.

Error codes of VFS Manager

`vfsErrBufferOverflow`
: The buffer passed in is too small.

`vfsErrFileGeneric` General file error.

`vfsErrFileBadRef` The fileref is invalid (has been closed, or was not obtained from VFSFileOpen()).

`vfsErrFileStillOpen`
: Returned from VFSFileDelete if the file is still open.

`vfsErrFilePermissionDenied`
: Cannot execute this API.

`vfsErrFileAlreadyExists`
A file with this name already exists in this location.

`vfsErrFileEOF` File pointer is at the end of file.

`vfsErrFileNotFound`
File was not found at the specified path.

`vfsErrVolumeBadRef`
The volume reference number is invalid.

`vfsErrVolumeStillMounted`
Returned from FSVolumeFormat if the volume is still mounted.

`vfsErrNoFileSystem`
No installed filesystem supports this operation.
(It might be returned if volume reference number or file reference number is invalid.)

`vfsErrBadData` Corrupted file data vfsErrDirNotEmpty
Cannot delete a non-empty directory.

`vfsErrBadName` Invalid filename, path, or volume label.

`vfsErrVolumeFull` Not enough space left in volume.

`vfsErrUnimplemented`
This call is not implemented.

`vfsErrNotADirectory`
This operation requires a directory.

`vfsErrIsADirectory`
This operation requires a file, not a directory.

`VfsErrDirectoryNotFound`
The path leading up to the new file does not exist.

File Stream APIs

VFSFileCreate

Purpose Generates new files.

Prototype
```
Err VFSFileCreate( UInt16 volRefNum,const Char *pathNameP)
```

Parameters
`-> volRefNum` Volume reference number
`-> pathNameP` Absolute full path name for the newly created file.

Result `errNone`
`expErrNotOpen`

```
vfsErrFileGeneric
vfsErrVolumeBadRef
vfsErrNoFileSystem
vfsErrFileAlreadyExists
vfsErrBadName
vfsErrVolumeFull
vfsErrDirectoryNotFound
```

etc.

Comments Only creates the file and it is not opened.
Prepares the directory in advance before creating the file.

VFSFileOpen

Purpose OPEN the file or directory.

Prototype

```
Err VFSFileOpen(UInt16 volRefNum, const Char *pathNameP,
UInt16 openMode, FileRef *fileRefP)
```

Parameters

`-> volRefNum` Volume reference number.

`-> pathNameP` Absolute full path name of file or directory.
It must not be NULL.

`-> openMode` Specify open mode as follows (See `VFSMgr.h`),

```
#define vfsModeExclusive
                        // do not let anyone else open it
#define vfsModeRead   // open for read access
#define vfsModeWrite  // open for write access, implies
exclusive
#define vfsModeReadWrite
                        // open for read/write access
```

`<-  fileRefP` The opened file reference.

Result

```
errNone
expErrNotOpen
expErrCardReadOnly
vfsErrFileGeneric
vfsErrVolumeBadRef
vfsErrNoFileSystem
```

`vfsErrFilePermissionDenied`
File cannot be opened (because the same file has been opened in vfsModeExclusive, for instance).

`vfsErrFileNotFound`

`vfsErrBadName`

etc.

Comments `openMode` is applicable in the case of file open, but not directory open. `vfsErrFilePermissionDenied` will be returned in two cases.

1. If a file has already been opened with the `openMode` parameter set to `vfsModeExclusive`, and you try to open the same file.
2. If the file has already been opened with the `openMode` set to some value other than `vfsModeExclusive`, and you try to open the same file with `vfsModeExclusive`.

VFSFileClose

Purpose Closing a file or directory.

Prototype
```
Err VFSFileClose(FileRef fileRef)
```

Parameters
`-> fileRef` File reference number returned from `VFSFileOpen()`.

Result
`errNone`

`expErrNotOpen`

`vfErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

etc.

VFSFileReadData

Purpose Read the contents of the opened file to data storage based chunk (record or resource) in storage heap.

Prototype
```
Err VFSFileReadData(FileRef fileRef, UInt32 numBytes, void
*bufBaseP, UInt32 offset, UInt32 *numBytesReadP)
```

Parameters
`-> fileRef` File reference number returned from `VFSFileOpen`.

`-> numBytes` Number of read byte

`-> bufBaseP`	Pointer to destination chunk in storage heap for READ data. This must be a valid pointer that is returned by MemoryMgr. This must be the beginning of the chunk.
`-> offset`	Offset in bytes from destination buffer's base pointer (bufBaseP)
`<- numBytesReadP`	Pointer to the number of bytes actually read. If it is not necessary to get this value, set to NULL.

Result

`errNone`

`expErrNotOpen`

`vfsErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrFilePermissionDenied`
Forbidden access to File READ. When this value is returned, `openMode` is not appropriate.

`vfsErrFileEOF`

`vfsErrNoFileSystem`

`vfsErrIsADirectory`

etc.

Comments

When opening a file to read data, specify `'vfsModeRead'` or `'vfsModeReadWrite'` as `openMode`.
When internal filePointer reaches at the end of file (EOF), this API has read data until EOF and returns `'vfsErrFileEOF'`. If `'NumBytesReadP'` is non-NULL, this API returns the actual read bytes as a result.
This API is applicable to the file, not the directory.

VFSFileRead

Purpose

Read data from a file into a dynamic heap (or any writable memory).

Prototype

```
Err VFSFileRead (FileRef fileRef, UInt32 numBytes, void
*bufP, UInt32 *numBytesReadP)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> numBytes`	The number of bytes to read
`-> bufP`	Pointer to destination buffer in dynamic Heap for the READ data.

`<-  numBytesReadP`
Pointer to the number of bytes actually read.
If it is not necessary to get this value, set to NULL.

Result

`errNone`

`expErrNotOpen`

`vfsErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrFilePermissionDenied`
openMode is not appropriate.

`vfsErrFileEOF`

`vsfErrNoFileSystem`

`vfsErrIsADirectory`

etc.

Comments

When opening a file to read data, specify `'vfsModeRead'` or `'vfsModeReadWrite`" as `openMode`.
When internal filePointer reaches at the end of file (EOF), this API has read data until EOF and returns `'vfsErrFileEOF'`. If `'NumBytesReadP'` is non-NULL, this API returns the actual read bytes as a result.
This API applies to the file, not the directory.

VFSFileWrite

Purpose

WRITE data to an open file.

Prototype

```
Err VFSFileWrite(FileRef fileRef, UInt32 numBytes,
 const void *dataP, UInt32 *numBytesWrittenP)
```

Parameters

`-> fileRef` File reference number returned from `VFSFileOpen()`.

`-> numBytes` The number of bytes to write.

`-> dataP` Pointer to data to write.

`<-  numBytesWrittenP`
Set to the `number` of bytes actually written on return if non-NULL.
If it is not necessary to get this value, set NULL.

Result

`errNone`

`expErrNotOpen`

`expErrCardReadOnly`

`vfsErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrFilePermissionDenied`
Attributes of File are ReadOnly, or openMode is inappropriate.

`vfsErrNoFileSystem`

`vfsErrIsADirectory`

`vfsErrVolumeFull`

etc.

Comments When opening a file to write data to, specify either '`vfsModeWrite`' or '`vfsModeReadWrite`' as `openMode`.
This API applies to the file, not the directory.

VFSFileDelete

Purpose Delete a closed file or directory.

Prototype
```
Err VFSFileDelete(UInt16 volRefNum, const Char *pathNameP)
```

Parameters

`-> volRefNum`	Volume reference number returned from `VFSFileOpen()`.
`-> pathNameP`	Full path of the file or directory to be deleted

Result `errNone`

`expErrNotOpen`

`vfsErrFileGeneric`

`vfsErrFileStillOpen`

`vfsErrFilePermissionDenied`
Attributes of File are ReadOnly.

`vfsErrFileNotFound`

`vfsErrVolumeBadRef`

`vfsErrNoFileSystem`

`vfsErrDirNotEmpty`

`vfsErrBadName`

etc.

Comments When this API is called, the file or the directory to be deleted must be closed.

VFSFileRename

Purpose Rename a closed file or directory.

Prototype
```
ErrVFSFileRename(UInt16 volRefNum,const Char *pathNameP,
const Char*newNameP)
```

Parameters

`-> volRefNum`	Volume reference number.
`-> pathNameP`	Full path of the file or directory to be renamed.
`-> newNameP`	New file name only (not the full path).

Result

`errNone`

`expErrNotOpen`

`expErrCardReadOnly`

`vfsErrFileGeneric`

`vfsErrFileStillOpen`

`vfsErrFilePermissionDenied`
Attributes of file are ReadOnly.

`vfsErrFileAlreadyExists`

`vfsErrFileNotFound`

`vfsErrVolumeBadRef`

`vfsErrNoFileSystem`

`vfsErrBadName`

`vfsErrVolumeFull`

etc.

Comments When this API is called, the file or the directory to be renamed must be closed. Renaming a file does not change its directory where it is located.

Ex)

```
VFSFileRename(volRefNum, “/palm/programs/test”,
“rename”);
```
`“/palm/programs/test”` changes to `“/palm/programs/rename”`

VFSFileSeek

Purpose Set the position of file pointer within an open file.

Prototype
```
Err VFSFileSeek( FileRef fileRef, FileOrigin origin, Int32 offset)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> origin`	Origin to use when calculating new position from the offset Assign `FileOrigin` constant.
`-> offset`	Offset from the origin to set the new position in the file. It can be set as positive (forward) or negative (backward).

Result

`errNone`	
`sysErrParamErr`	Origin is improper.
`expErrNotOpen`	
`vfsErrFileBadRef`	
`vfsErrFileEOF`	
`vfsErrNoFileSystem`	
`vfsErrIsADirectory`	
etc.	

Comments When offset is set to a negative value and the result of the new position becomes negative, the actual position is the beginning of file.
When offset is set to a positive value and the result of the new position exceeds the end of the file, the actual position is the end of file.
This API is applied to the file, not the directory.

VFSFileEOF

Purpose Get the status of End-Of-File of an open file.

Prototype
```
Err VFSFileEOF( FileRef fileRef)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.

Result

`errNone`	
`expErrNotOpen`	
`vfsErrFileEOF`	
`vfsErrFileBadRef`	

`vfsErrNoFileSystem`

`vfsErrIsADirectory`

etc.

Comments This API is applied to the file, not the directory.

VFSFileTell

Purpose Get current position of the file pointer within an open file.

Prototype
```
Err VFSFileTell(FileRef fileRef, UInt32 *filePosP)
```

Parameters

->	`fileRef`	File reference number returned from `VFSFileOpen`.
<-	`filePosP`	Pointer to the present position of the file.

Result

`errNone`

`expErrNotOpen`

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

`vfsErrIsADirectory`

etc.

VFSFileAttributesGet

Purpose Obtain the file attributes of an open file or directory.

Prototype
```
Err VFSFileAttributesGet( FileRef fileRef, UInt32
*attributesP)
```

Parameters

->	`fileRef`	File reference number returned from `VFSFileOpen`.
<-	`attributesP`	Pointer to file or directory attributes (See `VFSMgr.h`) Obtained as `FileAttributes` constant.

Result

`errNone`

`expErrNotOpen`

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

etc.

VFSFileAttributesSet

Purpose Change the attributes of an open file or directory.

Prototype
```
Err VFSFileAttributesSet(FileRef fileRef, UInt32 attributes)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> attributes`	The file attribute to set to the file. (Refer to `VFSFileAttributesGet`)

Result

`errNone`	
`sysErrParam`	The specified attributes are inappropriate.
`expErrNotOpen`	
`expErrCardReadOnly`	
`vfsErrFileGeneric`	
`vfsErrFileBadRef`	
`vfsErrNoFileSystem`	
etc.	

Comments File attributes can be changed to read only, hidden, system, or archive attributes. However, this function should not be used to change directory (`fsAttribDirectory`) or volume label (`fsAttribVolumeLabel`) attributes. If it is necessary to change the directory or volume label attributes, use `VFSDirCreate` or `VFSVolumeSetLabel`.

VFSFileDateGet

Purpose Obtain the dates of an open file or directory.

Prototype
```
Err VFSFileDateGet(FileRef fileRef, UInt16 whichDate, UInt32
*dateP)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> whichDate`	Specifies which date to get. (See VFSMgr.h.) Assign `FileDate` constant.
`<- dateP`	Pointer to dates data. Date represented by seconds counting since 1/1/1904 represents date.

Result

`errNone`	
`sysErrParamErr`	The date is inappropriate.
`expErrNotOpen`	

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

etc.

VFSFileDateSet

Purpose Change the dates of an open file or directory.

Prototype
```
Err VFSFileDateSet( FileRef fileRef, UInt32 whichDate,
UInt32 date)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> whichDate`	Specifies which date to set. (See `VFSFileDateGet`.)
`-> date`	Contains the date to set. Represented by seconds counting since 1/1/1904.

Result

`errNone`	
`sysErrParamErr`	whichDate is inappropriate.
`expErrNotOpen`	
`vfsErrFileGeneric`	
`vfsErrFileBadRef`	
`vfsErrFilePermissionDenied`	
`File attribute is ReadOnly.`	
`vfsErrNoFileSystem`	

etc.

VFSFileSize

Purpose Obtain the size of an open file.

Prototype
```
Err VFSFileSize(FileRef fileRef, UInt32 *fileSizeP)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`<- fileSizeP`	Pointer to file size.

Result

`errNone`

`expErrNotOpen`

`vfsErrNoFileSystem`

`vfsErrFileBadRef`

`vfsErrIsADirectory`

etc.

Comments This API is applied to the file, not the directory.

VFSFileResize

Purpose Change the size of an open file.

Prototype
```
Err VFSFileResize(FileRef fileRef, UInt32 newSize)
```

Parameters

`-> fileRef`	File reference number returned from `VFSFileOpen`.
`-> newSize`	The desired new size of the file. The new size can be larger or smaller than the current file size.

Result

`errNone`

`expErrNotOpen`

`vfsErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrFilePernissionDenied`
File attribute is ReadOnly, or openMode is inappropriate.

`vfsErrNoFileSystem`

`vfsErrVolumeFull`

`vfsErrIsADirectory`

etc.

Comments Specifies `vfsFileWrite` or `vfsFileReadWrite` when opening the file.
This API is applied to the file, not the directory.
It doesn't have to execute `VFSfileResize` specifically because file resizes automatically when executing `VFSfileWrite`.

Directory APIs

VFSDirCreate

Purpose Create a new directory.

Prototype
```
Err VFSDirCreate(UInt16 volRefNum, const Char *dirNameP)
```

Parameters

`-> volRefNum`	Volume reference number.

	`-> dirNameP`	Full path to the directory to be created.

Result

`errNone`

`expErrNotOpen`

`expErrCardReadOnly`

`vfsErrFileGeneric`

`vfsErrFileAlreadyExists`

`vfsErrVolumeBadRef`

`vfsErrNoFileSystem`

`vfsErrVolumeFull`

`vfsErrBadName`

etc.

VFSDirEntryEnumerate

Purpose

Enumerate the entries in the given directory.

Prototype

```
Err VFSDirEntryEnumerate(FileRef dirRef, UInt32
*dirEntryIteratorP, FileInfoType *infoP)
```

Parameters

`-> dirRef`	Reference number returned from `VFSFileOpen`.
`<-> dirEntryIteratorP`	Set the Pointer to the last enumerated directory entry. The Pointer to the next directory entry is returned.
`<- infoP`	The pointer to the file information (`FileInfoType`) about directory entry, which is specified by `dirEntryInteratorP`.

Result

`errNone`	
`sysErrParamErr`	dirEntryP is invalid.

`expErrNotOpen`

`expErrEnumerationEmpty`

`vfsErrBufferOverflow`

`vfsErrFileGeneric`

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

`vfsErrNotADirectory`

etc.

Comments

Before using this API, the directory to be enumerated must be opened by `VFSFileOpen()` API.

`dirEntryIteratorP` is a variable to obtain the next directory entry. If the last enumerated directory entry is set and this API is called, the next directory entry is returned.

To obtain all directory entry, set `expIteratorStart` and call this API to get the first entry. After the first entry is called, call this API repeatedly by setting the returned value until the end of directory. If `expIeratorStop` is returned in this parameter, this means all directory entries of the specified directory have been enumerated.

Return value depends on the number of directory entries as follows,

- There is nothing under the specified directory
  Return value: `expErrEnumerationEmpty`
  `dirEntryIteratorP`: `expIteratorStop`
- There is one directory entry to be enumerated.
  Return value: `errNone`
  `dirEntryIteratorP`: `expIteratorStop`
- There are more than 2 directory entries to be enumerated.
  Return value: `errNone`
  `dirEntryIteratorP`: the reference to obtain the next directory entry.

When `infoP->name` is set to NULL and this API is called, only the attributes information is returned as `infoP->attributes`.

When infoP is set to NULL, valid data is not returned as a result.

This API is applied to the directory, not the file.

Below are example codes to enumerate directory entries.

```
FileInfoType  info;
UInt32 dirIterator = expIteratorStart;
FileRef dirRef;

VFSFileOpen(volRefNum, “/palm”, vfsModeRead, &dirRef);
while(dirIterator != expIteratorStop){
   if(VFSDirEntryEnumerate(dirRef, &dirIterator,
    &info)) {
     /* get 1 entry */
   } else {
     /* error */
   }
}
VFSFileClose(dirRef);
```

Volume APIs

VFSVolumeFormat

Purpose Format and mount the first volume in the specified slot.

Prototype
```
Err VFSVolumeFormat(UInt8 flags,  UInt16 fsLibRefNum,
VFSAnyMountParamPtr vfsMountParamP)
```

Parameters

`-> flags` Specifies format.

`-> fsLibRefNum` Specifies the library reference number of File system to format with.

`<-> VFSMountParamP` The pointer to `VFSAnyMountParamType`.

Result

`errNone`

`expErrUnsupportedOperation`

`expErrNotOpen`

`expErrNotEnoughPower`

`vfsErrVolumeStillMounted`

etc.

Comments When `flags` is set to 0, Slot Native File System is chosen to format the Memory Stick. In this case, `fsLibRefNum` is set to 0.

To specify the library reference number of file system to format with, flags is set to `vfsMountFlagsUseThisFileSystem` and `fsLibRefNum` is set to the number.

`VFSMountParamP` is set to casted pointer to `VFSSlotMountParamType` structure variable.

For instance, it is possible to implement Volume Format as follows:

1. Application should call `VFSVolumeInfo()` API to get `slotLibRefNum` and `slotRefNum`.
2. Set the argument of `VFSSlotMountParamType`.
3. Use the result of casting `vfsSlotMountParam` to `VFSAnyMountParamType` as the parameter of `VFSVolumeFormat()` API.

```
VolumeInfoType   volInfo;
VFSSlotMountParamType   sltMntPrm;
err = VFSVolumeInfo(volRefNum, &volInfo);
sltMntPrm.vfsMountParam.mountClass = sysFileTSlotDriver;
sltMntPrm.slotLibRefNum = volInfo.slotLibRefNum;
```

```
sltMntPrm.slotRefNum = volInfo.slotRefNum;
vfsVolumeFormat(0,0, (VFSAnyMountParamPtr)&sltMntPrm);
```

VFSVolumeEnumerate

Purpose Enumerate the volume that is mounted.

Prototype
```
Err VFSVolumeEnumerate(UInt16 *volRefNumP, UInt32
*volIteratorP)
```

Parameters

<-	volRefNumP	Pointer to volume reference number.
<->	volIteratorP	Specifies pointer to the last enumerated volume. The pointer to next volume is returned.

Result

errNone	
SysParamErr	volIteratorP is invalid.
expErrNotOpen	
expErrEnumerationEmpty	
vfsErrVolumeBadRef	
etc.	

Comments volIteratorP is the variable to enumerate the next volume. Set the last enumerated volume and call this API, the next volume is returned.
To enumerate all volume, at first, set volIteratorP to expIeratorStart and call this API, the first volume can be obtained. Subsequently, set the last obtained volume and call this API repeatedly untill expIteratorStop is returned by volIteratorP.

VFSVolumeInfo

Purpose Get information about the specified volume.

Prototype
```
Err VFSVolumeInfo(UInt16 volRefNum, VolumeInfoType
*volInfoP)
```

Parameters

->	volRefNum	Volume reference number.
<->	volInfoP	Pointer to volume information.

Result

errNone
expErrNotOpen
vfsErrVolumeBadRef

`vfsErrNoFileSystem`

etc.

Comments In Memory File System, `VolumeInfoType` is defined as follows.

```
volumeInfo.attributes = 1;
volumeInfo.fsType = fsFilesystemType_VFAT;
volumeInfo.fsCreator = 'MSfs';
volumeInfo.mountClass =sysFileTSlotDriver;
volumeInfo.slotLibRefNum = 5;
volumeInfo.slotNumer = 1;
volumeInfo.mediaType = ExpMediaType_MemoryStick;
```

VFSVolumeLabelGet

Purpose Obtain the label of the Specified Volume.

Prototype

```
Err VFSVolumeLabelGet(UInt16 volRefNum, Char *labelP, UInt16
bufLen)
```

Parameters

->	`volRefNum`	Volume reference number.
<->	`labelP`	Pointer to destination volume label.
->	`bufLen`	Specify the length of `labelP` buffer.

Result

`errNone`

`expErrNotOpen`

`vfsErrBufferOverflow`

`vfsErrVolumeBadRef`

`vfsErrNoFileSystem`

etc.

Comments `labelP` requires Minimum 12 bytes length.

VFSVolumeLabelSet

Purpose Set volume label

Prototype

```
VFSVolumeLabelSet(UInt16 volRefNum, const Char *labelP)
```

Parameters

->	`volRefNum`	Volume reference number.

	`-> labelP`	Desired volume label

Result

`errNone`
`expErrNotOpen`
`expErrCardReadOnly`
`vfsErrFileGeneric`
`vfsErrVolumeBadRef`
`vfsErrNoFileSystem`
`vfsErrBadName`
`vfsErrVolumeFull`
etc.

Comments For Volume label restrictions, See Name Specification.

VFSVolumeSize

Purpose Obtain the total amount of space in a volume and the amount of space that is currently used.

Prototype

```
Err VFSVolumeSize(UInt16 volRefNum, UInt32 *volumeUsedP,
UInt32 *volumeTotalP)
```

Parameters

	`-> volRefNum`	Volume reference number.
	`<-> volumeUsedP`	Pointer to the amount of used space on the volume
	`<-> volumeTotalP`	Pointer to the total amount of space on the volume.

Result

`errNone`
`expErrNotOpen`
`vfsErrVolumeBadRef`
`vfsErrNoFileSystem`
etc.

Comments Obtain the amount in bytes.

Utility APIs

VFSImportDatabaseFromFile

Purpose Import database to the storage heap from a file that is on the Memory Stick media.

Prototype
```
Err VFSImportDatabaseFromFile(UInt16 volRefNum, const Char
*pathNameP, UInt16 *cardNoP, LocalID *dbIDP)
```

Parameters

->	`volRefNum`	Volume reference number.
->	`pathNameP`	Pointer to absolute full path to a source file (Memory Stick).
<-	`cardNop`	On success, pointer to card number of new database on destination side (palm storage heap).
<-	`dbIDP`	On success, pointer to database's database ID.

Result

`errNone`	
`expErrNotOpen`	
`vfsErrVolumeBadRef`	
`vfsErrNoFileSystem`	
`vfsErrBadName`	File name of a source is improper.
etc.	

Comments Twice the space of the file itself is needed to import it to the storage heap. Palm OS 3.5 has such a restriction[1].

VFSExportDatabaseToFile

Purpose Export the specified database in storage heap to a file in memory in .prc or .pdb format.

Prototype
```
Err VFSExportDatabaseToFile(UInt16 volRefNum, const Char
*pathNameP, UInt16 cardNo, LocalID dbID)
```

Parameters

->	`volRefNum`	Volume reference number.
->	`pathNameP`	Pointer to absolute path to a file on destination side (Memory Stick™).
->	`cardNo`	Card number of the card where the specified source database is.

1. It will be improved at the next release of Palm OS®.

	`-> dbID`	Database ID of source (Palm device side) database.

Result

`errNone`
`expErrNotOpen`
`expErrCardReadOnly`
`vfsErrVolumeBadRef`
`vfsErrNoFileSystem`
`vfsErrBadName` File on destination side is improper.
`vfsErrVolumeFull`
etc.

VFSFileDBGetResource

Purpose Get the database resource of the specified .prc file on the Memory Stick.

Prototype

```
Err VFSFileDBGetResource(FileRef fileRef, DmResType type,
DmResID resID, MemHandle *resHP)
```

Parameters

	`-> fileRef`	.prc file reference returned from `VFSFileOpen`.
	`-> type`	Resource type.
	`-> resID`	Resource ID.
	`<- resHP`	Handle pointer to resource data.

Result

`errNone`
`expErrNotOpen`
`vfsErrFileBadRef`
`vfsErrNoFileSystem`
`memErrNotEnoughSpace`
(Not enough space left in memory to store required resource.)
`dmErrResourceNotFound`
(Specified file is not a resource database.)
`sysErrParamErr` (Argument is invalid. `ResHP` is NULL.)
etc.

Comments `resHP` occupies a certain amount of memory on the dynamic storage heap which is necessary to execute `MemHandleFree(resHP)` to release it after the function call.

VFSFileDBInfo

Purpose Get database information of .prc or .pdb file, which specified on the Memory Stick.

Prototype

```
VFSFileDBInfo(
FileRef  fileRef,
Char *nameP,
UInt16 *attributesP,
UInt16 *versionP,
UInt32 *crDateP,
UInt32 *modDateP,
UInt32 *bckUpDateP,
UInt32 *modNumP,
MemHandle *appInfoHP,
MemHandle *sortInfoHP,
UInt32 *typeP,
UInt32 *creatorP,
UInt16 *numRecordsP)
```

Parameters

-> fileRef	.prc or .pdb File reference returned from VFSFileOpen.
<-> nameP	Pointer to database name. Pass by 32 byte characters to the pointer. Assign NULL if it is unnecessary.
<-> attributesP	Pointer to database attributes flag. Assign NULL if it is unnecessary.
<-> versionP	Pointer to application version number. Assign NULL if it is unnecessary.
<-> crDateP	The date that the database was created. Date is seconds counting since 1/1/1904 represents date. Assign NULL if it is unnecessary.
<-> modDateP	Date that the last time databases change. Assign NULL if it is unnecessary.
<-> *appInfoHP	Pointer to handle of application information. Assign NULL if it is unnecessary.
<-> *sortInfoHP	Pointer to handle of soft table. Assign NULL if it is unnecessary.
<-> bckUpDateP	Database backup date. Assign NULL if it is unnecessary.
<-> modNumP	The number of times that the file has been modified, for instance, add or delete records. Assign NULL if it is unnecessary.

	`<-> typeP`	Pointer returns to database type. Assign NULL if it is unnecessary.
	`<-> creatorP`	Pointer returns to creator ID. Assign NULL if it is unnecessary.
	`<-> numRecordsP`	Pointer returns to record number that is inside of database. Assign NULL if it is unnecessary.

Result

`errNone`

`expErrNotOpen`

`memErrNotEnoughSpace`
(Not enough space left in memory for store database header.)

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

`vfsErrBadData`

etc.

Comments `appInfoHP` and `sortInfoHP` occupy a certain amount of memory on the dynamic storage heap which is necessary to execute `MemHandleFree(resHP)` to release it after the function call.

VFSFileDBGetRecord

Purpose Get record handle and its attribute, which is specified by index from .prc and .pdb files on Memory Stick media.

Prototype

```
Err VFSFileDBGetRecord(FileRef fileref, UInt16 recIndex,
 MemHandle *recHP, UInt8 *recAttrP, UInt32 *uniqueIDP)
```

Parameters

	`-> fileRef`	.prc or .pdb File reference returned from `VFSFileOpen`
	`-> recIndex`	Record index to be retrieved.
	`<- recHP`	Pointer returns handle to record. Assign NULL if it is unnecessary.
	`<- recAttrP`	Pointer to record attribute. Assign NULL if it is unnecessary.
	`<- uniqueIDP`	Pointer returns ID, which specific record. Assign NULL if it is unnecessary.

Result

`errNone`

`expErrNotOpen`

`memErrEnoughSpace`
(Not enough space left in memory for required record entry.)

`dmErrNotRecordDB`
the file contains no records.

`dmErrIndexOutOfRange`
(recIndex is stands outside the scope.)

`vfsErrFileBadRef`

`vfsErrNoFileSystem`

etc.

Comments `resHP` occupies a certain amount of memory on the dynamic storage heap which is necessary to execute `MemHandleFree(resHP)` to release it after the function call.

Expansion APIs

ExpCardPresent

Purpose Verify that the Memory Stick media exists in the specified slot.

Prototype
```
Err ExpCardPresent(UInt16 slotRefNumber)
```

Parameters
`-> slotRefNumber` Slot reference number.
This value can be obtained from `VFSVolumeInfo()`.

Result
`errNone` Memory Stick media exists in the specified slot.

`expErrInvalidSlotRefNumber`

`expErrSlotDeallocated`

`expErrNotOpen`

`expErrCardNotPresent`

etc.

ExpCardInfo

Purpose Obtain expansion card information

Prototype
```
Err ExpCardInfo(UInt16 slotRefNumber, ExpCardInfoType
*infoP)
```

Parameters
`-> slotRefNumber` Slot reference number.
This value can be obtained from `VFSVolumeInfo()`.

	`<-  InfoP`	Pointer to ExpCard information.

Result

`errNone`

`expErrNotOpen`

`expErrCardNotPresent`

`expErrInvalidSlotRefNumber`

`expErrSlotDeallocated`

etc

Comments Information about the Memory Stick media in the slot is returned.

If `capabilityFlags` is set to `expCapabilityHasStorage`, the following params are returned.

```
manufacturerStr[]:""
productStr[]:""
deviceClassStr[]:"Memory Stick"
deviceUniqueIDStr[]:""
```

ExpSlotEnumerate

Purpose Obtain expansion slot list.

Prototype

```
Err ExpSlotEnumerate(UInt16 *slotRefNumP, UInt16
*slotIteratorP)
```

Parameters

	`<-  slotRefNumP`	Pointer that returns slot reference number.
	`<-> slotIteratorP`	Pointer to the last slot. Returns the pointer to the next slot.

Result

`errNone`

`expErrInvalidSlotRefNumber`

`expErrSlotDeallocated`

`expErrNotOpen`

`expErrCardNotPresent`

`SysParamErr` (`slotIteratorP` is invalid)

etc

Comments `slotIteratorP` is a valiable used to obtain the next slot. Set the last slot obtained and call API to get the next one.

To obtain all slots, set `expIteratorStart` to `slotIteratorP` to call API for the first slot. Then, set a value returned from the API. Repeat to call this until `expIteratorStop` is returned to `slotIteratorP`.

Note

Determining If File System Is Available

To determine if the Memory Stick file system is available, check the presence of VFS Manager on Feature.

Here is the sample code.

```
UInt32 vfsMgrVersion;
err = FtrGet(sysFileCVFSMgr, vfsFtrIDVersion,
&vfsMgrVersion);
if (err){
   /* VFS Manager is present */;
} else {
   /* VFS Manager is NOT present */;
}
```

Part II: Library

7

High Resolution : Sony HR Library

The CLIÉ™ enables users to provide a 320 x 320 dot high-definition display that first appeared as a Palm platform device. Applications are able to display impressively detailed pictures.

Screen mode and API

Glossary

Compatibility mode This mode enables a 160 x 160 VRAM image to stretch twice its height and width to display on a 320 x 320 LCD panel.

High resolution mode This mode displays a 320 x 320 VRAM image as it is on the LCD panel and has two ways of drawing: high-resolution API and existing API. These two modes are usable at the same time.

Existing API Existing API is a drawing with API provided PalmOS 3.5. In drawing, since OS expands 320 x 320 of the image automatically, The application allows you to program with the existing API as before without recognizing the hardware difference. Moreover, the existing API makes it possible to draw beautiful characters according to newly adopted fonts for high-resolution. (Exception: The characters made by bitmap are displayed as usual.)

High resolution API High resolution API is API to make the best use of 320 x 320 resolution. It displays more characters with beautiful fonts than usual, detailed figures, and elaborated bitmap on 320 x 320 coordinate system.
High-resolution API isn't planned to achieve all the drawing function of conventional PalmOS but is planned as an assumption to use existing API for unrealized functions such as a form and a coordinate input using pens. Thus, in creating a high-resolution application, we recommend you to plan it by existing API as usual then replace only the parts which needed to be drawn elaborately with high-resolution API. This makes it easier to keep the compatibility of a source and a binary level action among existing models and

is the effective way to utilize the limited resource of Palm OS for realizing a beautiful drawing.

Incompatibility of existing API for High Resolution

On high-resolution mode, the existing application works with few modification. Here, explanations can be given about the difference with conventional device.

1. Font drawing

   The characters to the display (screen) window are drawn with resolution font. (This font is newly adopted for PEG-N700C/N710C so the visibility differs because the glyph is different from the font of PalmOS.) For the drawing of an off-screen window, the same font is used as usual when drawing characters with existing API.

The Object of API

`WinDrawChar`
`WinDrawChars`
`WinDrawInvertedChars`
`WinDrawTruncChars`
`WinEraseChars`
`WinInvertChars`
`WinPaintChar`
`WinPaintChars`

Correspodance

Existing Font			High Resolution Font	
`stdFont`	(fontID:0)	->	`hrStdFont`	(fontID:8)
`boldFont`	(fontID:1)	->	`hrBoldFont`	(fontID:9)
`largeFont`	(fontID:2)	->	`hrLargeFont`	(fontID:10)
`symbolFont`	(fontID:3)	->	`hrSymbolFont`	(fontID:11)
`symbol11Font`	(fontID:4)	->	`hrSymbol11Font`	(fontID:12)
`symbol7Font`	(fontID:5)	->	`hrSymbol7Font`	(fontID:13)
`ledFont`	(fontID:6)	->	`hrLedFont`	(fontID:14)
`largeBoldFont`	(fontID:7)	->	`hrLargeBoldFont`	(fontID:15)

2. Line drawing

   When the functions for a series of WinXXXLine are valued on the screen:
   In the direction of parallel and vertical line, the compatibility remains (The line is drawn of 2 pixel thickness).
   The direct line at slant is drawn of 1 pixel thickness.
   The line drawing for off-screen has not changed at all.

   The Object of API

`WinDrawGrayLine`
`WinDrawLine`
`WinEraseLine`
`WinFillLine`
`WinInvertLine`
`WinPaintLine`

3. Pattern drawing:

   On screen window, a resolution of pattern is doubled. (For GrayPattern, a visibility is different due to its resolution.)
   The Pattern drawing for off-screen has remained in the same condition.

   The Object of API

`WinDrawGrayLine`
`WinDrawGrayRectangleFrame`
`WinFillLine`
`WinFillRectangle`

4. Frame drawing

   When drawing a Frame with existing API on screen window, the thickness of the line may differ depend on the types of frame. The thickness of the line is thinner than before with RoundFrame, boldRoundFrame, and dialogFrame. (In case that the Frame is radius>2)
   The frame drawing for off screen window remains the same.

   The Object of API

`WinDrawRectangleFrame`
`WinDrawGrayRectangleFrame`

The Object of API

`WinEraseRectangleFrame`
`WinInvertRectangleFrame`
`WinPaintRectangleFrame`

5. Rounded Rectangle

   When drawing a rectangle, radius>2 with existing API, the corner is rounded because of the high resolution. However, drawing rectangle on the off screen window is unchanging as usual.

6. `WinCopyRectangle`

   On high-resolution mode, when an existing API is used, a screen window actually handles 320 x 320 resolution on the inside. So, when coping between screen window and off-screen window with existing API, the resolution has changed over. Thus, if copying from screen to off-screen to screen, for example, restoring a display might be difficult.
   So, use `WinSaveBits()` and `WinRestoreBits()` instead of `WinCopyRectangle` in performing these procedures.

   Details:

   – When copying: from the screen window to off-screen with an existing API, the information is reduced by one-forth of its conventional size.
   – From the off-screen window to screen with an existing API, the information enlarged 4 times of its size.
   – The resolution for copying between the screen windows and the off-screen windows has not changed at all.
   – For copying with API `HRWinCopyRectangle`, the conversion of resolution hasn't undergone.

7. The visibility of drawing may differ (Characters, diagonal lines and patterns) depending on the drawing directly on the screen and the drawing on the off-screen window first then it is copied to the screen window as the process from 1 to 6.
8. A `WinGetPixel` returns (top, left) pixel out of four pixels. (The compatibility will remaine only among an existing API.)
9. It takes more time to transfer the data and more memory because the display data has increased four times.
10. Forms and objects are controlled in the 160 x 160 coordinate system. In high-resolution, the image is stretched twice its height and width to display. The resource size to create is 160 x 160 at the most by Constructor provided Code Warrior for Palm Release 6.
11. If application font is used, the display may not work properly.
12. Applicaion, the likes of drawing on VRAM directly isn’t drawn correctly.

High Resolution and existing API

When drawing with existing API in high-resolution mode on the screen window, the drawing is doubled in the directions of X and Y-axes and written on the VRAM. For example, if drawn at axes (50, 70) with `WinDrawPixel`, a current foreground color is set on the pixel of VRAM at (100, 140), (101, 140), (100, 141), (101, 141). However, on high-resolution API, it's drawn with 320 x 320 resolution. A foreground color is set only on one pixel of VRAM at axes (50, 70) if drawn at axes (50, 70), with `HRWinDrawPixel` of high-resolution API.

This table shows corresponding high-resolution and existing APIs.
If there is a blank on high-resolution API line, use the existing API. If you handle the axes data with these indicated APIs, NOTE that the scale of them will be converted into the coordinate system of 160 x 160 even in the high-resolution mode. The coordinate system change applies only to the display window.
A high-resolution API with limitation is noted.

Table 7-1 High-resolution APIs for Window

Existing API	High-resolution API	Hand instruction for high-resolution API
`WinClipRectangle`	`HRWinClipRectangle`	
`WinCopyRectangle`	`HRWinCopyRectangle`	
`WinCreateBitmapWindow`	`HRWinCreateBitmapWindow`	
`WinCreateOffscreenWindow`	`HRWinCreateOffscreenWindow`	
`WinCreateWindow`	`HRWinCreateWindow`	Bounds setting is limited.
`WinDeleteWindow`		
`WinDisplayToWindowPt`	`HRWinDisplayToWindowPt`	
`WinDrawBitmap`	`HRWinDrawBitmap`	
`WinDrawChar`	`HRWinDrawChar`	See "Font setting".
`WinDrawChars`	`HRWinDrawChars`	See "Font setting".
`WinDrawGrayLine`	`HRWinDrawGrayLine`	
`WinDrawGrayRectangleFrame`	`HRWinDrawGrayRectangleFrame`	
`WinDrawInvertedChars`	`HRWinDrawInvertedChars`	See "Font setting".
`WinDrawLine`	`HRWinDrawLine`	
`WinDrawPixel`	`HRWinDrawPixel`	

Table 7-1 High-resolution APIs for Window

WinDrawRectangle	HRWinDrawRectangle	
WinDrawRectangleFrame	HRWinDrawRectangleFrame	
WinDrawTruncChars	HRWinDrawTruncChars	See "Font setting".
WinEraseChars	HRWinEraseChars	See "Font setting".
WinEraseLine	HRWinEraseLine	
WinErasePixel	HRWinErasePixel	
WinEraseRectangle	HRWinEraseRectangle	
WinEraseRectangleFrame	HRWinEraseRectangleFrame	
WinEraseWindow		
WinFillLine	HRWinFillLine	
WinFillRectangle	HRWinFillRectangle	
WinGetActiveWindow		
WinGetBitmap		
WinGetClip	HRWinGetClip	
WinGetDisplayExtent	HRWinGetDisplayExtent	
WinGetDisplayWindow		
WinGetDrawWindow		
WinGetFirstWindow		
WinGetFramesRectangle	HRWinGetFramesRectangle	
WinGetPattern		
WinGetPatternType		
WinGetPixel	HRWinGetPixel	
WinGetWindowBounds	HRWinGetWindowBounds	
WinGetWindowExtent	HRWinGetWindowExtent	
WinGetWindowFrameRect	HRWinGetWindowFrameRect	
WinIndexToRGB		
WinInvertChars	HRWinInvertChars	See "Font setting".

Table 7-1 High-resolution APIs for Window

WinInvertLine	HRWinInvertLine	
WinInvertPixel	HRWinInvertPixel	
WinInvertRectangle	HRWinInvertRectangle	
WinInvertRectangleFrame	HRWinInvertRectangleFrame	
WinModal		
WinPaintBitmap	HRWinPaintBitmap	
WinPaintChar	HRWinPaintChar	See “Font setting”.
WinPaintChars	HRWinPaintChars	See “Font setting”.
WinPaintLine	HRWinPaintLine	
WinPaintLines	HRWinPaintLines	
WinPaintPixel	HRWinPaintPixel	
WinPaintPixels	HRWinPaintPixels	
WinPaintRectangle	HRWinPaintRectangle	
WinPaintRectangleFrame	HRWinPaintRectangleFrame	
WinPalette		
WinPopDrawState		
WinPushDrawState		
WinResetClip		
WinRestoreBits	HRWinRestoreBits	
WinRGBToIndex		
WinSaveBits	HRWinSaveBits	
WinScreenLock		
WinScreenMode	HRWinScreenMode	Use to switch between compatibility and high-resolution modes.
WinScreenUnlock		
WinScrollRectangle	HRWinScrollRectangle	
WinSetActiveWindow		

Table 7-1 High-resolution APIs for Window

WinSetBackColor		
WinSetClip	HRWinSetClip	Clipping rectangle setting is limited.
WinSetDrawMode		
WinSetDrawWindow		
WinSetForeColor		
WinSetPattern		
WinSetPatternType		
WinSetTextColor		
WinSetUnderlineMode		
WinSetWindowBounds	HRWinSetWindowBounds	Bounding rectangles setting is limited.
WinValidateHandle		
WinWindowToDisplayPt	HRWinWindowToDisplayPt	

Table 7-2 High-resolution API for Bitmap

Existing API	High-resolution API	Handling instruction for high-resolution API
BmpBitsSize	HRBmpBitsSize	
BmpColortableSize		
BmpCompress		Bitmap that exceeds 160 x 160 x 8 bit is not supported.
BmpCreate	HRBmpCreate	
BmpDelete		
BmpGetBits		
BmpGetColortable		
BmpSize	HRBmpSize	

Table 7-3 High-resolution API for Font

Existing API	High-resolution API	Handling instruction for high-resolution API
FntAverageCharWidth		
FntBaseLine		
FntCharHeight		
FntCharsInWidth		
FntCharsWidth		
FntCharWidth		
FntDefineFont		
FntDescenderHeight		
FntGetFont	HRFntGetFont	
FntGetFontPtr		
FntGetScrollValue		
FntLineHeight		
FntLineWidth		
FntSetFont	HRFntSetFont	
FntWidthToOffset		
FntWordWrap		
FntWordWrapReverseNLines		
FontSelect	HRFontSelect	

This table shows compatality of high-resolution and existing APIs with these models.

Table 7-4 compatality of high-resolution and existing APIs with these model

	High-resolution API	Existing API
Conventional model	NG (Fatal Error)	OK
High-resolution support model In compatibility mode	`HRWinScreenMode` : OK The other APIs : NG (Fatal Error)	OK
High-resolution support model In high-resolution mode	OK	OK (Enables distinct character display.)

Font setting

With an existing API, the fonts shown below are available. When any of these fonts is used, the system internally doubles its resolution and allows clear character display.

Table 7-5 FontID

Name	FontID
`stdFont`	`0`
`boldFont`	`1`
`largeFont`	`2`
`symbolFont`	`3`
`symbol11Font`	`4`
`symbol7Font`	`5`
`ledFont`	`6`
`largeBoldFont`	`7`

With a high-resolution API, in addition to those shown above, 8 fonts are also usable. To specify 16 kinds of fonts in high-resolution mode, HRFontID type is defined instead of the existing FontID type.

Table 7-6 HRFontID

Name	HRFontID	Remark
`hrTinyFont`	`0`	`stdFont`
`hrTinyBoldFont`	`1`	`boldFont`
`hrSmallFont`	`2`	`largeFont`

Table 7-6 HRFontID

hrSmallSymbolFont	3	symbolFont
hrSmallSymbol11Font	4	symbol11Font
hrSmallSymbol7Font	5	symbol7Font
hrSmallLedFont	6	ledFont
hrSmallBoldFont	7	largeBoldFont
hrStdFont	8	
hrBoldFont	9	
hrLargeFont	10	
hrSymbolFont	11	
hrSymbol11Font	12	
hrSymbol7Font	13	
HrLedFont	14	
HrLargeBoldFont	15	

High-resolution API displays the text in the original size of a specified font. Here is an example: When a chinese character was viewed on the display of 320 x 320 with its font set to `hrTinyFont(= stdFont)`, its size will be 8 x 8 pixels (a quarter of the conventional size). To display the character in the same size as the one on the conventional device, the font should be set to one of these: `HRFontID` 8 to 15.
To set a font or to get a specified font on high-resolution mode, these are used:

```
HRFont  HRFntGetFont( Ulnt16 refNum )
HRFont  HRFntSetFont( Ulnt16 refNum, HRFontID font )
```

When the text is to be displayed using existing API with the font set to one of these (`HRFontID8` to `15`), the actual font will be `hrStdFont(HRFontID=8)`.
Palm OS does not associate plotting commands with plotting attributes. For example, when font is set to `hrLargeBoldFont(HRFontID = 15)` in high-resolution mode and the text is plotted first with a high-resolution API (such as `HRWinDrawChars`) and then with an existing API (such as `WinDrawChars`), the font will be hrStdFont(`HRFontID=8`).
Thus, you should first set a font using `HRFntSetFont` to plot text with high-resolution API. And to plot a character with an existing API, reset a font using `FntSetFont` .
As for the API that gets width and height of a font, an existing API can be used also on high-resolution mode. When plotting is done with high-resolution API, the font size will be the one that corresponds to a 320 x 320 coordinate system; with existing API, it will be the one that corresponds to a 160 x 160 coordinate system.

Drawing on an off-screen window in high-resolution mode

Display on screen and off-screen windows

With an existing API, a screen window has a bitmap of 160 x 160. However, it will be 320 x 320 in actual use. On the other hand, an off-screen window will have a bitmap of the specified size.

For example,
the off-screen window defined as this using existing API will have a bitmap of 160 x 160:

```
winH = WinCreateOffscreenWindow(160, 160, genericFormat,
&error);
```

And the one defined as this using high-resolution API will have a bitmap of 320 x 320:

```
winH = HRWinCreateOffscreenWindow(refNum, 320, 320,
genericFormat, &error);
```

With exisitng API, there might be a differnce between the drawing in screen window and off-screen window. With high-resolution API, there will be no difference.

Drawing characters

The display in the screen window will be as shown in Figure 7-1. Characters in the left column are drawn using the existing API `WinDrawChars`; those in the right are drawn with the high-resolution API `HRWinDrawChars`.

Figure 7-1

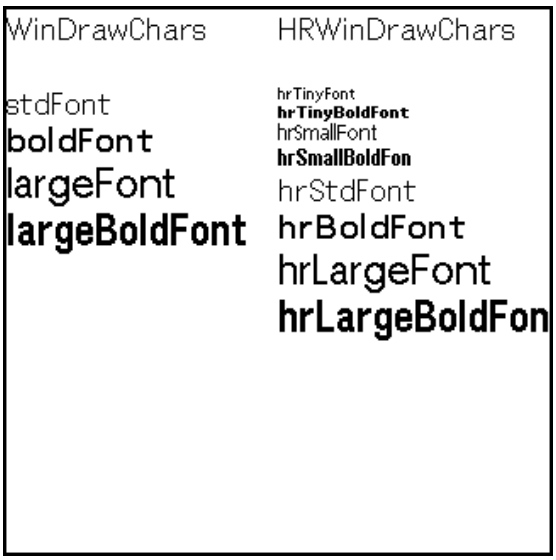
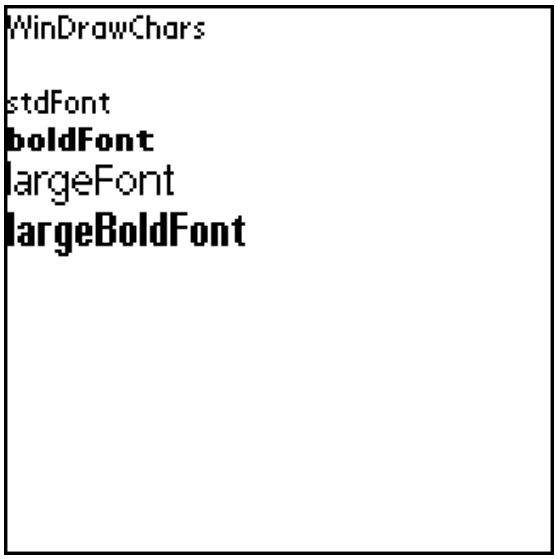


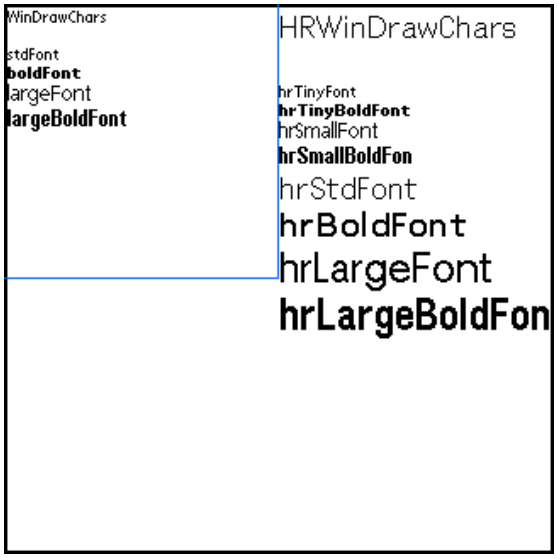
Figure 7-2 shows the off-screen window (the area of 160 x 160 outlined in blue) copied to the screen window using `HRCopyRectangle`.

Figure 7-2



When the area outlined in blue (Figure 7-2) is copied to a screen window using `WinCopyRectangle`, the display will be as shown in Figure 7-3.

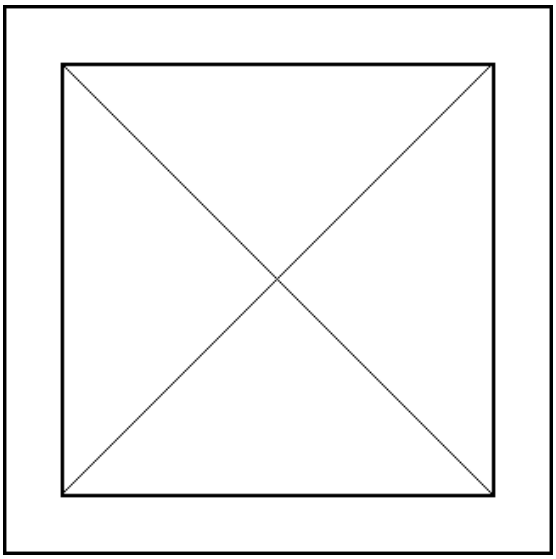
Figure 7-3



Drawing lines

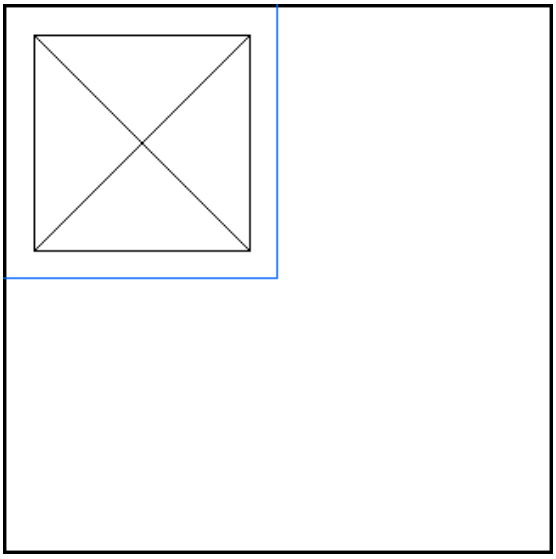
Figure 7-4 shows the figure consisting of six staight lines drawn with the existing API, `WinDrawLine`.

Figure 7-4



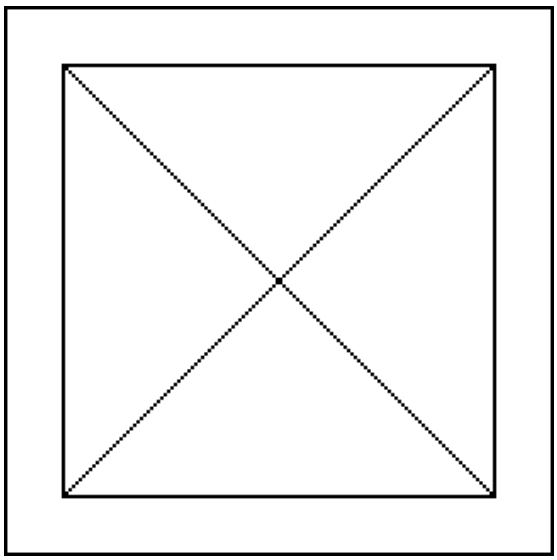
Same image as Figure 7-4 is drawn on the off-screen window as below.

Figure 7-5



The image of 160 x 160 bounds shown in the off-screen window is copied to the screen window with WinCopyRectangle as follow. In this case, the difference is the thickness of the diagonal line between direct drawing on screen window and copying.

Figure 7-6



Using High resolution API

Library loading

For a high-resolution support device, the library provides high-resolution API. To utilize the library, use `SysLibFind` to get reference No.

Example is shown below.[1]

```
#include <SonyCLIE.h>

SonySysFtrSysInfoP sonySysFtrSysInfoP;
Err error = 0;
Err status = 0;
UInt16 refNum;

if ((error = FtrGet(sonySysFtrCreator,
      sonySysFtrNumSysInfoP, (UInt32*)&sonySysFtrSysInfoP))) {
```

1. In this example the device is checked whether it's the CLIÉ™ ,however there is no guarantee that the CLIÉ™ has long been the only device that is accessible to the High-resolution.

```
   /* Not CLIE: maybe not available */
} else {
   if (sonySysFtrSysInfoP->libr & sonySysFtrSysInfoLibrHR) {
      /* HR available */
      if ((error = SysLibFind(sonySysLibNameHR, &refNum))){
         if (error == sysErrLibNotFound) {
            /* couldn't find lib */
            error = SysLibLoad( 'libr', sonySysFileCHRLib, &refNum );
         }
      }

      if (!error ) {
         /* Now we can use HR lib */
         ...
      }
   }
}
```

Any API is accessible using a reference No. obtained by `SysLibFind` or `SysLibLoad`. Only high-resolution support devices can get reference No. of "SonyHRLib". Without reference No., you cannot utilize the high-resolution API. In that case, the display will be in compatibility mode.

To start using the high-resolution API, an application should call the function `HROpen`; To exit, it should call `HRClose`.

Switching screen mode

An application programmed only with an existing API works in compatibility mode. To use the high-resolution mode, application needs to change the mode actively.
The two modes, the compatibility mode and the high-resolution mode, can be changed by `HRWinScreenMode()` API.
The comparison with `WinScreenMode()` API and the situation of a change in the screen mode by `HRWinScreenMode()` API are shown below.

Table 7-7 operation : `winScreenModeSet`

	`WinScreenMode`		`HRWinScreenMode`	
	width:160 height:160	width:320 height:320	width:160 height:160	width:320 height:320
compatibility mode	compatibility mode	invalid	compatibility mode	compatibility mode -> high-resolution mode
high-resolution mode	high-resolution mode	invalid	high-resolution mode -> compatibility mode	high-resolution mode

Table 7-8 operation : `winScreenModeSetToDefaults`

	`WinScreenMode`	`HRWinScreenMode`
compatibility mode	compatibility mode	compatibility mode
high-resolution mode	high-resolution mode -> compatibility mode	high-resolution mode

For the application that runs in high-resolution mode, set to high-resolution mode at its startup and reset to default at exit.
When another application is to be started (using `SysAppLaunch`) when a current application is running in high-resolution mode, you need to reset the mode to default. Set back to high-resolution mode again when later launched application is no longer in use.

In addition, a screen is cleared in the case of a mode change.
Examples are shown below.

Example 1: Switching from compatibility to high-resolution mode

```
#include <SonyCLIE.h>

Err    error;
UInt16 refNum;
UInt32 width, height, depth;

/*****************************************/
/*  Gets refNumof SonyHRLib              */
/*  Refer to section 5 for details.   */
/*****************************************/
/*****************************************/
```

```
/*  Executes Open library.           */
/*****************************************/
error = HROpen(refNum);
if (error) {
   /* error processing */
} else {
   width ÅÅ hrWidth; height = hrHeight;
   depth = 8;  /* (in color mode of 256 colors) */
   error = HRWinScreenMode ( refNum, winScreenModeSet,
    &width, &height, &depth, NULL );
   If ( error != errNone ){
      /* Screen mode remains unchanged. */
      - - - - - - - - - - -
   } else {
      /* high-resolution mode */
      - - - - - - - - - - -
   }
}
```

Example 2: Switching from high-resolution mode to default screen mode/ closing library

```
error = HRWinScreenMode ( refNum,
winScreenModeSetToDefaults, NULL, NULL, NULL, NULL );
if ( error != errNone ){
   /* Screen mode remains unchanged. */
   - - - - - - - - - - -
} else {
   /* Switched to default screen mode. */
   - - - - - - - - - - -
}
/*****************************************/
/*  Executes Close library.      */
/*****************************************/
error = HRClose(refNum);
```

High-Resolution API

System API

HROpen

Purpose Start to use high-resolution library.
Set plotting mode to high-resolution mode.

Prototype
```
Err HROpen ( UInt16 refNum)
```

Parameters

`-> refNum`	Reference number of high-resolution library.

Result

`errNone`	No error
`hrErrNoFeature`	High-resolution mode is not supported.
`memErrNotEnoughSpace`	Memory is insufficient.

Comments Handles the process to enables the use of high-resolution library.

HRClose

Purpose End an use of high-resolution library.

Prototype
```
Err HRClose ( UInt16 refNum)
```

Parameters

`-> refNum`	Reference number of high-resolution library

Result

`errNone`	No error
`hrErrNotOpen`	High-resolution library is not Open.
`hrStillOpen`	High-resolution library is still Open.

Comments Handles the process to end the use of high-resolution library.

HRGetAPIVersion

Purpose Get a version of high-resolution API.

Prototype
```
Err HRGetAPIVersion( UInt16 refNum, UInt16 *versionP )
```

Parameters

`-> refNum`	Reference number of high-resolution library

	`<-> versionP`	Pointer to a memory that stores API version.
Result	`errNone`	No error.
	`hrErrNotOpen`	High-resolution library is not Open.
	`hrErrParam`	Parameter error (versionP is NULL.)

Comments Obtains a version of high-resolution API.

version

15							8	7							0
Major Version								Minor Version							

Window API

HRWinClipRectangle

Purpose Clip a specified rectangular frame to clipping region in current draw window.

Prototype
```
void HRWinClipRectangle(UInt16 refNum, RectangleType *rP)
```

Parameters	`-> refNum`	Reference number of high-resolution library
	`<-> rP`	Pointer to a structure of a specified rectangular frame. Passed rectangle will be returned with it fitted into clipping region in the draw window.

Result Returns nothing.

HRWinCopyRectangle

Purpose Copy a rectangular region from one place to another.

Prototype
```
void HRWinCopyRectangle ( UInt16 refNum, WinHandle srcWin,
WinHandle dstWin, RectangleType *srcRect, Coord destX, Coord
destY, WinDrawOperation mode)
```

Parameters	`-> refNum`	Reference number of high-resolution library
	`-> srcWin`	Window from which the rectangle is copied. When NULL, this will be the draw window.
	`-> dstWin`	Window to which the rectangle is copied. When NULL, this will be the draw window.
	`-> srcRect`	Bounds of the region to copy.

`-> destX`	Top bound of the rectangle in destination window.
`-> destY`	Left bound of the rectangle in destination window.
`-> mode`	The method of transfer from the source to the destination window.

Result Returns nothing.

HRWinCreateBitmapWindow

Purpose Create a new off-screen window.

Prototype
```
WinHandle HRWinCreateBitmapWindow ( UInt16 refNum,
BitmapType *bitmapP, UInt16 *error )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> bitmapP`	Pointer to the bitmap which wil be associated to this window.
`<- error`	Pointer to any error this function encounters.

Result If no error, returns the handle of the new window. In case of error, returns NULL. One of the followings will be stored to errorParameter.

`errNone`	No error
`sysErrParamErr`	`bitmapP` Parameter is invalid. Bitmap should be uncompressed and its pixel size be valid(1,2,4,8). Screen bitmap is unacceptable.
`sysErrNoFreeResource`	Memory is insufficient to store new window structure.

HRWinCreateOffscreenWindow

Purpose Create a new off-screen window and add it to the window list.

Prototype
```
WinHandle HRWinCreateOffscreenWindow ( UInt16 refNum,
 Coord width, Coord height, WindowFormatType format,
 UInt16 *error )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> width`	Width of the window.
`-> height`	Height of the window.
`-> format`	Either screenFormat or genericFormat For an off-screen window, genericFormat is generally used.

	`<- error`	Pointer to any error this function encounters.

Result If no error, returns a new handle of the new window. In case of error, returns NULL.

ErrorParameter stores one of the followings.

`errNone`	No error
`sysErrParamErr`	Either width or height parameter is NULL; current color palette is invalid.
`SysErrNoFreeResource`	Memory is insufficient to execute this function.
`memErrNotEnoughSpace`	Memory is insufficient to execute this function.

HRWinCreateWindow

Purpose Create a new window and register it to the window list.

Prototype
```
WinHandle HRWinCreateWindow ( UInt16 refNum, RectangleType
*bounds, FrameType frame, Boolean modal, Boolean focusable,
UInt16 *error )
```

Parameters

	`-> refNum`	Reference number of high-resolution library
	`-> bounds`	Display relative bounds of the window. Every element of `bounds (topleft.x,topleft.y,extent.x,extent.y)` should be multiple of 2.
	`-> frame`	Type of frame around the window.
	`-> modal`	TRUE if the window is modal.
	`-> focusable`	TRUE if the window can be the active window.
	`<- error`	Pointer to any error encountered by this function.

Result Returns a handle for the new window. In case of error, returns NULL.

HRWinDisplayToWindowPt

Purpose Convert a display-relative coordinate to a window-relative coordinate. The coordinate returned is relative to the display window.

Prototype
```
void HRWinDisplayToWindowPt ( UInt16 refNum, Coord *extentX,
Coord *extentY )
```

Parameters

	`-> refNum`	Reference number of high-resolution library.

	`<-> extentX`	Pointer to x coordinate to convert.
	`<-> extentY`	Pointer to y coordinate to convert.

Result Returns nothing.

HRWinDrawBitmap

Purpose Draw a bitmap at the specified point in winPaint mode.

Prototype
```
void HRWinDrawBitmap ( UInt16 refNum, BitmapPtr bitmap,
Coord x, Coord Y )
```

Parameters

	`-> refNum`	Reference number of high-resolution library
	`-> bitmap`	Pointer to a bitmap
	`-> x`	The x coordinate of the top-left corner.
	`-> y`	The y coordinate of the top-left corner.

Result Returns nothing.

HRWinDrawChar

Purpose Draw the specified character in the draw window.

Prototype
```
void HRWinDrawChar ( UInt16 refNum, WChar theChar, Coord x,
Coord Y )
```

Parameters

	`-> refNum`	Reference number of high-resolution library
	`-> theChar`	The character to draw.
	`-> x`	x coordinate of the location where the character should be drawn (Left bound).
	`-> y`	y coordinate of the location where the character should be drawn (Left bound).

Result Returns nothing.

HRWinDrawChars

Purpose Draw the specified characters in the draw window.

Prototype
```
void HRWinDrawChars ( UInt16 refNum, const Char *chars,
Int16 len, Coord x, Coord y )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> chars`	Pointer to the characters to draw.
`-> len`	Length in bytes of the characters to draw.
`-> x`	x coordinate(left bound) of the first character to draw.
`-> y`	y coordinate (top bound) of the first character to draw.

Result Returns nothing.

HRWinDrawGrayLine

Purpose Draw a dotted line in the draw window.

Prototype
```
void HRWinDrawGrayLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> x1`	x coordinate of the start of the line.
`-> y1`	y coordinate of the start of the line.
`-> x2`	x coordinate of the end of the line.
`-> y2`	y coordinate of the end of the line.

Result Returns nothing.

HRWinDrawGrayRectangleFrame

Purpose Draw a gray rectangular frame in the draw window.

Prototype
```
void HRWinDrawGrayRectangleFrame ( UInt16 refNum, FrameType
frame, RectangleType *rP )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> frame`	Type of frame to draw.

	`-> rP`	Pointer to the rectangle to frame.

Result Returns nothing.

HRWinDrawInvertedChars

Purpose Draw the specified characters inverted (background color) in the draw window.

Prototype
```
void HRWinDrawInvertedChars ( UInt16 refNum, const Char
*chars, Int16 len, Coord x, Coord y )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> chars`	Pointer to the characters to draw.
`-> x`	x coordinate (left bound) of first charater to draw
`-> y`	y coordinate (top bound) of first charater to draw.

Result Returns nothing.

HRWinDrawLine

Purpose Draw a line in the draw window using current foreground color.

Prototype
```
void HRWinDrawLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> x1`	x coordinate of the start of the line.
`-> y1`	y coordinate of the start of the line.
`-> x2`	x coordinate of the end of the line.
`-> y2`	y coordinate of the end of the line.

Result Returns nothing.

HRWinDrawPixel

Purpose Draw a pixel in the draw window using current foreground color.

Prototype
```
void HRWinDrawPixel ( UInt16 refNum, Coord x, Coord y )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> x`	x coordinate of pixel.

	`-> y`	y coordinate of pixel.

Result Returns nothing.

HRWinDrawRectangle

Purpose Draw a rectangle in the draw window using current foreground color.

Prototype
```
void HRWinDrawRectangle ( UInt16 refNum, RectangleType *rP,
UInt16 cornerDiam )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> rP`	Pointer to the rectangle to draw.
`-> cornerDiam`	Diameter of corners.

Zero for square corners.

Result Returns nothing.

HRWinDrawRectangleFrame

Purpose Draw a rectangular frame in the draw window using current foreground color.

Prototype
```
void HRWinDrawRectangleFrame ( UInt16 refNum, FrameType
frame, RectangleType *rP)
```

Parameters

`-> refNum`	Reference number of high-resolution library.
`-> frame`	Type of frame to draw.
`-> rP`	Pointer to the rectangle to frame.

Result Returns nothing.

HRWinDrawTruncChars

Purpose Draw the specified characters in the draw window, truncating the characters to the specified width.

Prototype
```
void HRWinDrawTruncChars ( UInt16 refNum, const Char *chars,
Int16 len, Coord x, Coord y, Coord maxWidth )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> chars`	Pointer to the characters to draw.
`-> len`	Length in bytes of the characters to draw.

`-> x`	x coordinate of first character to draw (left bound).
`-> y`	y coordinate of first character to draw (top bound).
`-> maxWidth`	Maximum width of the characters that are to be drawn.

Result Returns nothing.

HRWinEraseChars

Purpose Erase specified characters in the draw window.

Prototype
```
void HRWinEraseChars ( UInt16 refNum, const Char *chars,
Int16 len, Coord x, Coord y )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> chars`	Pointer to the charcters to erase.
`-> len`	Length of the characters to erase.
`-> x`	x coordinate of first character to erase (left bound).
`-> y`	y coordinate of first character to erase (top bound).

Result Returns nothing.

HRWinEraseLine

Purpose Erase a line in the draw window using current background color.

Prototype
```
void HRWinEraseLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	Reference number of high-resolution library.
`-> x1`	x coordinate of the start of the line.
`-> y1`	y coordinate of the start of the line.
`-> x2`	x coordinate of the end of the line.
`-> y2`	y coordinate of the end of the line.

Result Returns nothing.

HRWinErasePixel

Purpose Erase a pixel in the draw window using current background color.

Prototype
```
void HRWinErasePixel ( UInt16 refNum, Coord x, Coord y )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> x`	x coordinate of a pixel.
`-> y`	y coordinate of a pixel.

Result Returns nothing.

HRWinEraseRectangle

Purpose Erase a rectangle in the draw window using current background color.

Prototype
```
void HRWinEraseRectangle ( UInt16 refNum, RectangleType *rP,
UInt16 cornerDiam )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> rP`	Pointer to the rectangle to erase.
`-> cornerDiam`	Diameter of corners; zero for square corners.

Result Returns nothing.

HRWinEraseRectangleFrame

Purpose Erase a rectangle in the draw window using current background color.

Prototype
```
void HRWinEraseRectangleFrame ( UInt16 refNum, FrameType
frame, RectangleType *rP )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> frame`	Type of frame to erase.
`-> rP`	Pointer to the rectangular frame.

Result Returns nothing.

HRWinFillLine

Purpose Fill a line in the draw window with the current pattern.

Prototype
```
void HRWinFillLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> x1`	x coordinate of the start of the line.
`-> y1`	y coordinate of the start of the line.
`-> x2`	x coordinate of the end of the line.
`-> y2`	y coordinate of the end of the line.

Result Returns nothing.

HRWinFillRectangle

Purpose Draw a rectanble with current pattern in the draw window.

Prototype
```
void HRWinFillRectangle ( UInt16 refNum, RectangleType *rP,
UInt16 cornerDiam )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`-> rP`	Pointer to the rectangle to draw.
`-> cornerDiam`	Diameter of corners; Zero for square corners.

Result Returns nothing.

HRWinGetClip

Purpose Return the clipping rectangle of the draw window.

Prototype
```
void HRWinGetClip ( UInt16 refNum, RectangleType *rP )
```

Parameters

`-> refNum`	Reference number of high-resolution library
`<- rP`	Pointer to a structure to hold the clipping bounds.

Result Returns nothing.

HRWinGetDisplayExtent

Purpose Return the width and height of the display (the screen).

Prototype
```
void HRWinGetDisplayExtent ( UInt16 refNum, Coord *extentX,
Coord *extentY )
```

Parameters

->	refNum	Reference number of high-resolution library
<-	extentX	Width of the display window.
<-	extentY	Height of the display window.

Result Returns nothing.

HRWinGetFramesRectangle

Purpose Return the region needed to draw a rectangle with a frame.

Prototype
```
void HRWinGetFramesRectangle ( UInt16 refNum, FrameType
frame, RectangleType *rP, RectangleType *obscuredRectP )
```

Parameters

->	refNum	Reference number of high-resolution library.
->	frame	Type of frame.
->	rP	Pointer to the rectangle to frame.
<-	obscuredRectP	Pointer to the rectangle obscured by the frame.

Result Returns nothing.

HRWinGetPixel

Purpose Return the current pixel color in the draw window.

Prototype
```
IndexedColorType HRWinGetPixel ( UInt16 refNum, Coord x,
Coord y )
```

Parameters

->	refNum	Reference number of high-resolution library
->	x	x coordinate of a pixel
->	y	y coordinate of a pixel

Result Returns the index color value of the pixel.

HRWinGetWindowBounds

Purpose Return the bounds of the current draw window in display-relative coordinates.

Prototype
```
void HRWinGetWindowsBounds ( UInt16 refNum, RectangleType
*rP )
```

Parameters

->	refNum	High-resolution library reference number.
<-	rP	Pointer to rectangle.

Result Returns nothing

HRWinGetWindowExtent

Purpose Returns the width and height of the current draw window.

Prototype
```
void HRWinGetWindowExtent ( UInt16 refNum, Coord *extentX,
Coord *extentY )
```

Parameters

->	refNum	High-resolution library reference number.
<-	extentX	Pointer to the width in pixels of the draw window.
<-	extentY	Pointer to the height in pixels of the draw window.

Result Returns nothing

HRWinGetWindowFrameRect

Purpose Returns a rectangle, in display –relative coordinates that defines the size and location of the window and its frame.

Prototype
```
void HRWinGetWindowFrameRect ( UInt16 refNum, WinHandle
winHandle, RectangleType *rP )
```

Parameters

->	refNum	High-resolution library reference number.
->	winHandle	Handle of window whose coordinates are desired.
<-	rP	A pointer to the coordinates of the window.

Result Returns nothing

HRWinInvertChars

Purpose Invert the specified characters in the draw window.

Prototype
```
void HRWinInvertChars ( UInt16 refNum, const Chars *chars,
Int16 len, Coord x, Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> chars`	Pointer to characters to invert.
`-> len`	Length in bytes of the characters to invert.
`-> x`	X coordinate of the first character to invert (left bound)
`-> y`	Y coordinate of the first character to invert (top bound)

Result Returns nothing

HRWinInvertLine

Purpose Inverts a line in the draw window.

Prototype
```
void HRWinInvertLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> x1`	x coordinate of line start point.
`-> y1`	y coordinate of line start point.
`-> x2`	x coordinate of line end point.
`-> y2`	y coordinate of line end point.

Result Returns nothing

HRWinInvertPixel

Purpose Inverts a pixel in the draw window.

Prototype
```
void HRWinInvertPixel ( UInt16 refNum, Coord x, Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> x`	Pointer to the x coordinate of a pixel.
`-> y`	Pointer to the y coordinate of a pixel.

Result Returns nothing

HRWinInvertRectangle

Purpose Invert a rectangle in the draw window.

Prototype
```
void HRWinInvertRectangle ( UInt16 refNum, RectangleType
*rP, UInt16 cornerDiam )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> rP`	pointer to the rectangle to invert.
`-> cornerDiam`	Radius of rounded corners. Specify zero for square corners.

Result Returns nothing

HRWinInvertRectangleFrame

Purpose Inverts a rectangular frame in the draw window.

Prototype
```
void HRWinInvertRectangleFrame ( UInt16 refNum, FrameType
frame, RectangleType *rP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> frame`	Type of frame to draw.
`-> rP`	Pointer to rectangle to frame.

Result Returns nothing

HRWinPaintBitmap

Purpose Draw a bitmap in the current draw window at the specified coordinates with the current draw mode.

Prototype
```
void HRWinPaintBitmap ( UInt16 refNum, BitmapType *bitmapP,
Coord x, Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> bitmapP`	Pointer to a bitmap.
`-> x`	The x coordinate of the upper-left corner.
`-> y`	The y coordinate of the upper-left corner.

Result Returns nothing

HRWinPaintChar

Purpose Draw a character in the draw windonw using the current drawing state.

Prototype
```
void HRWinPaintChar ( UInt16 refNum, WChar theChar, Coord x,
Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> theChar`	Pointer to a character to draw.
`-> x`	x coordinate of the location where the character is to be drawn (left bound).
`-> y`	Y coordinate of the location where the character is to be drawn (top bound).

Result Returns nothing

HRWinPaintChars

Purpose Draw the specified characters in the draw windonw with current draw state.

Prototype
```
void HRWinPaintChars ( UInt16 refNum, const Chara chars,
Int16 len, Coord x, Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> chars`	Pointer to the characters to draw.
`-> len`	Length in bytes of the characters to draw.
`-> x`	X coordinate of the first character to draw (left bound).
`-> y`	Y coordinate of the first character to draw (top bound).

Result Returns nothing

Comments

HRWinPaintLine

Purpose Draw a line in the draw window using the current drawing state.

Prototype
```
void HRWinPaintLine ( UInt16 refNum, Coord x1, Coord y1,
Coord x2, Coord y2 )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> x1`	X coordinate of line beginning point.
`-> y1`	Y coordinate of line beginning point.

	`-> x2`	X coordinate of line endpoint.
	`-> y2`	Y coordinate of line endpoint.

Result Returns nothing

HRWinPaintLines

Purpose Draw several lines in the draw window using the current drawing state.

Prototype
```
void HRWinPaintLines ( UInt16 refNum, UInt16 numLines,
WinLineType lines[] )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> numLines`	Number of lines to paint.
`-> lines`	Array of lines.

Result Returns nothing

HRWinPaintPixel

Purpose Render a pixel in the draw window with current drawing state.

Prototype
```
void HRWinPaintPixel ( UInt16 refNum, Coord x, Coord y )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> x`	Pointer to the x coordinate of a pixel.
`-> y`	Pointer to the y coordinate of a pixel.

Result Returns nothing

HRWinPaintPixels

Purpose Render several pixels in the draw window with current drawing state.

Prototype
```
void HRWinPaintPixels ( UInt16 refNum, UInt16 numPoints,
PointType pts[] )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> numPoints`	Number of pixels to paint.
`-> pts`	Array of pixels.

Result Returns nothing

HRWinPaintRectangle

Purpose Draw a rectangle in the draw window with current drawing state.

Prototype
```
void HRWinPaintRectangle ( UInt16 refNum, RectangleType *rP,
UInt16 cornerDiam )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> rP`	Pointer to rectangle to draw.
`-> cornerDiam`	Radius of rounded corners. Specify zero for square corners.

Result Returns nothing

HRWinPaintRectangleFrame

Purpose Draw a rectangular frame in the draw window with the current drawing state.

Prototype
```
void HRWinPaintRectangleFrame ( UInt16 refNum, FrameType
frame, RectangleType *rP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> frame`	Type of frame to draw.
`-> rP`	Pointer to rectangle to frame.

Result Returns nothing

HRWinRestoreBits

Purpose copy the contents of the specified window to the draw window and delete the passed window.

Prototype
```
void HRWinRestoreBits ( UInt16 refNum, WinHandle winHandle,
Coord destX, Coord destY )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> winHandle`	Handle of window to copy and delete.
`-> destX`	X coordinate in the draw window to copy to.
`-> destY`	Y coordinate in the draw window to copy to.

Result Returns nothing

HRWinSaveBits

Purpose Creates an off-screen window and copy the specified region from the draw window to the off-screen window.

Prototype
```
WinHandle HRWinSaveBits ( UInt16 refNum, RectangleType
*sourceP, UInt16 *error )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> sourceP`	Pointer to the bounds of the region to save, relative to the display.
`<- error`	Pointer to any error encountered by this function.

Result Returns the handle of the Window containing the saved image, or zero if an error occurred.

HRWinScreenMode

Purpose Sets or retunes display parameters, including display width and height, bit depth and color support.

Prototype
```
Err HRWinScreenMode ( UInt16 refNum, WinScreenModeOperation
operation, UInt32 *widthP, UInt32 *heightP, UInt32 *depthP,
Boolean *enableColorP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> operation`	The work this function is to perform, as specified by one of the following:

`winScreenModeGet`
Returns the current settings for the display.

`winScreenModeGetDefaults`
Returns the default settings for the display.

`winScreenModeGetSupportedDepths`
Returns the supported screen depth stored in depthP. See WinScreenMode of SDK for more information.

`winScreenModeGetSupportsColor`
Returns true as the value of the enableColorP, when color mode can be enabled.

`winScreenModeSet`
Change display settings to the values specified by the other arguments.

`winScreenModeSetToDefaults`
Change display settings to default values.

`<-> widthP`	Pointer to New/old screen width.
`<-> heightP`	Pointer to New/old screen height.
`<-> depthP`	Pointer to New/old /available screen depth.
`<-> enableColorP`	Pointer to Pass true to enable color drawing mode.

Result If no error, returns values as specified by the argument. Various invalid arguments may cause this function to return a sysErrParamErr result code. A failed allocation can cause this function to return a memErrNot EnoughSpace error.

Comments Return parameter (`width`, `height`) on each drawing mode

`operation`	`winScreenModeGet` `winScreenModeGetDefaults`

Table 7-9 operation : `winScreenModeGet`

	WinScreenMode	HRWinScreenMode
Compatibility mode	width: 160 height: 160	width: 160 height: 160
High-resolution mode	width: 160 height: 160	width: 320 height: 320

Table 7-10 operation : `winScreenModeGetDefaults`

	WinScreenMode	HRWinScreenMode
Compatibility mode	width: 160 height: 160	width: 160 height: 160
High-resolution mode	width: 160 height: 160	width: 160 height: 160

Operations associated with switching of drawing mode

`operation`	`winScreenModeSet` `winScreenModeSetToDefaults`

Table 7-11 operation : `winScreenModeSet`

	WinScreenMode		HRWinScreenMode	
	width: 160 height: 160	width: 320 height: 320	width: 160 height: 160	width: 320 height: 320
Compatibility mode	Compatibility mode	Invalid	Compatibility mode	Compatibility mode `->` High-resolution mode
High-resolution mode	High-resolution mode	Invalid	High-resolution mode `->` Compatibility mode	High-resolution mode

Table 7-12 operation : `winScreenModeSetToDefaults`

	WinScreenMode	HRWinScreenMode
Compatibility mode	Compatibility mode	Compatibility mode
High-resolution mode	High-resolution mode `->` Compatibility mode	High-resolution mode `->` Compatibility mode

HRWinScrollRectangle

Purpose Scroll a rectangle in the draw window.

Prototype

```
Err HRWinScrollRectangle ( UInt16 refNum, RectangleType *rP,
WinDirectionType direction, Coord distance, RectangleType
*vacatedP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> rP`	Pointer to rectangle to scroll.
`-> direction`	Direction to scroll(winUp, winDown, winLeft, winRight).
`-> distance`	Distance to scroll in pixels.
`<- vacatedP`	Pointer to the rectangle that needs to be redrawn because it has been vacated as a result of the scroll.

Result Return nothing

HRWinSetClip

Purpose Set the clipping rectangle of the draw window.

Prototype
```
void HRWinSetClip ( UInt16 refNum, RectangleType *rP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> rP`	Pointer to a structure holding the clipping bounds. Each parameter of `rP(topleft.x,topleft.y,extent.x,extent.y)` should be a multiple of two.

Result Return nothing

HRWinSetWindowBounds

Purpose Set the bounds of the window to display relative coordinates.

Prototype
```
void HRWinSetWindowBounds ( UInt16 refNum, WinHandle
winHandle, RectangleType *rP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> winHandle`	Handle for the window for which to set the bounds.
`-> rP`	Pointer to rectangle to use for bounds. Each parameter of `rP(topleft.x,topleft.y,extent.x,extent.y)` should be a multiple of two.

Result Return nothing

HRWinWindowToDisplayPt

Purpose Convert a window-relative coordinate to a display- relative coordinate.

Prototype
```
void HRWinWindowToDisplayPt ( UInt16 refNum, Coord *extentX,
Coord *extentY )
```

Parameters

`-> refNum`	High-resolution library reference number.
`<-> extentX`	Pointer to x coordinate to convert.
`<-> extentY`	Pointer to y coordinate to convert.

Result Return nothing

Bitmap API

HRBmpBitsSize

Purpose Return the size of the bit map's data.

Prototype
```
UInt32 HRBmpBitsSize ( UInt16 refNum, BitmapType *bitmapP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> bitmapP`	Pointer to bitmap.

Result Returns the size in bytes of the bitmap's data, excluding the header and the color table

HRBmpSize

Purpose Return the size of the bit map's data.

Prototype
```
UInt32 HRBmpSize ( UInt16 refNum, BitmapType *bitmapP )
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> bitmapP`	Pointer to bitmap.

Result Returns the size in bites of the bitmap's data, including the header and the color table.

HRBmpCreate

Purpose Create bitmap.

Prototype
```
BitmapType *HRBmpCreate ( UInt16 refNum, Coord width, Coord
height, UInt8 depth, ColorTableType *colortableP, Uint16
*error)
```

Parameters

`-> refNum`	High-resolution library reference number.
`-> width`	The width of the bitmap in pixels. Must not be 0.
`-> height`	The height of the bitmap in pixels. Must not be 0.
`-> depth`	The pixel depth of the bitmap. Must be 1,2,4 or 8. This value is used as the pixelSize field of BitmapType.
`-> colortableP`	A pointer to the color table associated with the bitmap, or NULL if the bitmap should not include a color table. If specified, The number of colors in the color table must match the depth parameter (2 for 1-bit, 4 for 2-bit, 16 for 4-bit, and 256 for 8-bit).

	`<- error`	Contains the error code if an error occurs.

Result Return a pointer to the new bitmap structure or NULL if an error occurs. The parameter Error contains one of the following:

`errNone`	Success
`sysErrParamErr`	The `width`, `height`, `depth` or `colorTableP` is invalid.
`memErrNotEnoughSpace`	There is not enough memory available to allocate the structure.

Fonts API

HRFntGetFontSize

Purpose Return the font ID of current font.

Prototype
```
HRFontID HRFntGetFont ( UInt16 refNum)
```

Parameters	`-> refNum`	High-resolution library reference number.

Result Return the font ID of current font

HRFntSetFont

Purpose Set the current font.

Prototype
```
HRFontID HRFntSetFont ( UInt16 refNum, HRFontID font )
```

Parameters]	`-> refNum`	High-resolution library reference number.
	`-> font`	ID of the font to make the active font.

Result Return the ID of the current font before the change.

HRFontSelect

Purpose Display a dialog box in which the user can choose and return a FontID value representing the user's choice.

Prototype
```
HRFontID HRFontSelect ( UInt16 refNum, HRFontID font )
```

Parameters	`-> refNum`	High-resolution library reference number.

`-> font` A font ID value specifying the font to be highlighted as the default choice in the dialog box this function displays. The value must be one of the following.

US:
```
hrStdFont
hrBoldFont
hrLargeBoldFont
```

J:
```
hrStdFont
hrBoldFont
hrLargeFont
hrLargeBoldFont
```

Result Return selected font ID

Notes

Determining If High Resolution Library Is Available

As shown in "Availability of library" to determine whether a device provides High-resolution library, use `sonySysFtrSysInfoLibrHR` bit in `libr` field of `SonySysFtrSysInfoType` which is obtained by `sonySysftrNumSysInfoP` as a feature number.[1]

Sub-Launch

Be careful of the screen mode when Sub-Launching other applications from the application program or being Sub-Launched by another application program.
In switching the screen mode, make sure to close the menu, command bar, or pop up window; otherwise, an error might be occurred.

Sub-Launching

In Sub-Launching the other application from a high-resolution mode application, switch the mode to normal before Sub-Launching, if the application is not corresponding to high-resolution mode.

Being sub-Launched

A sub-Launching application must be saved first with `WinSaveBits`, when sub-launching from an application activated with compatible mode to the one corresponding to high-resolution mode. Then switch the mode to high-resolution. When the application ends, change to the compatible mode to redraw the saved screen with (`WinRestoreBits`).

1. Other distinction methods may be offered in the future.

Switching a screen mode

It takes time to switch the screen mode. Try programming to reduce the number of switching as possible as you can.

BmpCompress

`BmpCompress` doesn't correspond to the bitmap that exceeds 160 x 160 x 8 bit, and is not supported.

About High Resolution Assist

Despite the use of High-Resolution API, this function enables activation of the existing applications in High-Resolution mode.
By using this function, clear high resolution display (such as characters) will be available in the application that run on Palm OS provided models.

However, some applications activate in one of the following ways if High Resolution Assist function is used.

- Performances are largely deteriorated (ex. game).
- Operational irregularities occur.
  such as Display divided in half or characters are distorted.

As for slow performance particularly , it's hard to distinguish for users whether the performance is right or not because it looks normal.
To avoid performance deteriorations in advance, use the codes below in your reference to run applications in compatible mode regardless the High-Resolution Assist settings.
For some software which enable the same functions of this, without using High-Resolution Assist function the compatible mode may not work.

CASE 1: The Screen Mode is fixed in the application

```
static Err AppStart(void)
{

  ...

  /* High Resolution Mode Set */
  error = SysLibFind( sonySysLibNameHR, &hrRefNum);
  if (error) {
    error= SysLibLoad( 'libr', sonySysFileCHRLib,
     &hrRefNum);
  }

  if (!error) {
    UInt32 width, height;

    width= height= 160;
```

```
    HROpen( hrRefNum);
    HRWinScreenMode( hrRefNum, winScreenModeSet, &width,
     &height, NULL, NULL);
    HRClose(hrRefNum);
  }

  ...

  return errNone;
}
```

CASE 2: The Screen Mode is switched frequently in the application

```
#include <SonyHRLib.h>

UInt16 hrRefNum = sysInvalidRefNum;
Booleanhrlib= false;

...

function FUNCTION(....)
{
  WinScreenMode( winScreenModeSetToDefaults, NULL, NULL,
   NULL, NULL);
  /* If you use above API-call, you must set to below again
*/
  if (hrlib) {
    UInt32 width, height;

    width= height= 160;
    HRWinScreenMode( hrRefNum, winScreenModeSet, &width,
     &height, NULL, NULL);
  }
}

...

static Err AppStart(void)
{

  ...

  /* High Resolution Mode Set */
  error = SysLibFind( sonySysLibNameHR, &hrRefNum);
  if (error) {
```

```
      error= SysLibLoad( 'libr', sonySysFileCHRLib,
       &hrRefNum);
   }

   if (!error) hrlib= true;

   if (hrlib) {
      UInt32 width, height;

      width= height= 160;
      HROpen( hrRefNum);
      HRWinScreenMode( hrRefNum, winScreenModeSet, &width,
       &height, NULL, NULL);
   }

   ...

   return errNone;
}

static void AppStop(void)
{

   ...

   if (hrlib) {
      HRWinScreenMode(hrRefNum, winScreenModeSetToDefaults,
       NULL, NULL, NULL, NULL);
      HRClose(hrRefNum);
   }

   ...

}
```

8

Memory Stick® Audio : Sony Msa Library

Some devices in the CLIÉ™ make it possible to replay ATRAC3 and MP3[1] form of music data and obtain music information. These functions are given by the Memory Stick audio library. By using it, application enables users to provide not only plain music player function but interface with music expression as an entertainment.

Configuration and Function

Configuration

The Memory Stick audio libray consists of two modules listed below.

- Audio interface (MSA I/F)

  It manages interface with application and provides API which can operate audio that is independent of codec and physical media and hides MsaOut.

- Audio out put control (MSAOut)

  It manages audio output control including volume and balance adjustment and provides API which can control sound output that is independent of music data and replay condition. API is hidden by MSA I/F so that the application does not recognize MsaOut.

MSA I/F funcitonal

Obtaining audio information

Msa I/F library provides users audio player replay information and the functions to obtain album and track information.
The replay information contains a replay list for play, replay status, replay mode, replay

[1] This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

speed, replay position, audio player replay information, and the list of replayed tracks when in shuffling mode.
The track information includes track names, artist names, and information for the limited replay mode.

Specifying audio information

Msa I/F library provides users Audio player replay information, the functions to specify album and track information and the function to edit Memory Stick audio.
The replay information contains a replay list for play, replay status, replay mode, replay speed, and replay position.
The edit function includes the replay order change and deletion of tracks.

Audio replay control

Msa I/F library provides the basics such as replaying and suspending the audio player.

The Utility for data structure

Msa I/F library provides the functions to convert the sound unit into time, the time into sound unit and the PBListIndex into Track No.

MsaOut functional

Audio output mode setting

The function that sets audio output mode to the one specified (It will be any of these: stereo, monaural, main sound, sub sound, and dual sounds). Each mode is represented by a specific numeric value. You specify a corresponding value to set to a particular mode.
The setting can be changed anytime; the change will be immediately reflected.
Your application should first get audio output control capability information (i.e. monaural setting, main-sub sounds switching) of a device to control them.

Audio volume control

The function that sets audio volume to a specified level.
Volume of L(left/main) and R(right) channels are set separately.
To enable AVLS function and such, the maximum volume can be also set (for L and R channels, respectively).
The settings are made by specifing a particular level: 0 represents no sound and resolution-1 represents the maximum. The volume is controled so that it will not exceed its maximum.
The setting can be changed anytime; the change will be immediately reflected.
Hardware with sufficient resolution, will convert the volume change to dB linear.
Your application should first get audio output control capability information (i.e. volume control, separate control of L/R channels, volume level resolution) of a device to control them.

Audio mute control

The function that sets audio mute ON/OFF.
The setting can be changed anytime; the change will be immediately reflected.
Depending on capability of a device, the change will be made gradually to prevent emitting any noise.
Your application should first get audio output capablity information (i.e. audio mute control) of a device to control it.

Audio output information retrieval

The function that gets audio output information as audio output peak level and spectrum data.
Audio output level of L (left/main) and R (right) channels are obtained separately.
Spectrum data is also obtained separately for each band.
The output and spectrum data can be obtained by specifing the specific level value: 0 represents no sound, and resolution-1 represents the maximum.
Hardware with sufficient resolution or equivalent function, will convert the value to dB linear.
Your application should first get audio output capability information (i.e. audio output peak level, separate control of L/R channels, resolution of output peak level, spectrum data retrieval, number of bands, resolution of spectrum data retrieval) of a device and interpret them.

Glossary

Album Several songs on the Memory Stick media or Database[1] and are the same as on CD and MD.
This information is saved on the Database.
The application can specify only one album to replay. This is called current album.

Track Audio track and normally a unit of one track. On the Memory Stick media, it corresponds to one audio file.
One album is composed of several tracks.

Track No A number for all the tracks in the Album in order of replay. Starts from 1 up to 400- in the greatest. Never use the same number twice. Excludes zero.
Usually, it replays in order of the track number, unless the list is re-specified.

PBList (PlayBack List, PBList) A series of tracks in order of replay position. (List of TrackNo) 400 is the greatest.
There are two kinds: One is made by default of an Album. The other is set by user (An application).
By editing the list, several "replay units" can be created from the same album.

1. This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

PB list is changeable. However, the album remains the same even though the list has changed.

PB List index A series of numbers in the PB List in order of replay.
Always starts from 1 to 400 in the greatest.
PB list index is not directly related to the track number.

Background Playback[1] Playing audio while another application is active.

SU/ Sound unit A unit of audio data held together in some standard. On the ATRAC3, regardless the bit rate, 23.2msec (44.1KHz 1024 sample) data is contained. On th MP3, when the bit rate is 128Kbps, 26.1msec (fixed sampling freqency: 44.1KHz) data is contained.

PB Mode PB Mode information is as below. All will be clear when the MSA Library closes.

Repeat replay	Repeat/Non-repeat
Replay extent	All tracks/1Track/of your choice
Kinds of PBList	Album (default)/Program (user definition)
The order of replay	Ascending/descending/Shuffle
Confirmation for the limited use of contents	Replay/skip it then go next/Stop

PB Status PB Status information is as below.

Status	Stopping/ replaying
PBrate	Replay direction(BWD/FWD). Consisting of the decoding SU number in 1 Block and decoding distance.
Position	Track number, the beginning position of the track (sound unit)

Audio Interface (MSA I/F) reference

Data Structures

MsaErr

On the Msa Function, if an error occurs, the error parameter contains one of the following.

`msaErrParam`	The parameter is invalid.
`msaErrNotOpen`	The library isn't open.

1. This function is not supported by Audio Adapter.

`msaErrStillOpen` The library is still open.

`msaErrMemory` The memory error occurs.

`msaErrNoVFSMgr` The file system error occurs.

`msaErrAlreadyOpen` The library has been open already.

`msaErrNotImplemented`
Not being implemented.

`msaErrSecurity` Security error occurs.

`msaErrPBListSet` The error occurs in setting the PB list.

`msaErrNotShuffleMode`
Not a shuffle mode.

`msaErrNoAlbum` No album is inside.

`msaErrNoMedia` No Memory Stick media is inserted.

`msaErrInvalidMedia`
No Memory Stick media resoponding to OpenMG™ jukebox is inserted.

`msaErrDifferentMode`
The operation is made in a different mode.

`msaErrEnumerationEmpty`
No Album information is in the Memory Stick.

`msaErrEnumerationdetail`
The error occurs in aquiring Album information.

`msaErrNotConnected`
Audio device is not connected.

`msaErrReadFail` MP3 file reading error occurs.

`msaErrNotEnoughSpace`
Disable to allocate memory for MP3 file.

`msaErrInvalidFormat`
Invalid file format of MP3.

`msaErrNotMP3File`
Not MP3 file.

AlbumInfoType[1]

Defines the form of album info that is obtainable by `MsaAlbumEnumerate()`. Refer to SonyMsaLib.h

1. This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

```
typedef struct{
  UInt16    albumtype;
  UInt16    albumRefNum;
  UInt16    volRef;
  Char      *nameP;
  UInt16    fileNameLength;
  UInt8     maskflag;
  UInt8     reserve1
  UInt16    code;
  MemHandle infoH;
  UInt32    reserve2;
}AlbumInfoType
```

Field Descriptions

<-	albumtype	Audio format of Album
<-	albumRefNum	Reference number of Album
<-	volRef	Volume reference number of Album
<-	nameP	File path of Album
->	fileNameLength	Buffer size of nameP
<->	maskflag	Bit field of acquiring information
->	code	Specifies the character code of acquiring information
<->	infoH	Handle with obtained information

MsaPBList

Structure used when obtainning `PBList` that is specified by `MsaGetPBList()`or specifying `PBList` by `MsaSetPBList()`.

```
typedef struct{
  UInt16 format;
  UInt16 reserve1;
  UInt32 creatorID;
  UInt32 appinfo;
  UInt32 reserve2;
  UInt16 pblistindex[1];
} MsaPBList,*MsaPBListPtr;
```

Field Descriptions

format	Indicates PBList format version. It's 0x0001 this time.
reserved1	Reservation. Not in use.
creatorID	Indicates CreatorID of the application where PBList is specified. Default is `msaLibCreatorID`.
appinfo	The value that the applicaion uses likewise distinguishing applications.

`reserved2`	Reserved. Not in use.
`pblistindex[1]`	PBList Index of the first track.

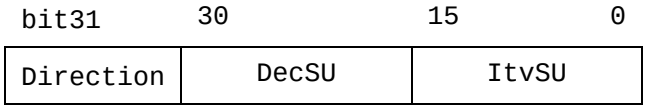
MsaPBStatus

Structure used when obtainning PBStatus that is specified by `MsaGetPBStaus()`or specifying PBStatus by `MsaSetPBStatus()`.

```
typedef struct{
  MsaPlayStatus  status;
  UInt32 pbRate;
  UInt16 currentTrackNo;
  UInt32 currentSU;
}MsaPBStatus, *MsaPBStatusPtr;
```

Field Descriptions

`status`	Status of the player During the stop or replay
`pbRate`	The speed of replay. See information below and glossary.



`currentTrackNo`	PBList (PB List Index)
`currentSU`	Off set from the top Sound Unit

MsaPlayStatusEnum

Defines status of the player that is obtainable by `MsaGetPBStatus()`

```
typedef enum{
  msa_PLAYSTATUS,
  msa_STOPSTATUS,
  msa_OTHERSTATUS
}MsaPlayStatus;
```

Field Descriptions

`msa_PLAYSTATUS`	Player is replaying.
`msa_STOPSTATUS`	Player is stopping.
`msa_OTHERSTATUS`	Other than these above.

MsaPBMode

Structure used when obtainning PB Mode that is specified by `MsaGetPBMode()`or specifying PBMode by `MsaSetPBMode()`.

```
typedef struct{
  MsaPlayloop loop;
  MsaScope  scope;
  MsaPbListType  pblisttype;
```

```
    MsaSequence seq;
    MsaConfirm  confirm;
    UInt8  reserve;
    UInt16 pblistindex1;
    UInt32 startTime;
    UInt16 pblistindex2;
    UInt32 endTime;
  } MsaPBMode, *MsaPBModePtr;
```

Field Descriptions

`loop`	Indicates whether it repeats after the replay of PBList.
`scope`	The scope of replay.
`pbListType`	Type of PBList.
`seq`	The form of order for replaying.
`confirm`	Confirmation form to replay tracks, including the one with limit numbers to replay.
`reserve`	Reservation. Not in use.
`pblistindex1`	The beginning PBListIndex during the AB repeat.
`startTime`	The Starting time of the AB repeat (sound unit).
`pblistindex2`	The ending PBListIndex during the AB repeat.
`endTime`	The end time of the AB repeat (sound unit).

MsaPlayloop Enum

Defines continuous replay after finishing the PBList that is obtainable by `MsaGetPBMode()`.

```
typedef enum{
  msa_PLAY_NOLOOP,
  msa_PLAY_LOOP,
  msa_PLAY_NOLIMIT = 0xffff
}MsaPlayloop;
```

Field Descriptions

`msa_PLAY_NOLOOP`	After finishing the PBList, it stops.
`msa_PLAY_LOOP`	After finishing the PBList, it replays from the top.
`msa_PLAY_NOLIMIT`	Haven’t yet settled. It replays unlimitedly.

MsaScopeEnum

Defines a scope to replay that is obtainable by `MsaGetPBMode()`.

```
typedef enum{
  msa_SCOPE_ALL,
  msa_SCOPE_ONETRACK,
  msa_SCOPE_ARB
```

```
}MsaScope;
```

Field Descriptions

`msa_SCOPE_ALL` Indicates all tracks in the PBList.

`msa_SCOPE_ONETRACK`
Indicates a track in the PB List.

`msa_SCOPE_ARB` Indicates the definable (defined) scope.

MsaPbListType Enum

Defines the form of PBList that is obtainable by `MsaGetPBMode()`.

```
typedef enum{
  msa_PBLIST_ALBUM,
  msa_PBLIST_PROGRAM
} MsaPbListType;
```

Field Descriptions

`msa_PBLIST_ALBUM` Indicates that it's made by Album default.

`msa_PBLIST_PROGRAM`
Indicates that it's defined by user.

MsaSequence Enum

Defines the form of replaying order that is obtainable by `MsaGetPBMode()`.

```
typedef enum{
  msa_SEQUENCE_CONTINUE,
  msa_SEQUENCE_REVERSE,
  msa_SEQUENCE_SHUFFLE
} MsaSequence;
```

Field Descriptions

`msa_SEQUENCE_CONTINUE`
Replay from the top of the PBList.

`msa_SEQUENCE_REVERSE`
Replay from the end of the PBList.

`msa_SEQUENCE_SHUFFLE`
Replay in shuffle.

MsaConfirm Enum

Defines the form of replay confirmation to the limited track to replay that is obtainable by `MsaGetPBMode()`.

```
typedef enum{
  Msa_CONFIRM_AUTO,
  Msa_CONFIRM_PASS,
  Msa_CONFIRM_STOP
} MsaConfirm;
```

Field Descriptions

`msa_CONFIRM_AUTO`	All of the confirmation related to the copyright turns automatically OK.
`msa_CONFIRM_PASS`	All of the confirmation related to the copyright turns automatically cancelled. (Haven't yet settled.)
`msa_CONFIRM_STOP`	Stops at the time of the confirmation related to the copyright.

MsaTrackInfo

Structure used when getting the track information by `MsaGetTrackInfo()`

```
typedef struct{
  UInt32 titleoffset;
  UInt32 artistoffset;
  UInt32 genreoffset;
  UInt32 commentoffset;
  UInt32 albumoffset;
  Char   *albumP;
  UInt32 totalsu;
  UInt16 tracknum;
  UInt16 limitinfo;
  UInt16 codecmode;
  MsaCodecType  codectype;
  UInt16 frequencey;
  Char   trackinfo[1];
} MsaTrackInfo, *MsaTrackInfoPtr;
```

Field Descriptions

`titleoffset`	Off set value from `trackinfo[0]` to `title` data
`artistoffset`	Off set value from `trackinfo[0]` to `artist` data.
`genreoffset`	Off set value from `trackinfo[0]` to `genre` data.
`commentoffset`	Off set value from `trackinfo[0]` to `comment` data.
`albumoffset`	Off set value from `trackinfo[0]` to `album` data.
`totalsu`	Album data: Total replay time(sound unit) Track data: Replay time.(sound unit)
`tracknum`	Album data: The track numbers in the Album. Track data: `TrackNO`
`limitinfo`	A flag to controll the repaly*
	`bit15` Indicates if the time is limited. If it is, 1 is set.
	`bit7` Indicates if number of times is limited. If it is, 1 is set.
	`bit6` Indicates if the content is outdated. If it is, 1 is set.
`codecmode`	Compression mode*
`frequency`	Sampling frequency.

`trackinfo`	The top data of string information (`Title/artist/genre/comment` data)

(`*` means that those are existing only on the Track data)

MsaCodecType Enum

Defines the form of compress mode that is obtainable by `MsaGetTrackInfo()`.

```
typedef enum{
  msa_CODEC_ATRAC,
  msa_CODEC_MP3
}MsaCodecType
```

Field Descriptions

`msa_CODEC_ATRAC`	ATRAC
`msa_CODEC_MP3`	MP3

MsaTrackRestrictionInfo

Structure used when obtainning restricted replay information of the track by `MsaGetTrackRestrictionInfo()`

```
typedef struct{
  DateTimeType  pbstartdatetime;
  DateTimeType  pbfinishdatetime;
  UInt8  maxplaytime;
  UInt8  curplaytime;
  UInt16 reserved;
}MsaTrackRestrictionInfo,*MsaTrackRestrictionInfoPtr;
```

Field Descriptions

`Pbstartdatetime`	The starting date and time of the replay.
`Pbfinishdatetime`	The ending date and time of the replay.
`Maxplaytime`	The maxumum number of the replay permission.
`Curplaytime`	The number of the `Replay`
`reserve`	Reservation. Not in use.

MsaControlKey Enum

Defines the control forms that can be specified by `MsaSetControlKey()`.

```
typedef enum{
  msaControlkeyNoKey,
  msaControlkeyPlayPause,
  msaControlkeyFRPlay,
  msaControlkeyFFPlay,
  msaControlkeyPause,
  msaControlkeyStop,
  msaControlkeyVolm,
```

```
    msaControlkeyVolp,
    msaControlkeyPlay,
    msaControlkeyCue,
    msaControlkeyRev,
    msaControlkeyAMSp,
    msaControlkeyAMSm,
    msaControlkeyFF,
    msaControlkeyFR,
    msaControlkeyRepeat,
    msaControlkeyPlay1Track,
    msaControlkeyPlayAllTrack,
    msaControlkeyPlaySection,
    msaControlkeySetSection,
    msaControlkeyOrderNormal,
    msaControlkeyOrderReverse,
    msaControlkeyOrderShuffle,
    msaControlkeyHold,
    msaControlkey_NUMCODE
}MsaControlKey;
```

MsaControlKeyState Enum

Defines the key status that can be specified by `MsaSetControlKey()`.

```
typedef enum{
    msaControlKeySet,
    msaControlKeyRelease,
    msaControlKeyLong
} MsaControlKeyState;
```

Field Descriptions

`msaControlKeySet` Key is pressed.

`msaControlKeyRelease` Key is released.

`msaControlKeyLong` Key is long pressed.

MsaTime

Structure used by `MsaSuToTime()` and `MsaTimeToSu()`

```
typedef struct{
    UInt16 minute;
    UInt16 second;
    UInt16 frame;// milli-second
}MsaTime,*MsaTimePtr;
```

System I/F

MsaLibOpen

Purpose Opens Memory Stick Audio library to initialize.

Prototype
```
Err MsaLibOpen(UInt16 msaLibRefNum, UInt16 mode)
```

Parameters

`-> msaLibRefNum`	Reference number of library.
`-> mode`	A mode to open library At present, only `msaLibOpenModeAlbum` is available.

Result

`errNone`	No error.
`msaErrAlreadyOpen`	
`msaErrMemory`	
`msaErrDifferentMode`	
`expErrCardNotPresent`	

Comments An application needs to call this function before using the Memory Stick audio library. If the Memory Stick audio library has already been opened, `MsaLibOpen` increases the open accounts.
Memory Stick audio replay continues to control other applications even though an applicaion is finished. So the MSA is accessible by multiple libraries or applications. (The control isn't available exlusively.)

MsaLibClose

Purpose Closes MSA library.

Prototype
```
Err MsaLibClose(UInt16 msaLibRefNum, UInt16 mode)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> mode`	A mode specified when opening library At present, only `msaLibOpenModeAlbum` is available.

Result

`errNone`	No error.
`msaErrStillOpen`	Library has been used by other modules. (no error)
`msaErrNotOpen`	No library has been opened.
`msaErrMemory`	
`msaErrDifferentMode`	
etc.	

Comments All information is clear when closed.

MsaLibGetCapability

Purpose It obtains the capability to replay.

Prototype
```
Boolean MsaGetCapability(UInt16 msaLibRefNum,
 MsaCodecType codectype, UInt32 pbrate)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib
`-> codectype`	Codec type
`-> pbrate`	pbrate (direction,decode su,interval su)

Result

True	Replay available
False	Replay unavailable

Comments

MsaGetAPIVersion

Purpose Obtains API version

Prototype
```
UInt32 MsaGetAPIVersion(UInt16 msaLibRefNum)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib

Result Version number returns.

1	Only ATRAC3 is available
2	ATRAC3 and MP3 are available

MsaLibEnforceOpen

Purpose Closes the current Msa libray that has been opened then opens it again.

Prototype
```
Err MsaLibEnforceOpen(UInt16 msaLibRefNum,
UInt16 mode, UInt32 creator)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib
`-> mode`	The mode to open
`-> creator`	CreatorID

Result

`errNone`	No error
`msaErrStillOpen`	

Comments `MsaLibEnforceOpen` broadcasts `EnforceOpen` event with Notification.
Follow the instructions below.
Register `EnforceOpen` event Notification. Then activate AppA which is an application that `MsaLibClose()` is put in this Notification handler, and keep it active on back ground. If `MsaLibEnforceOpen` is called on AppB, AppA enables to close Msa Library and AppB enables to open it through Notification.

Obtaining information I/F

MsaAlbumEnumerate[1]

Purpose Get Album list in a Memory Stick.

Prototype
```
Err MsaAlbumEnumerate(UInt16 msaLibrefNum,
 UInt32 *albumIteratorP,AlbumInfoType *infoP)
```

Parameters

`-> msaLibrefNum`	Reference number of MsaLib.
`<-> albumIteratorP`	Pointer to the last album. Returns a pointer to the next album.
`<-> infoP`	Pointer to album information specified by `albumIteratorP`.

Result

`errNone`	No error.
`msaErrNotOpen`	
`msaErrDifferentMode`	
`msaErrNoMedia`	
`msaErrInvalidMedia`	
`msaErrNoAlbum`	
`msaErrEnumerationEmpty`	
`msaErrEnumerationdetail`	
`msaErrParam`	

Comments Searches in /HIFI/PBLIST.MSF and the album file specified by the system.
To get such album information as the number of tracks and title: Set a required bit to `maskflag`, and the system returns the information to a handle.
`albumIteratorP` is a variable used to get the next album information. To get the next album information, call API by setting the last album information obtained.

---

1. This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

To get a list of all albums, call API by setting `albumIteratorStart` to `albumIteratorP`; then, call API again by setting a value returned. Repeat this until `albumIteratorStop` is returned to `albumIteratorP`.

The system sends the followings as a result and returned values.

- No album exists:

result	albumIteratorP
msaErrEnumerationEmpty	albumIteratorStop

- One album exists:

result	albumIteratorP
errNone	albumIteratorStop

- More than one album exist:

result	albumIteratorP
errNone	a value to get the next album information

When NULL is set to `infoP->nameP`, only `albumtype`, `albumRefNum` and `volRef` are obtained. Other information such as `infoP->infoH` will not be returned.

Here is a sample code that gets an album list:

```
AlbumInfoType info;
UInt32 albumIterator=albumIteratorStart;

info.maskflag = msa_INF_INFALL;
info.code = msa_LANG_CODE_ASCII;
while(albumIterator!=albumIteratorStop){
  if(MsaAlbumEnumerate(GMsaLibRefNum,&albumIterator,&info){
    /* Get Album Information */
  }else{
    /* Error */
  }
}
```

MsaGetAlbum[1]

Purpose Get current Reference number of a album.

Prototype
```
Err MsaGetAlbum(UInt16 msaLibRefNum, UInt16 *albumRefNum,
UInt32 *dummy)
```

Parameters

->	msaLibRefNum	Reference number of MSA Lib.
<-	albumRefNum	Reference number of a album.
->	dummy	Not used.

Result

errNone	No error.
msaErrNotOpen	
msaErrDifferentMode	
msaErrNoMedia	
msaErrInvalidMedia	
msaErrNoAlbum	
msaErrParam	

MsaGetPBList

Purpose Obtains the current specified PBList.

Prototype
```
Err MsaGetPBList(UInt16 msaLibRefNum,
 MSAPBListPtr pblistP, UInt16 *tracknum)
```

Parameters

->	msaLibRefNum	Reference number of MSA Lib
<->	pblistP	Pointer to the `MSAPBList` structre.
<->	tracknum	Track number in the PBList

Result

errNone	No error
msaErrNotOpen:	
msaErrDifferentMode:	
msaErrNoMedia:	
msaErrInvalidMedia:	
msaErrNoAlbum:	

1. This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

Comments If `PblistP` is NULL, it obtains PBList size. Before obtainning PBList, Users must obtain its size first. If `Tracknum` is 0, it returns the header information of `MsaPBList` structure. (the member, excluding `pblistindex`)

MsaGetPBStatus

Purpose Obtains the current replay status (PB or Stop/PBrate/Position etc).

Prototype
```
Err MsaGetPBStatus(UInt16 msaLibRefNum, MSAPBStatusPtr
pbstatusP)
```

Parameters

->	`msaLibRefNum`	Reference number of MSA Lib.
<-	`pbstatusP`	Pointer to the `MSAPBStatus` structre.

Result

`errNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	

Comments

MsaGetPBMode

Purpose Obtains the current replay status.

Prototype
```
Err MsaGetPBMode(UInt16 msaLibRefNum, MSAPBModePtr pbmodeP)
```

Parameters

->	`msaLibRefNum`	Reference number of MSA Lib.
<-	`pbmodeP`	Pointer to the `MSAPBMode` structre.

Result

`errNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	

MsaGetPBRate

Purpose Obtains the replay speed.

Prototype
```
Err MsaGetPBRate(UInt16 msaLibRefNum, UInt32 * pbrateP)
```

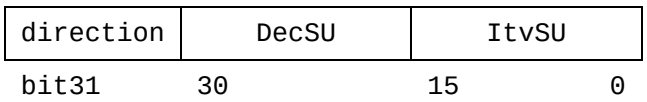
Parameters

->	msaLibRefNum	Reference number of MSA Lib.
<-	pbrateP	The pointer to the memory for storing the replay speed.

Result

errNone	No error
msaErrNotOpen:	
msaErrDifferentMode:	
msaErrNoMedia:	
msaErrInvalidMedia:	
msaErrNoAlbum:	
msaErrParam:	

Comments The replay speed is made of direction and DecSU/ItvSU. See below.



MsaGetPBPosition

Purpose Obtains the replying position.

Prototype
```
Err MsaGetPBPosition(UInt16 msaLibRefNum, UInt16
*currenttrack,
 UInt32*currentposition)
```

Parameters

->	msaLibRefNum	Reference number of MSA Lib.
<-	currenttrack	The pointer to the replaying PB List Index.
<-	currentposition	The pointer to the starting position of the replay.

Result

errNone	No error
msaErrNotOpen:	
msaErrDifferentMode:	
msaErrNoMedia:	

`msaErrInvalidMedia:`
`msaErrNoAlbum:`
`msaErrParam;`

Comments

MsaGetTrackInfo

Purpose Obtains the information of Album and each track.

Prototype
```
MsaGetTrackInfo(UInt16 msaLibRefNum, UInt16 trackNo,
 UInt8 *maskP, UInt16 code, MemHandle *hdlP)
```

Parameters

`-> msaLibRefNum` Reference number of MSA Lib.

`-> trackNo` Track number.

`<-> maskP` Specifies the bit field of obtainning info.

bit7	6	5	4	3		0
title	Artist	Genre	Comment	Album Title	Reserve	NotGetsu

`-> code` Specify the code to obtainning information (1byte or 2byte code).

`-> hdlP` The pointer to the handle with obtained information.

Result

`errNone` No error
`msaErrNotOpen:`
`msaErrDifferentMode:`
`msaErrNoMedia:`
`msaErrInvalidMedia:`
`msaErrNoAlbum:`
`msaErrParam:`
`msaErrMemory:`
`expErrCardNotPresent:`

Comments Obtains `AlbumInfo` if `TrackNo` is 0.
For Track information, the system obtains the memory. After obtainning the information, it releases the memory in the program. The system obtains the specified items by `MaskP` and store it to the specified area (including Null).
Also, it calculates the total playing time of album, if the bit of `NotGetsu`[1] is 0. (If it's 1, the calculations can't be made.)

It sets 1 in the responding bit of `maskP`, if specified data is obtained. If not, sets 0 in the bit. If there is information that is unable to obtain, the writing on the off-set value of `MsaTrackInfo` can't be made. Before using off set value, check the bit of maskP first.

MsaGetShufflePlayedList

Purpose On shuffle mode, it obtains the list of replayed PBListIndex number.

Prototype
```
Err MsaGetShufflePlayedList(UInt16 msaLibRefNum,
UInt32 *shuffleplayedlist)
```

Parameters

`-> msaLibRefNum` Reference number of MSA Lib.

`-> shuffleplayedlist`
The pointer to the list of replayed `PBListIndex` number.

Result

`errNone` No error

`msaErrNotOpen:`

`msaErrDifferentMode:`

`msaErrNoMedia:`

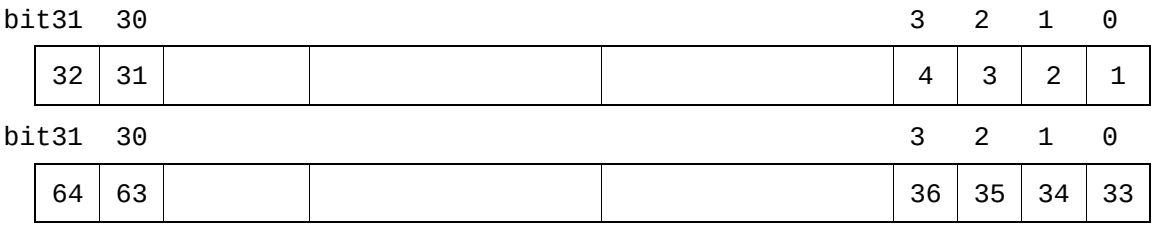
`msaErrInvalidMedia:`

`msaErrNoAlbum:`

`msaErrParam:`

`msaErrNotShuffleMode:`

Comments Available only on shuffle mode. Acquire the area for the size of current PBList (for 32bit) and pass it as an argument. Bit 1 is allocated for the replayed PBListIndex. The list of PBListIndex number is allotted as below.



[1] This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

MsaGetTrackRestrictionInfo

Purpose It obtains the detailed information for the replay restriction.

Prototype
```
Err MsaGetTrackRestrictionInfo(UInt16 msaLibRefNum,
 UInt16 trackNo, MsaTrackRestrictionInfoPtr resrictionP)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> trackNo`	The pointer to the list of `PBListIndex` number that has already replayed.
`-> resrictionP`	The pointer to the detailed information for the replay restriction.

Result

`errNone:`	no error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	
`expErrCardNotPresent:`	

Comments If there is a replay restriction on the `GetTrackInfo`, specify same TrackNO to call this function. Same as (`limittime`).

Specifying information I/F

MsaSetAlbum[1]

Purpose Specify an Album to replay

Prototype
```
Err MsaSetAlbum(UInt16 msaLibrefNum,
 UInt16 albumRefNum, UInt32 *dummy)
```

Parameters

`-> msaLibrefNum`	Reference number of MsaLib.
`-> albumRefNum`	Reference number of Album.

1. This is available only when a version number obtained by using `MsaGetAPIVersion()` is 2.

	-> dummy	Not in use

Result

errNone	No error
msaErrNotOpen	
msaErrDifferentMode	
msaErrNoMedia	
msaErrInvalidMedia	
msaErrNoAlbum	

Comments By setting albumRefNum, obtained by MsaAlbumEnumerate(), it becomes available to replay the Album.

MsaSetPBList

Purpose It specifies the PBList.

Prototype

```
Err MsaSetPBList(UInt16 msaLibRefNum, MSAPBListPtr pblistP,
UInt16 tracknum)
```

Parameters

-> msaLibRefNum	Reference number of MSA Lib.
-> pblistP	The pointer to MSAPBList structre.
-> tracknum	The size of PBList to specify.

Result

errNone:	No error
msaErrNotOpen:	
msaErrDifferentMode:	
msaErrNoMedia:	
msaErrInvalidMedia:	
msaErrNoAlbum:	
msaErrParam:	
msaErrMemory:	
expErrCardNotPresent:	

Comments During the replay, specification can’t be made. Make sure to do it during the stop.

MsaSetPBStatus

Purpose It specifies the replaying status.(PBrate/Position etc).

Prototype
```
Err MSAGetPBStatus(UInt16 msaLibRefNum, MSAPBStatusPtr
*pbstatusP)
```

Parameters

-> msaLibRefNum	Reference number of MSA Lib.
-> pbstatusP	The pointer to MSAPBStatus structure.

Result

errNone	No error

Comments Can't specified status simply.
During the replay, specification can't be made. Make sure to do it when it stops.

MsaSetPBMode

Purpose Specifies the replay status.

Prototype
```
Err MSASetPBMode(UInt16 msaLibRefNum, MSAPBModePtr pbmodeP)
```

Parameters

-> msaLibRefNum	Reference number of MSA Lib.
-> pbmodeP	The pointer to MSAPBMode structure.

Result

errNone	No error
msaErrNotOpen:	
msaErrDifferentMode:	
msaErrNoMedia:	
msaErrInvalidMedia:	
msaErrNoAlbum:	
msaErrParam:	

Comments

MsaSetPBRate

Purpose Set the replaying speed.

Prototype
```
Err MsaSetPBRate(UInt16 msaLibRefNum, UInt32 pbrateP)
```

Parameters

-> msaLibRefNum	Reference number of MSA Lib.

`-> pbrateP` The replay speed.

Result

`errNone` No error

`msaErrNotOpen:`

`msaErrDifferentMode:`

`msaErrNoMedia:`

`msaErrInvalidMedia:`

`msaErrNoAlbum:`

`msaErrParam:`

Comments

The change of "Replay" and "Stop," which is Audio replay system status hasn't occurred. If called during the replay, the specified speed is reflected soon. During the stop, only the speed status has changed. It replays at that speed when in replay status.

The replaying speed is consist of direction and `DecSU/ItvSU`. See below or the reference.

direction	DecSU	ItvSU
bit31	30	15 0

`DecSU=0` can't be specified. The speed to set will be defined later.

MsaSetPBPosition

Purpose

It sets the replaying position.

Prototype

```
Err MSASetPBPosition(UInt16 msaLibRefNum,
 UInt16 currenttrack, UInt32 currentposition)
```

Parameters

`-> msaLibRefNum` Reference number of MSA Lib.

`-> currenttrack` The track number to replay.

`-> currentposition` The position to start to replay.

Result

`errNone` No error

`msaErrNotOpen:`

`msaErrDifferentMode:`

`msaErrNoMedia:`

`msaErrInvalidMedia:`

`msaErrNoAlbum:`

```
msaErrParam;
```

Comments During the replay, the setting isn't available. Be sure to do it when stopped.

MsaEdit

Purpose Edits the audio file on the Memory Stick.

Prototype
```
Err MsaEdit(UInt16 msaLibRefNum, UInt8 command,
UInt16 track1, UInt16 track2, UInt32 su)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> command`	What to Edit.
`-> track1`	Source track.
`-> track2`	Destination track (Use "move" alone).
`-> su`	Sound unit (Use "devide" alone). HMS/SU.

Result

`errNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	

Comments If delete (as `command`) is executed, the data on Memory Stick media is erased.

Playback control I/F

MsaPlay

Purpose Starts to replay.

Prototype
```
Err MsaPlay (UInt16 msaLibRefNum, UInt16 currenttrack,
 UInt32 currentposition, Uint32 pbrate)
```

Parameters

`-> msaLibRefNum`	The reference number of MSA Lib.
`-> currenttrack`	The track number to replay.
`-> currentposition`	The starting position to replay.

	`-> pbrate`	The replay speed.

Result

`errNone`	No error
`msaErrNotOpen`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	

Comments

If `currenttrack` is `0xffff`, current holding track and position are used. If `msaPBRate` is `0xFFFFFFFF`, current holding information is used.
During the replay, 0 is set on AutoOffTimer (=never power off).
Using the call of this function, the replay command to the audio replay system is issued. To know the actual success, it's recommended to check the replay is in the status. If error, the event is issued.

MsaStop

Purpose

Stops to replay.

Prototype

```
Err MsaStop (UInt16 msaLibRefNum, Boolean reset)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> reset`	True if the current status:PBStatus is clear. False if current status: `PBStatus` is remained.

Result

`errNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	

Comments

If stopped, AutoOffTimer is available.

Initial value of `PBStatus`

`status:`	`msa_STOPSTATUS`
`pbRate:`	Normal speed `Dir 0 DecSU 6 InvSU 6`
`currentTrackNo:`	`1`

```
currentSU:        0
```

MsaSetControlKey

Purpose Specifies a Virtual key.

Prototype
```
Err MsaSetControlKey(UInt16 msaLibRefNum,
 MsaControlKey controlkey, MsaControlKeyState keystatus)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> controlkey`	Types of Virtual key.
`-> keystatus`	Status of Virtual key(Set/Release/Long).

Result

`errNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam;`	

Comments If there is a constant interval between Set and Release, it notifies the long key press of VirtualKey to TrackPlayer.

Utility I/F

MsaSuToTime

Purpose Converts sound unit number to `MsaTime` structure.

Prototype
```
Err MsaSuToTime(UInt16 msaLibRefNum,UInt32 SU,
 MsaTimePtr timeP)
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> SU`	Sound unit number from the top track.
`<- timeP`	Pointer to MsaTime structre.

Result

`ErrNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	

Comments **MsaTimeToSu**

Purpose Converts `MsaTime` structure to sound unit number.

Prototype
```
Err MsaTimeToSu(UInt16 msaLibRefNum, MsaTimePtr timeP,
UInt32 *SU)
```

Parameters

->	`msaLibRefNum`	Reference number of MSA Lib.
->	`timeP`	Pointer to MsaTime structre.
<-	`SU`	Sound unit number out of the top track.

Result

`ErrNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	

Comments **MsaPBListIndexToTrackNo**

Purpose Converts `PBListIndex` number to TrackNo.

Prototype
```
Err MsaPBListIndexToTrackNo(UInt16 msaLibRefNum,
 UInt16 pblistindex, UInt16 *trackno)
```

Parameters

->	`msaLibRefNum`	Reference number of MSA Lib.
->	`pblistindex`	PBListindex number.
<-	`trackno`	TrackNO.

Result

`ErrNone`	No error
`msaErrNotOpen:`	
`msaErrDifferentMode:`	
`msaErrNoMedia:`	
`msaErrInvalidMedia:`	
`msaErrNoAlbum:`	
`msaErrParam:`	

MsaOut API

Data structure

Here is the list of the data structure defined by MsaOut.

MsaOutErr

Error number of MsaOut module.

```
typedef Err MsaOutErr;
#define msaOutErrClass (sonyMsaErrorClass|0x40)
#define msaOutErrNone (0)
#define msaOutErrInvalidParam (msaOutErrClass| 1)
#define msaOutErrBandOutOfRange (msaOutErrClass| 2)
#define msaOutErrLevelOutOfRange (msaOutErrClass| 3)
#define msaOutErrFreqOutOfRange (msaOutErrClass| 4)
#define msaOutErrPatternOutOfRange (msaOutErrClass| 5)
#define msaOutErrAlreadyStopped (msaOutErrClass| 6)
#define msaOutErrAlreadyOpened (msaOutErrClass| 7)
#define msaOutErrAlreadyClosed (msaOutErrClass| 8)
#define msaOutErrClosed (msaOutErrClass| 9)
#define msaOutErrHwr (msaOutErrClass|10)
#define msaOutErrNotSupported (msaOutErrClass|11)
```

Field Descriptions

`MsaOutErrNone` Successfully executed.

`MsaOutErrInvalidParam`
Specified parameter is invalid.
NULL pointer is specified.

`MsaOutErrBandOutOfRange`
Specified band number is out of range.

`MsaOutErrLevelOutOfRange`
Specified level is out of range.

`MsaOutErrFreqOutOfRange`
Specified frequencey number is out of range.

`MsaOutErrPatternOutOfRange`
Specified pattern number is out of range.

`MsaOutErrAlreadyStopped`
It is already stopped.

`MsaOutErrClosed` It is closed.

`MsaOutErrHwr` Hardware error occurred.

`MsaOutErrNotSupported`
Specified function is not supported.

MsaOutOutputMode

Set value of audio output mode.

```
typedef enum {
  msaOutOutputStereo = 0,
  msaOutOutputMonoral,
```

```
    msaOutOutputMain,
    msaOutOutputSub,
    msaOutOutputDual
  } MsaOutOutputMode;
```

Field Descriptions

`msaOutOutputStereo` Stereo output.

`msaOutOutputMonoral` Monaural output.

`msaOutOutputMain` Main sound output.

`msaOutOutputSub` Sub sound output.

`msaOutOutputDual` Dual sounds output.

MsaOutMuteSwitch

Set value of mute mode.

```
typedef enum {
  msaOutMuteOFF = 0,
  msaOutMuteON
} MsaOutMuteSwitch;
```

Field Descriptions

`msaOutMuteOFF` Mute is OFF.

`msaOutMuteON` Mute is ON.

MsaOutInfoType

set value/ status

```
typedef struct {
  MsaOutOutputMode outputMode;
  UInt16 volumeL;
  UInt16 volumeR;
  UInt16 volumeLimitL;
  UInt16 volumeLimitR;
  MsaOutMuteSwitchType muteSwitch;
  MsaOutEQSwitchType EQSwitch;
  UInt16 *EQvalueP;
  UInt16 BBLevel;
  UInt16 beepLevel;
} MsaOutInfoType, *MsaOutInfoPtr;
```

Field Descriptions

`outputMode` Audio output mode.

`volumeL` Volume of channel L.

`volumeR` Volume of channel R.

`volumeLimitL`	`Maximum v`olume of channel L.
`volumeLimitR`	`Maximum v`olume of channel R.
`muteSwitch`	Mute status.
	`MsaOutMuteON` Mute is ON.
	`MsaOutMuteOFF` Mute is OFF.
`EQSwitch`	State of EQualizer switch.
`EQvalueP`	Pointer to the value table of EQualizer level.
`BBLevel`	Bassboost level.
`beepLevel`	Beep level.

MsaOutCapabilityType

Audio/beep output control capability information

```
typedef struct {
#define msaOutIncapable (0)
#define msaOutCapable (1)
   UInt32 monoral:1;
   UInt32 bilingual:1;
   UInt32 volumeL:1;
   UInt32 volumeR:1;
   UInt32 volumeLLimit:1;
   UInt32 volumeRLimit:1;
   UInt32 deEmphasis:1;
   UInt32 mute:1;
   UInt32 EQ:1;
   UInt32 EQL:1;
   UInt32 EQR:1;
   UInt32 BB:1;
   UInt32 beep:1;
   UInt32 levelL:1;
   UInt32 levelR:1;
   UInt32 spectrumL:1;
   UInt32 spectrumR:1;
   UInt32 reservedFlag:15;
   UInt16 volumeReso;
   UInt16 volumeLimitReso;
   UInt16 volumeLimitForAVLS;
   UInt16 volumeDefault;
   UInt16 EQReso;
   UInt16 EQNumBand;
   UInt16 BBMaxLevel;
```

```
    UInt16 beepMaxLevel;
    UInt16 beepMaxFreq;
    UInt16 beepMaxPattern;
    UInt16 levelReso;
    UInt16 spectrumReso;
    UInt16 spectrumNumBand;
  } MsaOutCapabilityType, *MsaOutCapabilityPtr;
```

Field Descriptions

`monoral:1` — Monaural output is:

`msaOutCapable` available.

`msaOutIncapable` unavailable.

`bilingual:1` — Main/sub sounds switching is:

`msaOutCapable` available.

`msaOutIncapable` unavailable.

`volumeL:1`

`volumeR:1` — Audio volume control (channel L/channel R) is:

`msaOutCapable`
available.

`msaOutIncapable`
unavailable.

If both channels are `msaOutIncapable`, a device does not have audio volume control function.
If only channel R is `msaOutIncapable`, audio volume will be controled by channel L.

`volumeLLimit:1`

`volumeRLimit:1` — Audio maximum volume control(channel L/channel R) is:

`msaOutCapable`
available.

`msaOutIncapable`
unavailable.

If both channels are `msaOutIncapable`, a device does not have this function.
If only channel R is `msaOutIncapable`, audio maximum volume will be controled by channel L.

`mute:1` — Mute control is:

`msaOutCapable`
available.

`msaOutIncapable`
unavailable.

`EQ:1`	Reserved.
`EQL:1`	Reserved.
`EQR:1`	Reserved.
`BB:1`	Reserved.
`beep:1`	Reserved.
`levelL:1`	

`levelR:1`	Audio output level retrieval function(channel L/channel R) is:
	`msaOutCapable`
	available.
	`msaOutIncapable`
	unavailable.
	If both channels are `msaOutIncapable`, a device does not have this function.
	If only channel R is msaOutIncapable, central value/average of audio output level will be obtained from channel L.

`spectrumL:1`	

`spectrumR:1`	Spectrum data retrieval funtion(channel L/channel R) is:
	`msaOutCapable`
	available.
	`msaOutIncapable`
	unavailable.
	If both channels are `msaOutIncapable`, a device does not have this function.
	If R channel is `msaOutIncapable`, central value/average of spectrum data will be obtained from channel L.

`volumeReso`	Set resolution of audio volume: `1` to `0xffff`.
`volumeLimitReso`	Set resolution of maximum audio volume: `1` to `0xffff`.
`volumeLimitForAVLS`	Recommended volume set value of AVLS function: `0` to `volumeReso-1`.
`volumeDefault`	Volume set value at default: `0` to `volumeReso-1`.
`EQReso`	Reserved.
`EQNumBand`	Reserved.
`BBMaxLevel`	The maximum level number of Bass boost.
`beepMaxLebel`	Reserved.

`beepMaxFreq`	Reserved.
`beepMaxPattern`	Reserved.
`levelReso`	Received resolution of audio output peak level: `1` to `0xffff`.
`spectrumReso`	Received resolution of spectrum data: `1` to `0xffff`.
`spectrumNumBand`	Number of bands of spectrum data: `0` to `32`.

MsaOutBeepPattern Enum

The pattern of the Beep sound which can be set up by `MsaOutStartBeep()` is defined. Some Beep Pattern is not defined but is silent. These may be defined in the future.

```
typedef enum {
  msaOutBeepPatternPlay = 0,
  msaOutBeepPatternStop,
  msaOutBeepPatternPause,
  msaOutBeepPatternAMSp,
  msaOutBeepPatternAMSm,
  msaOutBeepPatternFirst,
  msaOutBeepPatternWarn,
  msaOutBeepPatternErr,
  msaOutBeepPatternSkip,
  msaOutBeepPatternOK,
  msaOutBeepPatternCancel,
  msaOutBeepPatternClick,
  msaOutBeepPatternReset,
  msaOutBeepPattern13,
  msaOutBeepPattern14,
  msaOutBeepPattern15
} MsaOutBeepPattern;
```

Audio output control I/F

Here is the detail specification of audio output control APIs.

MsaOutSetOutputMode

Purpose Set audio output mode.

Prototype

```
MsaOutErr MsaOutSetOutputMode( UInt16 msaLibRefNum,
MsaOutOutputMode mode );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.

	`-> mode`	Specified audio output mode.
		`msaOutOutputStereo`
		stereo output
		`msaOutOutputMonoral`
		monaural output
		`msaOutOutputMain`
		main sound output
		`msaOutOutputSub`
		sub sound output
		`msaOutOutputDual`
		dual sounds output

Result	`msaOutErrNone`	Successfully executed.
	`msaOutErrInvalidParam`	Specified mode is invalid.
	`msaOutErrNotSupported`	The function is not supported.

Comments Audio output mode will be set to the one specified at `mode`.
The mode will be changed immediately, even if performed during playback.

Compatibility Depending on audio output mode control capability information returned from `MsaOutGetCapability()`, a selected mode might be unavailable.

Control capability of PEG-N7x0C are:

- Monaural/stereo output
- Main-sub sounds switching (Haven't yet settled.)

Control capability of Audio Adapter is:

- Unsupported

MsaOutSetVolume

Purpose Set output volume level.

Prototype
```
MsaOutErr MsaOutSetVolume( UInt16 msaLibRefNum, UInt16
lValue, UInt16 rValue );
```

Parameters	`-> msaLibRefNum`	Reference number of MSA Lib.
	`-> lValue`	`Output v`olume level of channel L.
	`-> rValue`	`Output v`olume level of channel R.
Result	`msaOutErrNone`	Successfully executed.

`msaOutErrLevelOutOfRange`
Specified volume is out of range.

`msaOutErrNotSupported`
The function is not supported.

Comments Audio volume will be set to those specified at `lValue` and `rValue`.
Specifiy any of 0 to (resolution-1) to `lValue` for volume of channel L and to `rValue` for volume of channel R.
If specified volume is larger than the maximum set by `MsaOutSetVolumeLimit()`, it will be adjusted to the set maximum.
The volume level will be changed immediately, even if performed during playback.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, the setting of `rValue` or of both `lValue` and `rValue` can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Setting of both L and R channels
- Resolution: 32

MsaOutVolumeUp

Purpose Raise volume by one level.

Prototype
```
MsaOutErr MsaOutVolumeUp( UInt16 msaLibRefNum );
```

Parameters `-> msaLibRefNum` Reference number of MSA Lib.

Result `msaOutErrNone` Successfully executed.

`msaOutErrLevelOutOfRange`
Specified volume level is out of range.

`msaOutErrNotSupported`
The function is not supported.

Comments Audio volume will be turned up by one resolution.
Even if a device allows individual setting of channels L and R, the volume of these channels will be turned up at the same time.
If the volume of either channel L or R is larger than that set by `MsaOutSetVolumeLimit()`, it will be adjusted to the set maximum(`msaOutErrLevelOutOfRange`).
The volume level will be changed immediately, even if performed during playback..

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, this function call can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Setting of both L and R channels.
- Resolution: 32

MsaOutVolumeDown

Purpose Turn down the volume by one level.

Prototype
```
MsaOutErr MsaOutVolumeDown( UInt16 msaLibRefNum );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrLevelOutOfRange`	Specified volume level is out of range.
`msaOutErrNotSupported`	The function is not supported.

Comments Turns down audio volume by one resolution.
Even if a device allows individual setting of channels L and R, the volume of these will be turned down at the same time.
If the volume of either channel L or R is set to 0, it will remaine at 0. `(msaOutErrLevelOutOfRange)`.
The volume level will be changed immediately, even if performed during playback.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, this function call can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Setting of both L and R channels.
- Resolution: 32

MsaOutSetVolumeLimit

Purpose Set maximum volume.

Prototype
```
MsaOutErr MsaOutSetVolumeLimit( UInt16 msaLibRefNum, UInt16
lLimit, UInt16 rLimit );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> lLimit`	Maximum volume level of channel L.

	`-> rLimit`	Maximum volume level of channel R.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrLevelOutOfRange`	Specified volume is out of range.
`msaOutErrNotSupported`	The function is not supported.

Comments

Sets maximum volume to those set at `lLimit` and `rLimit`.
Even if specified maximum volume is larger than that set by `MsaOutSetVolume()`, it will be set as specified.
Specify any of 0 to (resolution-1) to `lLimit` for channel L and to `rLimit` for channel R.
The volume level will be changed immediately, even if performed during playback..

Compatibility

Depending on volume control capability information returned from `MsaOutGetCapability()`, the setting of `rLimit` or of both `lLimit` and `rLimit` can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Setting of both L and R channels.
- Resolution: 32

MsaOutSetMute

Purpose

Set mute status.

Prototype

```
MsaOutErr MsaOutSetMute( UInt16 msaLibRefNum,
MsaOutMuteSwitchType switch );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> switch`	Mute status is:
	`msaOutMuteON` ON.
	`msaOutMuteOFF` OFF.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Specified mute status is invalid.
`msaOutErrNotSupported`	The function is not supported.

Comments Enables audio mute by using switch.
The mute status wil be changed immedately, even if performed during playback.

Compatibility Depending on mute control capability information returned from `MsaOutGetCapability()`, the setting of `switch` can be invalid.

Control capability of PEG-N7x0C and Audio Adapter are:

- Mute function.

Beep output control I/F

MsaOutSetBBLevel

Purpose Set Bass Boost function.

Prototype
```
MsaOutErr MsaOutSetBBLevel( UInt16 msaLibRefNum, UInt16
level );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> level`	Level of Bass Boost

Result

`errNone`	Success
otherwise	Failure

Comments Applied to L and R channel.

Compatibility `level` must not exceed the value of the maximum Bass Boost level obtained from `MsaOutGetCapability()`.

Control capability of PEG-N7x0C are:

- Unsupported

Control capability of Audio Adapter is:

- Usable level value: 0 or 1

MsaOutStartBeep

Purpose Sound the beep.

Prototype
```
MsaOutErr MsaOutStartBeep( UInt16 msaLibRefNum, UInt16 freq,
MsaOutBeepPattern pattern );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`-> freq`	Specify the frequency[Hz].

	`-> pattern`	Specify the beep pattern.

Result

`errNone`	Success
otherwise	Failure

Comments In Audio Adapter, the music is muted while the beep is sounding.

Compatibility Control capability of PEG-N7x0C are:

- Unsupported

Control capability of Audio Adapter is:

- Usable frequency: 400 - 4kHz
- The number of the maximum patterns: 16

Setting information retrieval I/F

Here is the detail specification of APIs that get setting information.

MsaOutGetOutputMode

Purpose Get current audio output mode.

Prototype

```
MsaOutErr MsaOutGetOutputMode( UInt16 msaLibRefNum,
MsaOutOutputMode *modeP);
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> modeP`	Pointer to memory that store audio output mode.

`msaOutOutputStereo`
Stereo output

`msaOutOutputMonoral`
Monaural output

`msaOutOutputMain`
Main sound output

`msaOutOutputSub`
Sub sound output

`msaOutOutputDual`
Dual sounds output

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.

`msaOutErrNotSupported`
The function is not supported.

Comments Stores audio output mode to the location specified by modeP.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, the information in modeP can be invalid.

Control capabilities of PEG-N7x0C are:

- Monaural/stereo output
- Main-sub sound switching

Control capabilities of Audio Adapter is:

- Unsupported

MsaOutGetVolume

Purpose Get current volume level.

Prototype
```
MsaOutErr MsaOutGetVolume(UInt16 msaLibRefNum, UInt16
*lValueP,UInt16 *rValueP);
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> lValueP`	Pointer to a memory where volume level of channel L is stored.
`<-> rValueP`	Pointer to a memory where volume level of channel R is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores current volume level to locations specified by `lValueP` and `rValueP`, respectively.
Specify any of 0 to (resolution-1) to `lLimit` for volume level of channel L and to `rLimit` for volume level of channel R.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, the setting of `rValueP` or of both `lValueP` and `rValueP` can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Setting of both L and R channels.

- Resolution: 32

MsaOutGetVolumeLimit

Purpose Get current maximum volume set value.

Prototype
```
MsaOutErr MsaOutGetVolumeLimit(UInt16 msaLibRefNum, UInt16
*lLimitP, UInt16 *rLimitP);
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> lLimitP`	Pointer to a memory where maximum volume level of channel L is stored.
`<-> rLimitP`	Pointer to a memory where maximum volume level of channel R is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores current maximum volume level to locations specified by `lLimitP` and `rLimitP`, respectively.
Specify any of 0 to (resolution-1) to `lLimitP` for maximum volume level of channel L and to `rLimitP` for maximum volume level of channel R.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, the setting of both `lLimitP` and `rLimitP` or only `rLimitP` can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- The setting can be made for both L and R channels.
- Resolution: 32

MsaOutGetMute

Purpose Get current mute status.

Prototype
```
MsaOutErr MsaOutGetMute(UInt16 msaLibRefNum,
MsaOutMuteSwitchType *switchP );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> switchP`	Pointer to a memory where current mute status is stored.

`msaOutMuteON`
Mute is ON.

`msaOutMuteOFF`
Mute is OFF.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores current audio mute status to a location specified by `switchP`.

Compatibility Depending on volume control information returned from `MsaOutGetCapability()`, information in `switchP` can be invalid.

Control capabilities of PEG-N7x0C and Audio Adapter are:

- Mute function

MsaOutGetInfo

Purpose Get set values/status in block.

Prototype

```
MSAOurErr MsaOutGetInfo( UInt16 msaLibRefNum, MsaOutInfoType
*infoP );
```

Parameters

->	`msaLibRefNum`	Reference number of MSA Lib.
<-	`infoP`	Pointer to a memory where every set value/status is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.

Comments Stores current set values/status to a location specified by `infoP`.

Compatibility It depends on volume control capability information of a particular function returned from `MsaOutGetCapability()`.

Audio output information retrieval I/F

Lists the detailed specification of APIs that get audio output peak information.

MsaOutGetLevel

Purpose Get output peak level.

Prototype

```
MsaOutErr MsaOutGetLevel( UInt16 msaLibRefNum, UInt16
*lValueP, UInt16 *rValueP );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> lValueP`	Pointer to a memory where output peak level of channel L is stored.
`<-> rValueP`	Pointer to a memory where output peak level of channel R is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	Null pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores current output level to locations specified by `lValueP` and `rValueP`. Specify any of 0 to (resolution-1) to `lValueP` as output peak level of channel L and `rValueP` as output peak level of channel R.

Compatibility Depending on volume control capability information returned from `MsaOutGetCapability()`, the information in `rValueP` or in both `lValueP` and `rValueP` can be invalid.

Control capability of PEG-N7x0C are:

- Setting of both L and R channels
- Resolution: 16

Control capabilities of Audio Adapter is:

- Unsupported

MsaOutGetSpectrum

Purpose Get spectrum data.

Prototype
```
MsaOutErr MsaOutGetSpectrum( UInt16 msaLibRefNum, UInt16
*lValurP, UInt16 *rValueP );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> lValueP`	Pointer to a memory where spectrum data of channel L is stored.
`<-> rValueP`	Pointer to a memory where spectrum data of channel R is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	NULL pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores spectrum data of all bands to a location specified by `lValueP` and `rValueP`, respectively.
Specify any of 0 to (resolution-1) to `lValueP` as spectrum data of channel L and `rValueP` as spectrum data of channel R.

Compatibility Depending on volume control information obtained by `MsaOutGetCapability()`, the information in `rValueP` or in both `lValueP` and `rValueP` can be invalid.

Control capabilities of PEG-N7x0C are:

- Setting of both L and R channels.
- Number of bands: 8
- Resolution 16

Control capabilities of Audio Adapter is:

- Unsupported

System I/F

Lists the detailed specification of APIs of MsaOut system.

MsaOutGetCapability

Purpose Get audio/beep output capability information.

Prototype

```
MsaOutErr MsaOutGetCapability( UInt16 msaLibRefNum,
MsaOutCapabilityType *capabilityP );
```

Parameters

`-> msaLibRefNum`	Reference number of MSA Lib.
`<-> capabilityP`	Pointer to a memory where control capability information is stored.

Result

`msaOutErrNone`	Successfully executed.
`msaOutErrInvalidParam`	NULL pointer is specified.
`msaOutErrNotSupported`	The function is not supported.

Comments Stores audio/beep output control and status retrieval capabilities to a location specified by `capabilityP`.

Notes

Determining If Memory Stick Audio Library Is Available

To determine if the Memory Stick audio library is available on a device, as shown in "Availability of library", check `sonySysFtrInfoLibrMsa` bit in `Libr` field for `SonySysFtrSysInfoType` obtained by using `sonySysFtrNumSysInfoP` as a feature number.[1]

Power Auto-Off

During playback (including background playing), Auto-Off is set to Forever. So, your application does not need to disable Auto-Off while `MsaPlay` is executed.

1. Some other way of device detection may be provided in the future.

9

Audio remote control : Sony Rmc Library

It is a library for using more highly the audio remote control which can be used only as a key event in usual.[1]

Audio remote control API

Data structure

RmcRegEnum

Priority processing level of a callback function registered using **`RmcRegister()`** is defined as below:

```
typedef enum RmcRegisterEnum {
   rmcRegTypeWeak,
   rmcRegTypeStrong
} RmcRegEnum;
```

Field Descriptions

`rmcRegTypeWeak`	Indicates low priority processing level. A callback function registered in this level can be stopped temporarily by another application using `RmcDisableKeyHandler()`.
`rmcRegTypeStrong`	Indicates high priority processing level. A callback function registered in this level cannnot be stopped by another application using `RmcDisableKeyHandler()`.

1. Using with Audio Adapter is not recommended.

RmcStatusType

The structure used to get the status of audio remote control library by `RmcGetStatus()`.

```
typedef struct{
  UInt32 creatorID;
  UInt32 reserved;
} RmcStatusType;
```

Field Descriptions

`creatorID`	CreatorID of an application which registered a callback function.
`reserved`	Reserved. Not usable.

RmcKeyCodeEnum

Key identification number which will be returned from `GetRmcKey()` macro whenever an operation was performed using PEG-N700C-supplied remote control.

```
typedef enum {
  rmcKeyOther = 0, // Unknown keys
  rmcKeyPlay,    // Play
  rmcKeyFrPlay,  // FR/Play
  rmcKeyFfPlay,  // FF/Play
  rmcKeyStop, // Stop
  rmcKeyDown, // Down
  rmcKeyUp,   // Up
  rmcKeyNum   // Num of all RMC keys
} RmcKeyCodeEnum;
```

Field Descriptions

`rmcKeyOther`	Button which will not occur by using supplied remote control
`rmcKeyPlay`	Play button
`rmcKeyFrPlay`	FR Play button
`rmcKeyFfPlay`	FF Play button
`rmcKeyStop`	Stop button
`rmcKeyDown`	Volume Down button
`rmcKeyUp`	Volume Up button
`rmcKeyNum`	Number of buttons on supplied remote control

Audio remote control functions

RmcLibOpen

Purpose Start to use the audio remote control library.

Prototype
```
Err RmcLibOpen ( UInt16 refNum )
```

Parameters

`-> refNum`	Reference number of the audio remote control library.

Result

`errNone`	No error
`rmcErrNotAvailable`	Audio remote control is not available.
`memErrNotEnoughSpace`	Insufficient memory

Comments Does processing to open the audio remote control library.

RmcLibClose

Purpose Closes the audio remote control library.

Prototype
```
Err RmcLibClose ( UInt16 refNum )
```

Parameters

`-> refNum`	Reference number of audio remote control library

Result

`errNone`	No error
`rmcErrNotOpen`	Audio remote control library hasn't opened yet.
`rmcErrStillOpen`	Audio remote control library is still opened.

Comments It performs the procedure to complete audio remote control library.

RmcRegister

Purpose Register function which will be called back every time audio remote control-related event is issued.

Prototype
```
Err RmcRegister(UInt16 refNum, RmcRegEnum type,
RmcKeyHandleProcPtr callbackP, UInt32 creatorID)
```

Parameters

`-> refNum`	Library reference number
`-> type`	Priority processing level of registered function

	`-> callbackP`	Pointer to callback function
	`-> creatorID`	CreatorID of registered application
Result	`errNone`	No error.
	`rmcErrNotOpen`	Audio remote control library hasn't opened yet.
	`rmcErrRegister`	The function is already registered by another application.

Comments To unregister a particular callback function, put NULL into `RmcKeyHandleProcPtr` and call the function.
Regardless of type, only one callback function can be registered to a library. Overwriting is not allowed.
This function is generally used by an application that wants to get remote control event even after it is finished. In that case, data base where a specified callback function is stored must remain locked.
Be sure not to delete an application which registered a function, or fatal error will occur.
Note that function call of those registered using `rmcRegTypeStrong` cannot be cancelled by `RmcDisableKeyHandler()`.

RmcDisableKeyHandler

Purpose Stops calling a registered call back function.

Prototype
```
Err RmcDisableKeyHandler(UInt16 refNum)
```

Parameters	`-> refNum`	Reference number of the library
Result	`errNone`	No error
	`rmcErrNotOpen`	Audio remote control library hasn't opened yet.
	`rmcErrRegister`	Registered with `rmcRegTypeStrong`.

Comments In general, when an application on the back ground continues to obtain remote control events, this function enables an application on the foreground to obtain them. But a calling can be stopped only when the corresponding call back function is registered as `type = rmcRegTypeWeak` by `RmcRegsiter()`. If the calling of that function is stopped with this function, make sure to call it again by `RmcEnableKeyHandler()`before finishing the application.

RmcEnableKeyHandler

Purpose Restarts to call a registered call back function.

Prototype
```
Err RmcEnableKeyHandler(UInt16 refNum)
```

Parameters

`-> refNum`	Reference number of the library

Result

`errNone`	No error
`rmcErrNotOpen`	Audio remote control library hasn't opened yet.
`rmcErrRegister`	Already available for calling.

Comments Usually, it's used along with `RmcDisableKeyHandler()`.

RmcGetStatus

Purpose Obtains the library status.

Prototype
```
Err RmcGetStatus(UInt16 refNum, RmcStatusType *status)
```

Parameters

`-> refNum`	Reference number of the library
`<-` `status`	Pointer to `RmcStatusType`

Result

`errNone`	No error
`rmcErrNotOpen`	Audio remote control library hasn't opened yet.

Comments The application can determine whether call back function is registered on its own by the returned value to the `creatorID` field of `status`.

RmcKeyRates

Purpose Specifies or obtains the timing of remote control event.

Prototype
```
Err RmcKeyRates(UInt16 refNum, Boolean set,
 UInt16 *initDelayP, UInt16 *periodP)
```

Parameters

`-> refNum`	Reference number of the library
`->` `set`	Set to true if it's specified. False if it obtains the current value.
`->` `initDelayP`	The amount of time of the initial delay till auto repeat in system tick.

	`-> periodP`	Auto repeat period, in system tick.
Result	`errNone`	No error
	`rmcErrNotOpen`	Audio remote control library hasn't opened yet.

Comments Usually the application doesn't use it.

The constants defined by an application

RmcKeyHandleProcPtr

Purpose Handles remote control key events.

Prototype
```
void (*RmcKeyHandleProcPtr)(KeyDownEventType *keyDown)
```

Parameters	`-> keyDown`	Event structre defined by PalmOS. See PalmOS documents for your reference.

Result Returns nothing.

Comments It is called when audio remote control event is issued except that the calling is stopped by `RmcDisableKeyHandler()`.

It starts up from `SysHandleEvent()`. In this case, `SysHandleEvent()`returns true.

Note

Determining If Audio Remote Control Library Is Available

To determine whether the audio remote control library is available on a device, as shown in "Availability of library", check `sonySysFtrSysInfoLibrRmc` bit in libr field for `SonySysFtrSysInfoType` obtained by using `sonySysFtrNumSysInfoP` as a feature number.[1]
To determine whether the audio remote control library is available on a device, check `sonySysFtrSysInfoLibrRmc` bit in libr field for `SonySysFtrSysInfoType` obtained by using `sonySysFtrNumSysInfoP` as a feature number.
For more information about event support in the system , see "Audio Remote Control".

1. Some other way of device detection may be provided in the future.

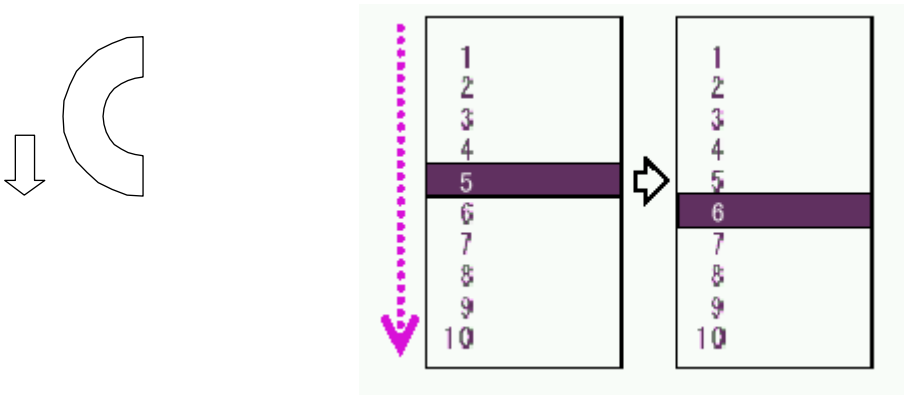
A

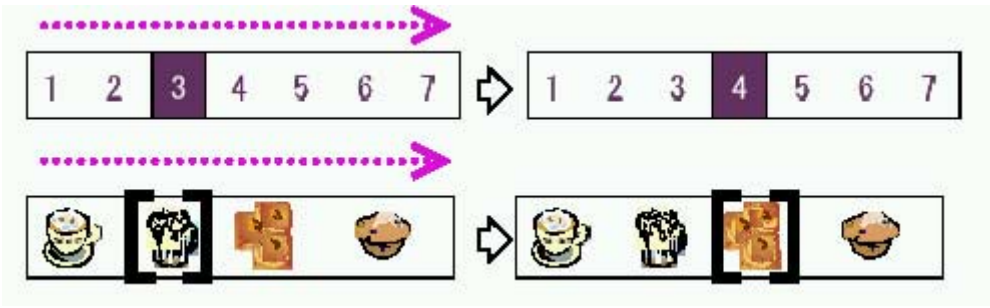
User Interface Guideline

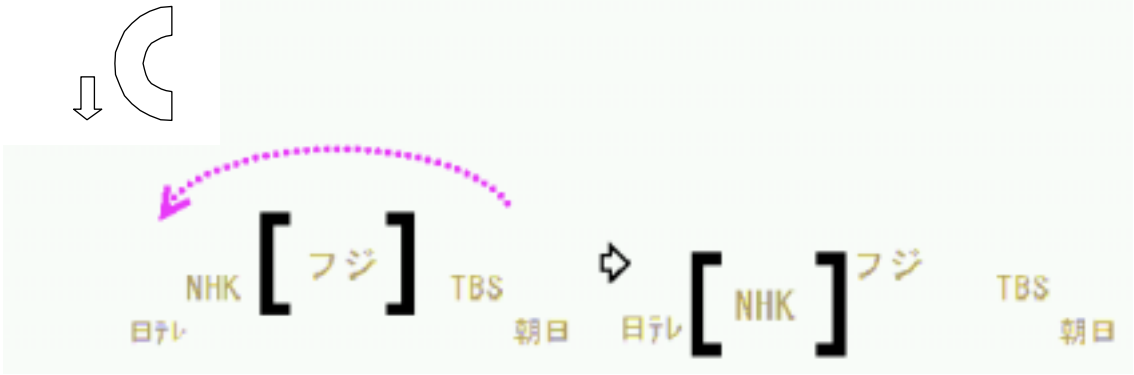
This is a guideline for developers who want to use the Jog Dial navigator in their applications. Users should expect the Jog Dial navigator to influence programs in similar ways. By following these guidelines, developers can ensure that their application's user interface responds to the Jog Dial navigator appropriately.

- When continuing to press the Jog Dial navigator and then releasing, `vchrJogPush` is executed with the initial first press and `vchrJogRelease` is executed upon release. Unless the application is a kind of launcher, both actions are basically considered as an Enter function, however it is recommended to use it as an Enter function when the Jog Dial navigator is pressed down rather than released unless continuing to press the Jog Dial navigator down has a special purpose. In the case of a launcher application, it is recommended to use `vchrJogRelease` as an Enter function.
- It's possible to add new meanings: When the Jog Dial navigator is rotated clockwise(`vchrJogUp` is issued), this will mean “Increase.” When it is rotated counter-clockwise(`vchrJogDown` is issued), this will mean “decrease.” Those are for the volume adjustment of audio player and other purposes.
- When a Back key is pressed, `vchrJogBack` is issued. Since this code is designed for the system use, including JogAssist, the use on the application is banned in general. However, in case using the application, make sure to program it to behave the same way as JogAssist. (see JogAssist processing)
- We distinguish between two types of scrolling. The first type is when the background remains in place, but the cursor moves around on screen. When the Jog Dial navigator is rotated counter-clockwise, `vchrJogDown` is called. When it is rotated clockwise, vchrJogUp is called. In this case, when `vchrJogDown` is called, the cursor's position should be moved from the top to the bottom of a vertical list that indicates items, or from left to the right of a horizontal list. The opposite scrolling should occur in the case of a `vchrJogUp`

call. In the case of a circular list, the cursor should be moved in the same direction as the Jog Dial navigator while the list/wheel holds its position.

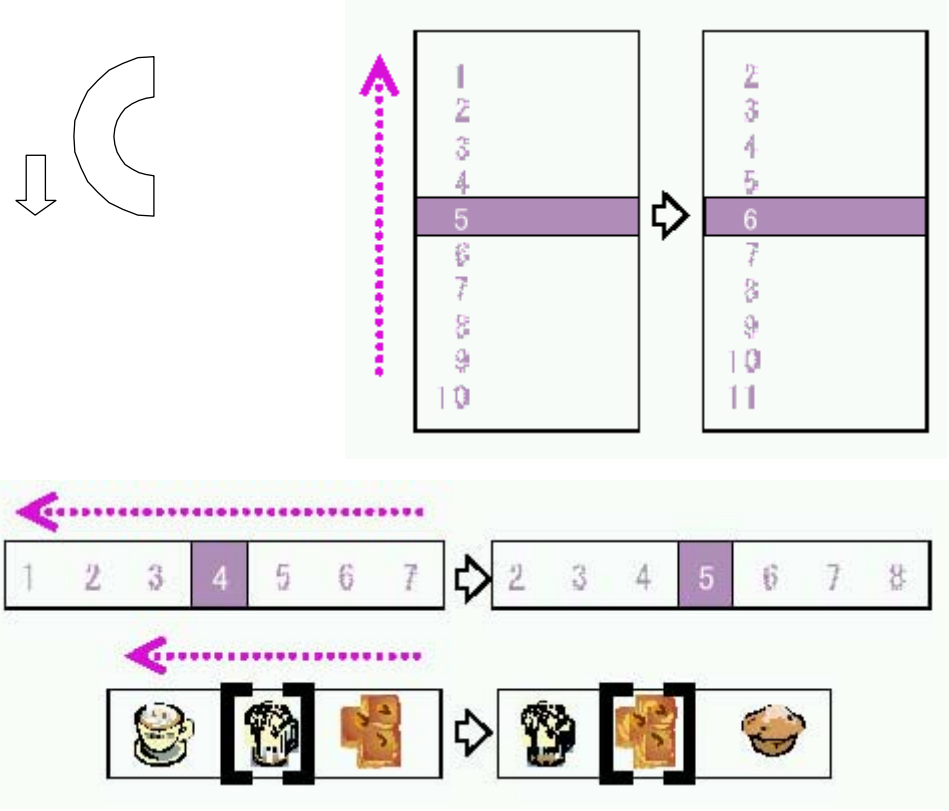






- The second type of scrolling is when the cursor remains fixed onscreen while the background scrolls behind it (for example, when the cursor is at the bottom of a page, and the user scrolls down). In this case, when the Jog Dial navigator is rotated counter-clockwise and vchrJogDown is called, a vertical list of items

should be scrolled up, and a horizontal list should be scrolled from right to the left. When Jog Dial navigator is rotated clockwise, vchrJogUp is called and all movement is the opposite as mentioned above. In the case of a circular list, the list/wheel will rotate behind the cursor in the same rotate direction as the Jog Dial navigator.



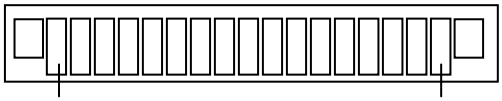
B

External Interface

This is a reference of external interface. For more details, see CLIÉ™ developer site <http://www.us.sonypdadev.com/>. Note that some devices have no external interface. Additionally, this is designed to explain the equipment loaded into the CLIÉ™. There is no guarantee that all of the developed device based on this reference will connect properly.

Cradle interface

Pin Specification

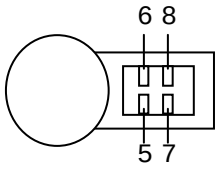


Pin No	Name	In brief
1	USB D-	USB Data-
2	USB D+	USB Data+
3	DTR	Data Terminal Ready
4	RXD	Receive Data
5	RTS	Request to Send
6	TXD	Transmit Data
7	CTS	Clear to Send
8	NC	-
9	DC_B+	Power terminal post
10	HOT SYNC	Hot Sync

11	UNREG OUT	Power supply
12	CNT	Accessory detection
13	GND	Ground

Audio remote control interface

Pin Specification



Pin No	Name
5	GND
6	KEY
7	DATA(NC)
8	B+(2.5V)

Index

R

S

V