SONY

Programmer's Companion for Sony CLIÉ™ Handheld

CLIÉ Software Development Kit Release 5
for Palm OS 5

CLIÉ

Table of Contents

Table of Contents 3

Introduction 7

Purpose of this manual 7
How to read this manual 7
CLIÉ™ SDK Components 7
  Directory components 7
  Header file 8
Software Development Environment 9
  CodeWarrior for Palm 9
  Palm OS SDK Version 5 9
  Palm OS 5 Simulator 9
Installing CLIÉ™ SDK 9
  Copying SDK 9
  Adding an access path 9
  Adding a header file 10
History 10

Part I : System Function 11

1 Palm OS® System Features 13

Features 13
  Feature Creator 13
  Feature number 13
Device Detection 14
  How to distinguish the CLIÉ™ Handheld 14

2 Jog Dial™ Navigator 15

Jog Event 15
  Virtual key 15
  Event interval 16
  Event processing 17

3 Hardware: Left and Right Buttons 19

Key Event 19
  Virtual Key 19
Compatibility 20

4 JogAssist 23

Features 23

JogAssist feature control 24
JogAssist processing 25
vchrJogBack Assist 25
vchrJogUp/Down Assist 26
vchrJogPushedUp/PushedDown Assist 28
vchrJogPush/PushRepeat/Release Assist 28
vchrJogLeft/vchrJogRight Assist 29
JogAssist Mask Specification 30
JogAssist Mask Data 30
JogAssist Mask Pointer 32
JogAssist Mask Owner 32
Support to JogAssist mask system. 33
Notes 33
Mask Setting 33

5 Audio Remote Control 35

Remote Control Event 35
Virtual Key 35
Event intervals 36
Event processing 38
Notes 39
Auto-On 39
Application of Remote Control Interface 39

Part II : Library 41

6 High Resolution: Sony HR Library 43

Overview 43
Screen mode and API 43
Terminology 43
Incompatibilities with the existing API. 44
Working with Existing API. 47
Font Selection. 54
Screen Window and Off Screen Window 56
Using High Resolution. 60
Loading the Library 60
Changing Screen Modes 61
High resolution API 64
System I/F Functions 64
Window APIs 65
Bitmap APIs 86
Font APIs 87
Compatibility with PalmOS 5 installed CLIÉ™s 88

Areas incompatible with the past 88
APIs that handle font size 89
Drawing with SmallFont/TinyFont fonts 94
Drawing using Patterns 94
Modified drawing as implemented in PalmOS 5 95
Notes 95
Sublaunching 95
Switching screen mode 95
BmpCompress 95
About HighRes Assist 96

7 Virtual Silkscreen: Virtual Silkscreen Manager 99

Overview 99
Feature Specifications 99
Terminology 99
Display Structure 101
User Interface Specifications 102
Resize 102
Full Screen Display 103
Using the Virtual Silkscreen 104
Functional Environment 104
Resize 105
Application Switching 106
Guidelines 107
Special issues 108
Silk Manager Reference 109
Virtual Key 109
Notifications 109
Silk Manager API 111
Silk Manager API (For Version 1 compatibility) 117

8 Audio remote control : Sony Rmc Library 120

Audio remote control API 120
Data structure 120
Audio remote control functions 122
The constants defined by an application 125

9 JPEG Utility: Sony JpegUtil Library 126

Function specifications 126
Function list 126
Using the JPEG utility 127
Loading the library 127
Relation of JPEG utility to Resolution 127
Encoding/decoding progress display and cancel notification 129

JPEG Utility API 131
Data Structures. 131
System I/F API 136
Utility API 137
Utility API (Parts added from Ver.2.0) 144
Notes. 151
Usage example. 151

A Jog Dial navigator User Interface Guideline 154

B External Interface 158

Interface Connector. 158
Pin Specification(PEG-NZ, NX, TG, TJ Series) 158
Pin Specification(PEG-S, PEG-N Series). 159
Audio remote control interface 160
Pin Specification 160

C CF Memory Card 162

Using VFS Manager 162
Notes. 164
Usage notes 164

Index 166

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.

CLIÉ™ SDK Components

Directory components

The CLIÉ™ SDK Release 5.0 is composed of the following directories

```
Sony SDK Support\
 └ R5.0\
    ├ 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
    │  │  └ etc\        Stores the header file for Silk Plug-In of the CLIÉ™
    │  │                 SDK
    │  └ Libraries\     Stores the library related header file of the CLIÉ™
    │                    SDK
    └ Samples\          Stores the sample program 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.
`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.

System\etc Directory

`SonySlkw.h`	Header file for Silk PlugIn use.

Library Directory

`SonyLibPublic.h`	The library related header files are integrated in this file.
`SonyHRLib.h`	For High-resolution library.
`SonyRmcLib.h`	For audio remote control library.
`SonySilkLib.h`	For Virtual Silkscreen library.
`SonyJpegUtilLib.h`	For JPEG Utility library.

Software Development Environment

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

CodeWarrior for Palm

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 Version 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.palmsource.com/>.

Palm OS 5 Simulator

The Palm OS Simulator software that simulates the CLIÉ™ as a Palm OS platform device. It simulates a Palm OS device, and through additional features such as error check and debug, it is possible for application tests to proceed efficiently before performing them on a real machine. The simulator can be downloaded from the CLIÉ™ development site at <http://www.cliedeveloper.com/>.

Installing CLIÉ™ SDK

Copying SDK

Copy directory structure under `Sony SDK Support` to CodeWarrior SDK directory (example: C:\Program Files\Metrowerks\CodeWarrior\Other SDKs\Sony).

Adding an access path

To add a path to allow access to CLIÉ™ SDK header files using CodeWarrior for Palm Version 8:

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 “Other SDKs\Sony\Sony SDK Support\R5.0” under CodeWarrior directory and click [OK] button.

4. Check that “{Compiler}Other SDKs\Sony\Sony SDK Support\R5.0” 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 1.0β	– available on PEG-NX70V, NX60 [2002/10/29]
Version 1.0	– adds SilkSample and HRlibSample [2002/12/05]
Version 1.0.1	– modifies codes at “JogAssist feature control” [2003/01/31]
Version 1.1	- adds Appendix C, “CF Memory Card.” [2003/06/01]
Version 1.2	- adds Version 3 specification to Chapter 7, “Virtual Silkscreen: Virtual Silkscreen Manager.” [2003/08/01]
Version 1.3	- adds Chapter 3, “Hardware: Left and Right Buttons.” [2003/10/14]
Version 1.4	- adds type of Jog Dial at “`sonySysFtrNumSysInfoP`” [2004/02/10]

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.)

Please refer to Chapter 4, "Features." for the details of the following feature numbers.

```
sonySysFtrNumJogAstControlP
sonySysFtrNumJogAstMaskP
sonySysFtrNumJogAstMOCardNoP
sonySysFtrNumJogAstMODbIDP
```

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`

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 ... */
}
```

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.
`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.
`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.

`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.
`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)

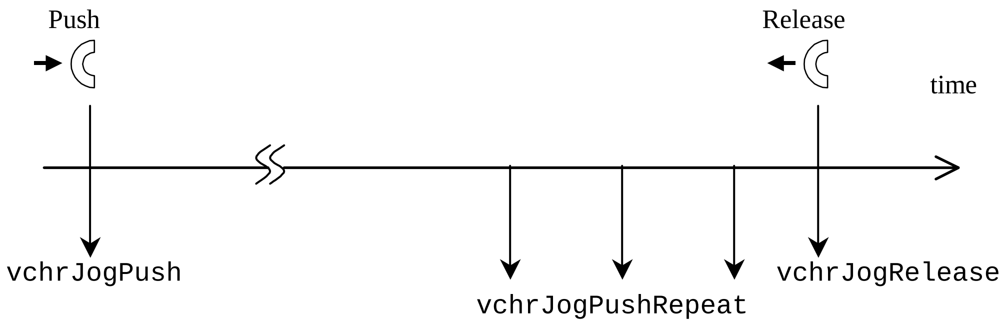
NOTE: Note that this event key is made for the system and not for an application. In case of use, the processing should conform to the guideline to keep user interface consistent.
The code might be processed by the system extension so your application should not assume this event will be issued.

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

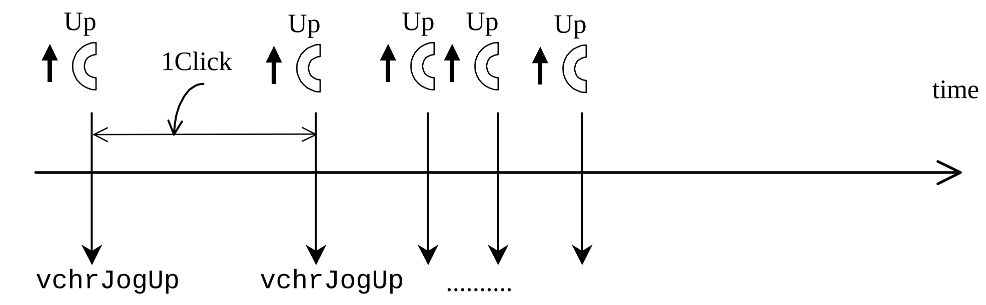
Now, we will show you how the issued events are related to one another.

1. Push/PushRepeat/Release



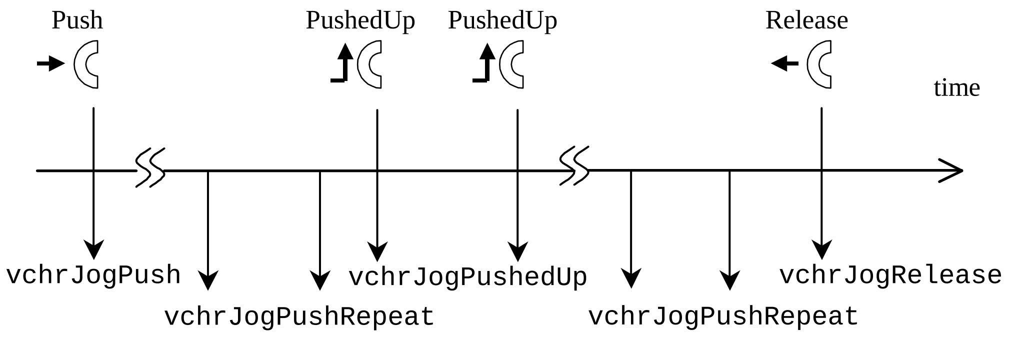
If the Jog Dial navigator has kept pressed down, after the initial delay the first `vchrJogPushRepeat` is issued, after which `vchrJogPushRepeat` is issued at set intervals.

2. Up(Down)



`vchrJogUp` is generated whenever rotating the Jog Dial navigator one time.

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, if the button is pressed and kept still for only for the initial delay period, `vchrJogPushRepeat` will be generated again afterward.

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) {
```

1. On some divice, `vchrJogPushRepeat` is issued.

```
      /* do ‘Up’ */
    } else if (eventP->data.keyDown.chr == vchrJogDown) {
      /* do ‘Down’ */
    } else {
      ...
    }
  }
}
```

3

Hardware: Left and Right Buttons

This chapter describes the actions of the left and right buttons in certain CLIÉ™ Handheld models.

Key Event

Virtual Key

A virtual `keyDownEvent` occurs when the left or right button is pressed. The key is identified by the `chr` value in the `KeyDownEventType` data fields and when the `commandKeyMask` bit in the `modifier` field has been set. Refer to Palm OS® documentation for more details about `keyDownEvent`.

The `chr` values for the left and right buttons are listed as follows:

`vchrJogLeft`	Occurs when the left button is pressed. Press and hold to set the `autoRepeatKeyMask` bit in the `modifiers` field and generate a continuous `keyDownEvent`
`vchrJogRight`	Occurs when the right button is pressed. Press and hold to set the `autoRepeatKeyMask` bit in the `modifiers` field and generate a continuous `keyDownEvent`

Compatibility

The availability of the left and right buttons can be determined from the `sonySysFtrNumSysInfoP` feature pointer.

sonySysFtrNumSysInfoP

The pointer to `SonySysFtrSysInfoType` structure stores system information, such as hardware features. The pointer value does not change, even after a reset. Memory that is currently selected by the pointer, cannot be overwritten.

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 Memory Stick® media
status */
          UInt32 rsv32_10;

          UInt16 msSlotNum; /* number Memory Stick® media
slot*/
          UInt16 jogType;
          UInt16 rmcType;
} SonySysFtrSysInfoType;
```

Field Description

`jogType`	Identifies the type of Jog Dial control on the device, as follows:
	`SonySysFtrSysInfoJogTypeNone`
	No Jog Dial® control support
	`sonySysFtrSysInfoJogType1`
	2D type (PEG-S300/S500)
	`sonySysFtrSysInfoJogType2`
	2D type + Back button
	`sonySysFtrSysInfoJogType3`
	2D type + Left and Right buttons(w/o Back button)
	`sonySysFtrSysInfoJogType4`
	2D type + Back button + Left and Right buttons

The following sample code demonstrates how to determine the Jog Dial® control type.

```
#include <SonyCLIE.h>
...
SonySysFtrSysInfoP infoP;
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumSysInfoP,
 (UInt32 *)&infoP)) {
   if (infoP &&
      (infoP->jogType == sonySysFtrSysInfoJogType3)) {
      /* Left/Right Buttons are available */
   } else {
      /* Left/Right Buttons are not available */
   }
} else {
   /* error -  equipment may not be a CLIÉ handheld */
}
```

4 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.

Features

This describes the feature numbers defined for the JogAssist feature.

sonySysFtrNumJogAstControlP

Acquires the address of the flag sonyJogAstControlEnable that sets usage/complete stop of the JogAssist feature. This feature is only enabled on CLIÉ™s installed with PalmOS 5.

The address does not change after a reset.

See "JogAssist feature control" for more information.

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.

JogAssist feature control

NOTE: This control is possible only on CLIEs installed with PalmOS 5.

JogAssist can be completely stopped when wanting to use software other than JogAssist that has similar features, or when a problem that affects operability occurs when JogAssist is used, etc.

Applications that want to completely stop JogAssist will reset the `sonyJogAstControlEnable` flag.

The system defined address that becomes the flag setting destination can be acquired, as shown in the code below, by specifying `sonySysFtrNumJogAstControlP` as the feature number and using the `FtrGet()` API.

```
#include <SonyCLIE.h>
...
UInt32 *flagP;
...
if(!FtrGet(sonySysFtrCreator, sonySysFtrNumJogAstControlP,
(UInt32*)&flagP)) {
  /* JogAssist Terminates */
  *flagP &= ~sonyJogAstControlEnable;
}
```

The setting can be performed with optional timing.

Directly after a system reset the `sonyJogAstControlEnable` flag is set.
This address does not change after a system reset.

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

Priority	Buttons
(High)	Cancel, Previous
	No, Close
	Done
(Low)	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.

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.

vchrJogLeft/vchrJogRight Assist

`vchrJogLeft/vchrJogRight` functions are available on the CLIÉ™ Handheld models equipped with left and right buttons. The left and right buttons replace the traditional up and down scroll buttons.

A) Normal Usage

- Behavior

  Equivalent to using the up and down scroll buttons.

- Details

  `PageUpKey (pageUpChr)` function is executed when the left button is pressed.
  `PageDownKey (pageDownChr)` function is executed when the right button is pressed.

B) Brightness Adjustment Dialog

- Behavior

  Move the slider to the left or the right.

- Details

  Use the left or right buttons to move the slider to the left or right. This behavior is reversed in traditional Palm OS® devices, where the up scroll button (action corresponds to the left button) moves the slider to the right and the down scroll button (action corresponds to the right button) moves the slider to the left.

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™.

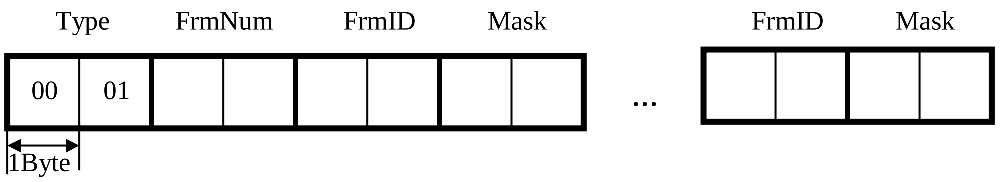
JogAssist Mask Data

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

Below is the format of the currently defined Mask data.

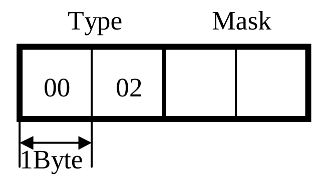
- Type 1

  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

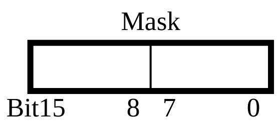
  Specifies effective masks for all forms in the application, including system forms such as alert or help.



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: `vchrJogLeft`
Bit9: `vchrJogRight`
Bit10-15: `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

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.

5

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 A/D value generated when one button is pressed depends on the device, and the value has some variance, but normally the specific reference A/D value within the range of the compressed values shown in the table is obtained. With the standard 6 button audio remote control offered, the relationship becomes as the button—A/D value

correspondence shown in the table below. When 2 buttons are pressed the A/D value such that the higher priority (Play-side) button becomes enabled is obtained.

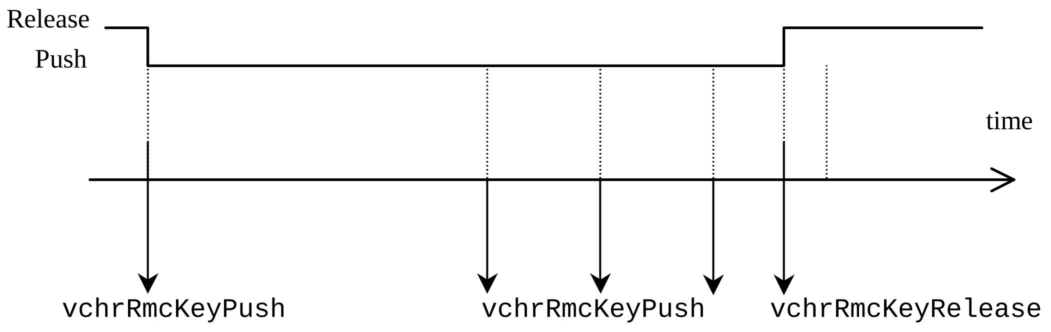
Table 5-1 Button - A/D value table

Button	A/D value		Reference A/D value
	Min	Max	
Play	3235	3372	3303
FR Play	3030	3167	3098
FF Play	2430	2566	2498
Stop	1938	2048	1993
Volume Down	1802	1911	1856
Volume Up	1665	1761	1731

Event intervals

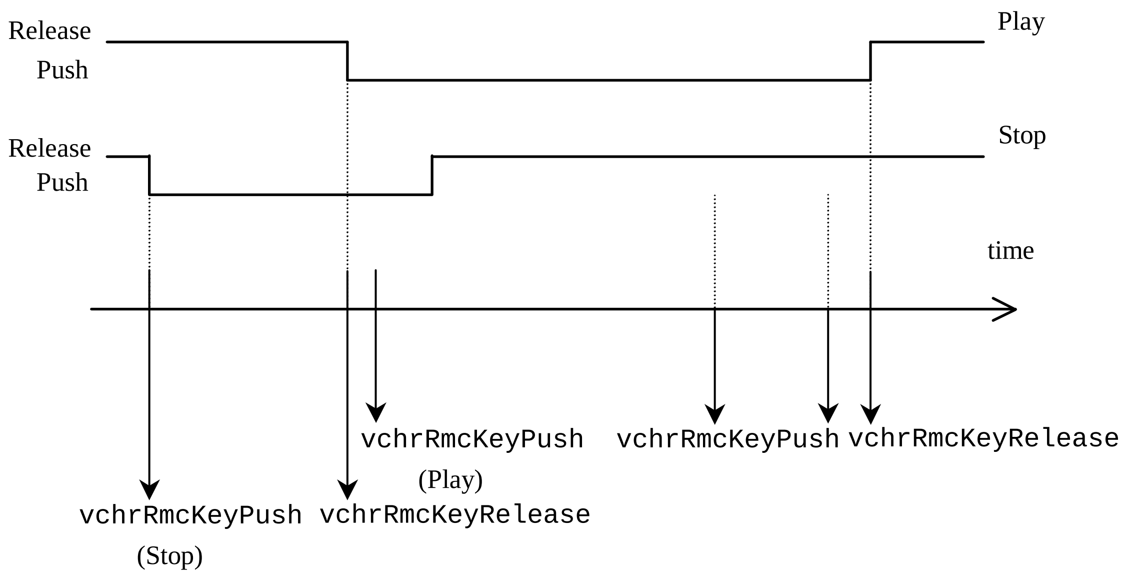
How the events are associated with one another will be explained.

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.

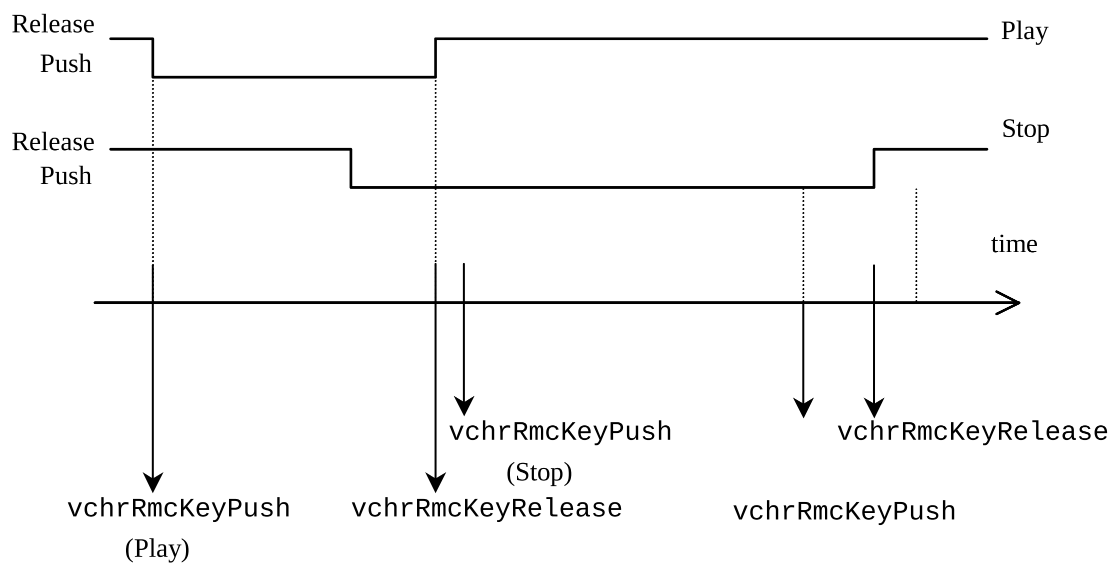
2. Push of two buttons overlapped (When pushing the button with a high priority later.)[1]



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

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

The standard driver software offered by the CLIÉ™ also can support remote controls other than the connected audio remote control. With the initial setting state of the driver software, the reference A/D values for the included audio remote control are statically stored within the event's settings. However, by changing these settings, it is possible to obtain continuous A/D values. (Refer to Audio remote control : Sony Rmc Library).

If continuously obtained A/D values are used as the settings, the values can be stored as is within the even without converting the A/D values from the remote control and its physical interface's such as Play or Stop to static constants. Accordingly, as long as the same hardware requirements are met, it is possible to connect and use other remote controls other than the standard offered audio remote control.

[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.

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.

Part II: Library

6

High Resolution: Sony HR Library

With the CLIÉ™ a 320x320 dot high resolution screen is possible for the first time on a Palm platform. This chapter explains how to effectively use the high resolution screen through the proprietary CLIÉ™ library.

Overview

The high resolution library offers the 320x320 dot high resolution display feature. By use of this library, it is possible for applications to have a rich expression and active presentation that cannot be compared with the past.

However note that with Palm OS 5, a high density API is offered for handling high resolution screens. It is recommended to use this high resolution library on CLIÉ™s installed with Palm OS 4.x and loser, and to use the high density API with Palm OS 5. This high resolution library is offered to support compatibility with Palm OS 5, but there are some parts in which compatibility cannot be guaranteed. For details, refer to the section "Compatibility with PalmOS 5 installed CLIÉ™s"

Screen mode and API

Terminology

Compatibility Mode

Compatibility Mode is the mode that doubles the height and width of the 160x160 VRAM image and displays it on the 320x320 LCD panel. The look of applications executed in compatibility mode is the exact same as on older devices.

NOTE: Compatibility mode is not available on Palm OS devices that support the High-Density feature set.

High resolution mode

High resolution mode is the mode that displays the 320x320 VRAM image as is on the LCD. In high resolution mode there is a method for drawing with the high resolution API and one for drawing with existing APIs. It is possible to use both APIs at the same time.

Existing API

These are APIs offered by the PalmOS. Since applications are automatically spread out into the 320x320 image when drawn by the OS, they do not notice the difference in hardware, and programming can be done the same as always using existing APIs. Also, by simply using the existing APIs, better looking characters will be drawn via the prettier, newly added high resolution fonts.

NOTE: this refers to the API of Palm OS 4 and earlier.

High Resolution (HR) API

This is the Sony API for fully using the 320x320 drawing resolution. Display with beautiful fonts, more characters than can be displayed with current fonts, detailed figure drawing via the 320x320 coordinate system, and highly detailed bitmaps is possible.
The high resolution API does not realize all the drawing functions of the older PalmOS, using the existing API for functions not supported (forms, pen input coordinates, etc.) is designed as a prerequisite. From this, it is recommended that when creating high resolution supported applications, design with the existing API as before, replacing only the parts that you want to draw in high impact with the high resolution API. This keeps source and binary compatibility with older devices, and allows effective use of resources uniquely available in the PalmOS while being a effective means of achieving good looking graphics.

High Density API

An API offered by a new Windows Manager for Palm OS platform devices that have a high resolution screen. A new concept called the Coordinate System is introduced, and a Window Manager performs conversion of this Coordinate System. For details, refer to the Palm OS Programmer's Companion included with the Palm OS SDK 5.0.

Existing Applications

Applications that do not use the native high density API.

Incompatibilities with the existing API

Existing applications will work with almost no changes in high resolution mode, but there are some differences in operation from older devices. The following describes the parts whose operation differes from that of older devices.

1. Regarding Font Drawing

   Drawing characters to the display (screen) window uses double resolution fonts. These are fonts that were newly developed for high resolution usage, and because the glyphs are different than fonts supplied with the PalmOS their has also changed.
   When drawing characters with the existing API to the off screen window older fonts are used.

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. Regarding Line Drawing

   The WinXXXLine functions draw to the the display window (screen). Compatibility is preserved regarding drawing in the horizontal and veritcal directions (the line is drawn with a thickness of 2 pixels).
   Straight lines in the diagonal direction are drawn with width of 1 pixel.
   Drawing lines to the off screen window is as before.

The Object of API

`WinDrawGrayLine`
`WinDrawLine`

The Object of API

```
WinEraseLine
WinFillLine
WinInvertLine
WinPaintLine
```

3. Regarding Pattern Drawing

   The resolution of patterns in the screen window is doubled (Even with GrayPattern the rather than a difference in resolution, it’s a difference in look). Drawing patterns to the off screen window is as before.

NOTE: The behaviorof patterns is different on devices that support the High-Density feature set. Please refer to the SonyHRLib Compatibility Guide at the end of this chapter.

The Object of API

```
WinDrawGrayLine
WinDrawGrayRectangleFrame
WinFillLine
WinFillRectangle
```

4. Regarding Frame Drawing

   When drawing frames into the screen window with the existing API, there are occasions when, depending on the frame type, the line width differs than before. The line width of RoundFrame, boldRoundFrame, dialogFrame is smaller (when frame radius > 2).
   Drawing frames to the off screen window is as before.

The Object of API

```
WinDrawRectangleFrame
WinDrawGrayRectangleFrame
WinEraseRectangleFrame
WinInvertRectangleFrame
WinPaintRectangleFrame
```

5. Regarding Rounded Rectangle Drawing

   Due to increased resolution, when drawing rectangles of radius > 2 with the existing API the roundness of corners is smoother.
   Drawing rectangles to the off screen window is as before.

6. Regarding `WinCopyRectangle`

   With high resolution mode, even when using the existing API the screen window is internally handled as an actually having 320x320 resolution. Due to this, because a change in resolution performed when copying between the screen window and the off screen window with the existing APIs, when a copy such as screen off screen screen is performed, there are occasions when the display cannot be correctly restored. When performing operations such as this, instead of `WinCopyRectangle`, use the `WinSaveBits() and WinRestoreBits()` API.

   Specifically,

   – When copying from the screen window to the off screen window with the existing API, 1/4 of the data will be taken.

   – When copying from the off screen window to the screen windows with existing API, a 4x magnification will result.

   – When copying between off screen windows or screen windows, resolution is not converted.

   – When copying with `HRWinCopyRectangle` in the high resolution API, resolution conversion is not performed.

7. When copying items drawn as in items 1-6 in the off screen window to the screen window, there are occasions where the look will differ from when items are drawn directly to the screen window (characters, lines, patterns, etc.)

8. `WinGetPixel` returns the value of the top-left pixel in the 2x2 high-resolution pixel group that correspond to the 1 pixel in normal resolution. (Compatibility is only preserved with the existing API)

9. Since the amount of display data has a 4x increase, memory usage and time required to transfer data will increase.

10. Forms and objects above them are managed with a 160x160 coordinate system, and when in high resolution mode they are converted to be double height and width and then displayed. However, Resources created with the Constructor offered by CodeWarrior for Palm Release 6 will at their largest be 160x160.

11. When using application fonts, the display is not correctly performed.

12. Applications that draw directly to VRAM are not drawn correctly.

13. With the high resolution library's environment, application defined fonts cannot be used.

Working with Existing API

When using the existing API to draw to the screen window in high resolution mode, the X direction and the Y direction will be doubled and written to VRAM. For example when using `WinDrawPixel` to draw to the position (50, 70), the pixel position in VRAM will be (100,140), (101, 140), (100, 141), (101,141) and set to the current foreground color. In this case using the high resolution API would perform the draw in 320x320 resolution. For example drawing with high resolution API `HRWinDrawPixel`, when drawing (50, 70), the pixel value in VRAM will be the 1 pixel (50,70) and set to the foreground color.

The table below shows the correspondence between the high-resolution and existing APIs. (Limitations for the high resolution API are also shown in the notes). A blank in the high resolution API column indicates that there is no equivalent high resolution function and that the existing API should be used. Note: all existing API functions use the 160x160 coordinate system, even in high –resolution mode. The coordinate system change applies only to the display window.

Table 6-1 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 Selection”.
`WinDrawChars`	`HRWinDrawChars`	See “Font Selection”.
`WinDrawGrayLine`	`HRWinDrawGrayLine`	
`WinDrawGrayRectangleFrame`	`HRWinDrawGrayRectangleFrame`	
`WinDrawInvertedChars`	`HRWinDrawInvertedChars`	See “Font Selection”.
`WinDrawLine`	`HRWinDrawLine`	
`WinDrawPixel`	`HRWinDrawPixel`	
`WinDrawRectangle`	`HRWinDrawRectangle`	
`WinDrawRectangleFrame`	`HRWinDrawRectangleFrame`	
`WinDrawTruncChars`	`HRWinDrawTruncChars`	See “Font Selection”.
`WinEraseChars`	`HRWinEraseChars`	See “Font Selection”.

Table 6-1 High-resolution APIs for Window

Existing API	High-resolution API	Hand instruction for high-resolution API
`WinEraseLine`	`HRWinEraseLine`	
`WinErasePixel`	`HRWinErasePixel`	
`WinEraseRectangle`	`HRWinEraseRectangle`	
`WinEraseRectangleFrame`	`HRWinEraseRectangleFrame`	
`WinEraseWindow`		
`WinFillLine`	`HRWinFillLine`	
`WinFillRectangle`	`HRWinFillRectangle`	
`WinGetActiveWindow`		
`WinGetBitmap`		
`WinGetClip`	`HRWinGetClip`	
`WinGetDisplayExtent`	`HRWinGetDisplayExtent`	
`WinGetDisplayWindow`		
`WinGetDrawWindow`		
`WinGetDrawWindowBounds`		Newly added on PalmOS 4.0
`WinGetFirstWindow`		
`WinGetFramesRectangle`	`HRWinGetFramesRectangle`	
`WinGetPattern`		
`WinGetPatternType`		
`WinGetPixel`	`HRWinGetPixel`	
`WinGetPixelRGB`	`HRWinGetPixelRGB`	Newly added on PalmOS 4.0
`WinGetWindowBounds`	`HRWinGetWindowBounds`	
`WinGetWindowExtent`	`HRWinGetWindowExtent`	
`WinGetWindowFrameRect`	`HRWinGetWindowFrameRect`	
`WinIndexToRGB`		
`WinInvertChars`	`HRWinInvertChars`	See “Font Selection”.

Table 6-1 High-resolution APIs for Window

Existing API	High-resolution API	Hand instruction for high-resolution API
`WinInvertLine`	`HRWinInvertLine`	
`WinInvertPixel`	`HRWinInvertPixel`	
`WinInvertRectangle`	`HRWinInvertRectangle`	
`WinInvertRectangleFrame`	`HRWinInvertRectangleFrame`	
`WinModal`		
`WinPaintBitmap`	`HRWinPaintBitmap`	
`WinPaintChar`	`HRWinPaintChar`	See "Font Selection".
`WinPaintChars`	`HRWinPaintChars`	See "Font Selection".
`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. In PalmOS 5 device, mode change is invalid.

Table 6-1 High-resolution APIs for Window

Existing API	High-resolution API	Hand instruction for high-resolution API
`WinScreenUnlock`		
`WinScrollRectangle`	`HRWinScrollRectangle`	
`WinSetActiveWindow`		
`WinSetBackColor`		
`WinSetBackColorRGB`		Newly added on PalmOS 4.0
`WinSetClip`	`HRWinSetClip`	Clipping rectangle setting is limited.
`WinSetDrawMode`		
`WinSetDrawWindow`		
`WinSetForeColor`		
`WinSetForeColorRGB`		Newly added on PalmOS 4.0
`WinSetPattern`		
`WinSetPatternType`		
`WinSetTextColor`		
`WinSetTextColorRGB`		Newly added on PalmOS 4.0
`WinSetUnderlineMode`		
`WinSetWindowBounds`	`HRWinSetWindowBounds`	Bounding rectangles setting is limited.
`WinValidateHandle`		
`WinWindowToDisplayPt`	`HRWinWindowToDisplayPt`	

Table 6-2 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.

Table 6-2 High-resolution API for Bitmap

Existing API	High-resolution API	Handling instruction for high-resolution API
`BmpCreate`	`HRBmpCreate`	
`BmpDelete`		
`BmpGetBits`		
`BmpGetBitDepth`		Newly added on PalmOS 4.0
`BmpGetColortable`		
`BmpGetDimensions`		Newly added on PalmOS 4.0
`BmpGetNextBitmap`		Newly added on PalmOS 4.0
`BmpGetSizes`		Newly added on PalmOS 4.0
`BmpSize`	`HRBmpSize`	

Table 6-3 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`	

Table 6-3 High-resolution API for Font

Existing API	High-resolution API	Handling instruction for high-resolution API
`FntWCharWidth`		Newly added on PalmOS 4.0
`FntWidthToOffset`		
`FntWordWrap`		
`FntWordWrapReverseNLines`		
`FontSelect`	`HRFontSelect`	

However, with CLIÉ handhelds using PalmOS 5, to preserve compatibility with older devices, it is necessary to replace the following Font APIs with the newly added high resolution APIs. For details refer to Compatibility with PalmOS 5 installed CLIÉ™s.

Table 6-4 List of high resolution Font APIs added in PalmOS 5

Existing API	High resolution API
`FntBaseLine`	`HRFntBaseLine`
`FntCharHeight`	`HRFntCharHeight`
`FntLineHeight`	`HRFntLineHeight`
`FntAverageCharWidth`	`HRFntAverageCharWidth`
`FntCharWidth`	`HRFntCharWidth`
`FntWCharWidth`	`HRFntWCharWidth`
`FntCharsWidth`	`HRFntCharsWidth`
`FntWidthToOffset`	`HRFntWidthToOffset`
`FntCharsInWidth`	`HRFntCharsInWidth`
`FntDescenderHeight`	`HRFntDescenderHeight`
`FntLineWidth`	`HRFntLineWidth`
`FntWordWrap`	`HRFntWordWrap`
`FntWordWrapReverseNLines`	`HRFntWordWrapReverseNLines`
`FntGetScrollValues`	`HRFntGetScrollValues`

Below shows the corresponding behavior of the existing/high resolution API on each device.

Compatality of high-resolution and existing APIs with these model

Device	Mode	High resolution API	Existing API
Conventional model	--	NG (Fatal Error)	OK
High-resolution support model	Compatibility Mode	`HRWinScreenMode` : OK The other APIs : NG (Fatal Error)	OK
	High Resolution Mode	OK	OK(Enables distinct character display.)
High Density support model	--	OK	OK(Enables distinct character display.)

Font Selection

Character output with the existing API uses the fonts shown below as before. When the fonts below are specified, they are internally replaced with a double resolution font and better looking characters are drawn.

Table 6-5 FontID

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

When using the high resolution API, in addition to the above fonts 8 more fonts can be used.

To specify the 16 types of fonts in high resolution mode, instead of the existing FontID type the HRFontID type is defined.

Table 6-6 HRFontID

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

With the high resolution API, the fonts are displayed with their original size, so for example when specifying `hrTinyFont`(= `stdFont`) and displaying kanji, on the 320x320 sized display, the original stdFont is displayed with an 8x8 pixel size (1/4 the original size). When wishing to display fonts the same size as older devices using the high resolution API, it is necessary to specify the fontset `HRFontID` 8-15.
When setting fonts and getting the current specified font in high resolution mode, use

```
HRFont  HRFntGetFont( UInt16 refNum )
HRFont  HRFntSetFont( UInt16 refNum, HRFontID font )
```

When drawing using the existing API with font `HRFontID` 8-15 specified, `hrStdFont(HRFontID = 8)` is used for the acutal drawing.

With the Palm OS, the draw command and the draw attributes are independent. For example, with `hrLargeBoldFont(HRFontID = 15)` set in high resolution mode and after drawing characters with the high resolution API (`HRWinDrawChars`, etc), and then drawing characters with the existing API (`WinDrawChars`, etc), the font is FontID

= 15, take care since `hrStdFont(HRFontID = 8)`will be used in replacement. Hence, when drawing characters with the high resolution API, before drawing set `HRFntSetFont.` After that, if drawing characters with the existing API, before drawing specify the font with `FntSetFont.`

As for APIs that get the font size properties such as width and height, etc., the existing API can also be used in high resolution mode. When high resolution APIs are used for drawing, since high resolution fonts are specified, the font size can be acquired on the 320×320 coordinate system. When drawing in high resolution mode with existing APIs, since exisiting fonts are specified the size is returned on the 160×160 coordinate system.

Screen Window and Off Screen Window

The screen window, when considered under existing APIs, is a window that holds a 160×160 sized bitmap, but in reality it holds a 320×320 sized bitmap. In comparison, the off screen window is a window that holds a bitmap of a specified size.

For example, when using existing APIs

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

the created off screen window becomes a window that holds a 160×160 sized bitmap.

Also, when using high resolution APIs

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

the created off screen window becomes a window that holds a 320×320 sized bitmap.

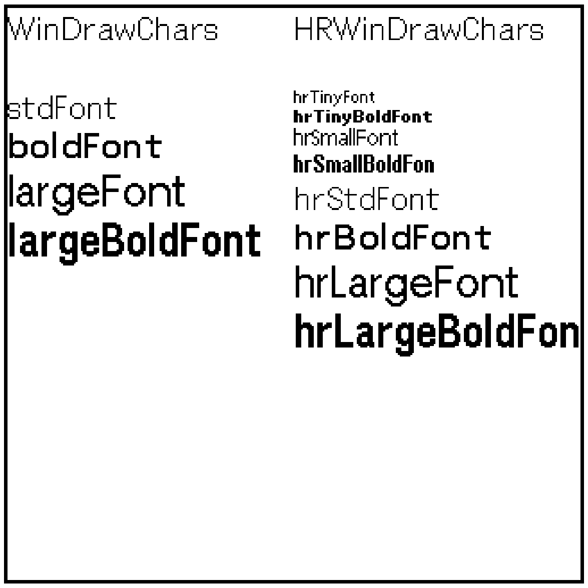
When drawing using the existing API, there are occasionally diffrences in drawing to the screen window and to the off screen window. When drawing using the high resolution API, there is no difference between the screen window and the off screen window.

The following explains examples of character and line drawing.

Character drawing

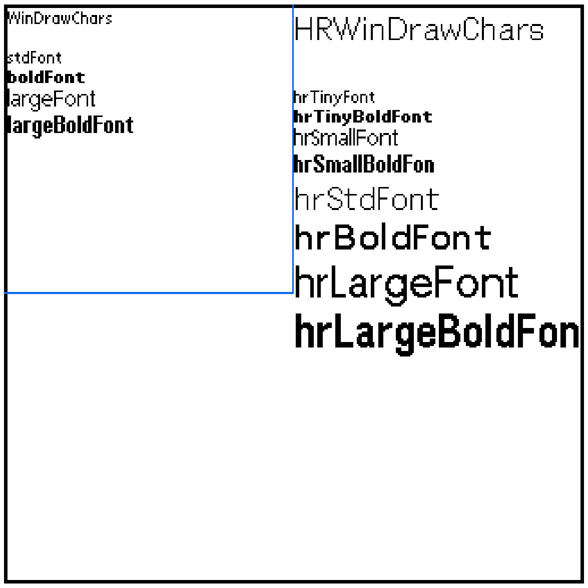
Drawing to the screen window is as shown in Figure 1. The left side shows drawing using the `WinDrawChars` existing API; the right side shows drawing using the high resolution API `HRWinDrawChars`.

Figure 1



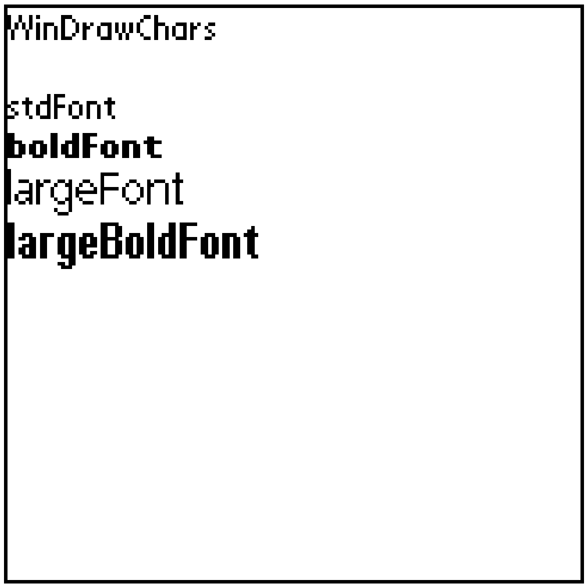
Drawing to the off screen window is as shown in Figure 2. (Internally the rectangle has a 160×160 range) When this window is copied to the screen window with `HRCopyRectangle`, it appears on the screen as shown below.

Figure 2



When copying the 160x160 range of Figure 2 to the screen window using `WinCopyRectangle`, it is displayed on the screen as seen below.

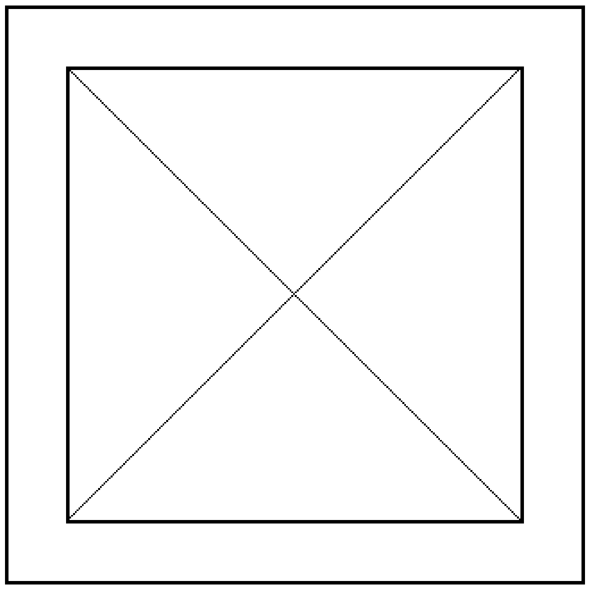
Figure 3



Line drawing

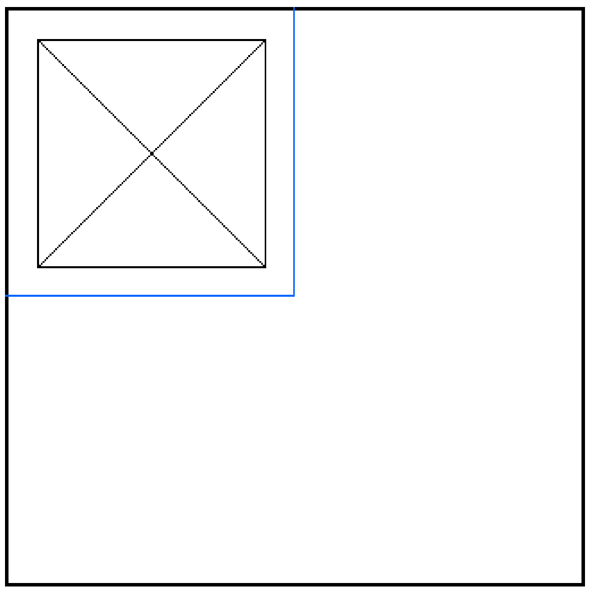
When drawing 6 lines to the screen window under the existing API `WinDrawLine`, the following occurs.

Figure 4



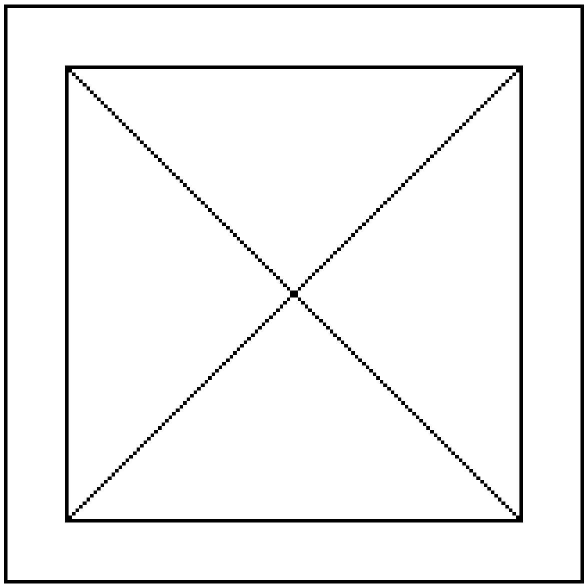
And when drawing the same thing to the offscreen window, the following occurs.

Figure 5



When a 160x160 range of this offscreen window is copied to the screen window with `WinCopyRectangle`,the following occurs. In this case the line width is different than when drawing direct to the screen window.

Figure 6



Using High Resolution

Loading the Library

The high resolution API is offered through a library. To use the library, acquire the library reference number through `SysLibFind`.

The following shows an implementation example.

```
#include <SonyCLIE.h>

Err error = 0;
UInt16 refNum, version;
UInt32 width,height,depth;
Boolean color;

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 */
  HROpen( refNum );
  HRGetAPIVersion( refNum, &version );

  /* Change resolution if HRLib is older one */
  if ( version < HR_VERSION_SUPPORT_FNTSIZE ) {
    /* Refer to “Changing Screen Modes” */
  }
  ...
}
```

All of the high resolution APIs are accessed by the reference number acquired by `SysLibFind` (or `SysLibLoad`). The reference number cannot be acquired on devices that do not support high resolution. When the reference number can not be acquired, the high resolution API cannot be used. In these cases, the drawing mode will only occur in compatibility mode.

To use the high resolution API, it is necessary to first call the open function `HROpen.` Also, when closing call the close function `HRClose`.

Changing Screen Modes

NOTE: Since there is no compatible mode on Palm OS 5 installed CLIÉ™s, the screen mode cannot be changed.

When programming only with the existing APIs, the program will run in compatibility mode. To use high resolution mode, it is necessary for applications to independently change the mode.
The 2 modes, compatible and high resolution modes, can be switched with the `HRWinScreenMode()` API. The following shows and compares switching modes with `WinScreenMode()` API and `HRWinScreenMode()` API.

Table 6-7 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 6-8 operation : `winScreenModeSetToDefaults`

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

For applications that use high resolution mode, have the application switch to high resolution mode at startup and return to the default resolution mode at shutdown.
When starting another application (`SysAppLaunch`) while currently in high resolution mode, start the application via `SysAppLaunch` after first returning to default resolution mode. Then set the mode back to high resolution mode after the returning from the started application.
However, when changing the mode the screen is cleared.

The following shows an implementation example

Example 1 Changing to high resolution mode

```
#include <SonyCLIE.h>


Err    error;
UInt16 refNum;
UInt32 width, height;

/*********************************************/
/*  Get refNum of SonyHRLib      */
/*********************************************/
```

```
/*******************************************/
/*  Open the library       */
/*******************************************/
error = HROpen(refNum);
if (error) {
  /* Error Processing */
} else {
  width = hrWidth;
  height = hrHeight;
  error = HRWinScreenMode ( refNum, winScreenModeSet,
  &width, &height, NULL, NULL );
  if ( error != errNone ){
    /* Still in the previous screen mode */
    - - - - - - - - - - -
  } else {
    /* High resolution mode */
    - - - - - - - - - - -
  }
}
```

Example 2 Changing back to default resolution mode from high resolution mode and closing

```
error = HRWinScreenMode ( refNum,
winScreenModeSetToDefaults, NULL, NULL, NULL, NULL );
if ( error != errNone ){
  /* Still in high resolution mode */
  - - - - - - - - - - -
} else {
  /* Chaning back to default screen mode successful */
  - - - - - - - - - - -
}
/*******************************************/
/*  Close the library      */
/*******************************************/
error = HRClose(refNum);
```

High resolution API

For details on the following APIs and when corresponding existing API exist, refer to the PalmOS documentation.

System I/F Functions

HROpen

Purpose Begin using the high resolution library

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`	Insufficient memory

Comments Performs processing to begin using the high resolution library.

HRClose

Purpose End usage of the high resolution library

Prototype `Err HRClose ( UInt16 refNum )`

Parameters

`-> refNum`	Reference number of high resolution library

Result

`errNone`	No error
`hrErrNotOpen`	The high resolution library is not open
`hrStillOpen`	The high resolution library is still open

Comments Performs processing for closing the high resolution library.

HRGetAPIVersion

Purpose Get the high resolution API version

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

Parameters

`-> refNum`	Reference number of high resolution library

`<-  versionP`	Pointer to the memory area holding the API version

Result

`errNone`	No error
`hrErrNotOpen`	High resolution library is not Open
`hrErrParam`	Parameter error (versionP is NULL)

Comments Acquires the high resolution API version.

version

15							8	7							0
Major Version								Minor Version							

Window APIs

HRWinClipRectangle

Purpose Clips and fits the specified retangular area into the current draw window’s clipping area

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`<-> rP`	Pointer to the structure of the rectangular area to be clipped Returns the intersection of the argument rectangle with the clipping bounds of the draw window

Result None

HRWinCopyRectangle

Purpose Copy a rectangular area

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`	Copy source’s rectangular area window When NULL, becomes the draw window
`-> dstWin`	Copy destination’s Rectangular area window When NULL, becomes the draw window

`-> srcRect`	Copy range
`-> destX`	Top of copy destination window's rectangular area
`-> destY`	Leftmost side of copy destination window's rectangular area
`-> mode`	Method for moving from source to destination

Result None

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 bitmap to relate to this window
`<-  error`	Pointer to errors sourced by this function

Result When successful, returns the handle for the new window. When there is an error, returns NULL.

Error parameter holds one of the following.

`errNone`	No error
`sysErrParamErr`	`bitmapP` parameter is invalid. The bitmap is uncompressed, and valid pixel sizes must be (1,2,4,8,16). Cannot be screen bitmaps.
`SysErrNoFreeResource`	Reserved the new window structure but memory is insufficient.

HRWinCreateOffscreenWindow

Purpose Create new off screen window, and add it to window list

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> width`	Window width
`-> height`	Window heigth
`-> format`	screenFormat or genericFormat Generally, the off screen window is genericFormat.

`<- error`	Pointer to errors sourced by this function

Result When successful, returns the handle for the new window. When there is an error, returns NULL.

Error parameter holds one of the following.

`errNone`	No error
`sysErrParamErr`	width or height paramter is NULL. Or, the current color pallete is invalid.
`sysErrNoFreeResource`	This function executed but memory was insufficient.
`memErrNotEnoughSpace`	This function executed but memory was insufficient.

HRWinCreateWindow

Purpose Create new window, and register it to 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`	Window display range the elements of `bounds`, `(topleft.x, topleft.y, extent.x, extent.y)`, must be multiples of 2
`-> frame`	Windows' frame type
`-> modal`	If window is modal, then true
`-> focusable`	If window can become active window, then true
`<- error`	Pointer to errors sourced by this function

Result When successful, returns handle for window; when error occurs, returns NULL.

HRWinDisplayToWindowPt

Purpose Convert display coordinates to window coordinates. Coordinates in the display window.are returned

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 None

HRWinDrawBitmap

Purpose Draws a bitmap with `winPaint` mode into the specified location

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> bitmap`	Pointer to bitmap
`-> x`	x coordinate of upper left corner
`-> y`	y coordinate of upper left corner

Result None

HRWinDrawChar

Purpose Draw specified characters into the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> theChar`	Character to draw
`-> x`	x coordinate of drawing position (leftmost side)
`-> y`	y coordinate of drawing position (top)

Result None

HRWinDrawChars

Purpose Draws a specified characters string into 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 character string to draw

`-> len`	Length of character string to draw (bytes)
`-> x`	x coordinate of first character in string to draw (leftmost side)
`-> y`	y coordinate of first character in string to draw (top)

Result None

HRWinDrawGrayLine

Purpose Draw a dotted line into 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 starting point
`-> y1`	y coordinate of starting point
`-> x2`	x coordinate of endpoint
`-> y2`	y coordinate of endpoint

Result None

HRWinDrawGrayRectangleFrame

Purpose Draw a gray rectangular frame into the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type to draw
`-> rP`	Pointer to frame’s rectangular area

Result None

HRWinDrawInvertedChars

Purpose Draw the specified character string inverted (in the background color) into the drawing 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 character string to draw
`-> x`	x coordinate of first character in string to draw (leftmost side)
`-> y`	y coordinate of first character in string to draw (top)

Result None

HRWinDrawLine

Purpose Draw a line using the current forground color into the draw window

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 starting point
`-> y1`	y coordinate of starting point
`-> x2`	x coordinate of end point
`-> y2`	y coordinate of end point

Result None

HRWinDrawPixel

Purpose Draw a pixel using the current foreground color into the draw window

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 None

HRWinDrawRectangle

Purpose Draw a rectangle using the current foreground color into the draw window

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to rectangle to draw
`-> cornerDiam`	Corner radius When 0, uses square corners

Result None

HRWinDrawRectangleFrame

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

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type to draw
`-> rP`	Pointer to frame's rectuangular area

Result None

HRWinDrawTruncChars

Purpose Truncate a specified character string truncated into a specified width and draw into the draw window

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 character string to draw
`-> len`	Length of character string
`-> x`	x coordinate of first character in string to draw (leftmost sid)
`-> y`	y coordinate of first character in string to draw (top)

`-> maxWidth`	Maximum value for width of character string to draw

Result None

HRWinEraseChars

Purpose Erase specified character string from 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 character string to erase
`-> len`	Length of character string to erase
`-> x`	x coordinate of first character in string to erase (leftmost side)
`-> y`	y coordinate of first character in string to erase (top)

Result None

HRWinEraseLine

Purpose Draw a line using the current background color into the draw window

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 starting point
`-> y1`	y coordinate of starting point
`-> x2`	x coordinate of end point
`-> y2`	y coordinate of end point

Result None

HRWinErasePixel

Purpose Draw a pixel using the current background color into the draw window

Prototype

```
void HRWinErasePixel ( 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 None

HRWinEraseRectangle

Purpose Draw rectangle using current background color into the draw window

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to rectangle to erase
`-> cornerDiam`	Corner radius When 0, uses square corners

Result None

HRWinEraseRectangleFrame

Purpose Draw rectangular frame using current background color into the draw window

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type to erase
`-> rP`	Pointer to frame’s rectangular area

Result None

HRWinFillLine

Purpose Fill line in draw window with 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 starting point
`-> y1`	y coordinate of starting point

`-> x2`	x coordinate of end point
`-> y2`	y coordinate of end point

Result None

HRWinFillRectangle

Purpose Draw a rectangle 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 rectangle to draw
`-> cornerDiam`	Corner radius When 0 uses square corners

Result None

HRWinGetClip

Purpose Return the clipping area of the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`<-  rP`	Pointer to structure holding the clipping area

Result None

HRWinGetDisplayExtent

Purpose Return the display (screen) height and width

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

Parameters

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

Result None

HRWinGetFramesRectangle

Purpose Return a rectangular area including its surrounding frame

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type
`-> rP`	Pointer to frame’s rectangular area
`<- obscuredRectP`	Pointer to specified rectangle including its frame

Result None

HRWinGetPixel

Purpose Return color value for pixel 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 pixel
`-> y`	y coordinate of pixel

Result Index color value of the pixel

HRWinGetPixelRGB

Purpose Return RGB color value for pixel in draw window

Prototype

```
Err HRWinGetPixelRGB(UInt16 refNum, Coord x, Coord y,
RGBColorType *rgbP)
```

Parameters

`-> refNum`	Reference number of high resolution library
`-> x`	x coordinate of pixel
`-> y`	y coordinate of pixel
`<- rgbP`	RGB color component of pixel

Result `errNone`

`sysErrParamErr`	`x, y is less than 0 or out of draw window limits`

HRWinGetWindowBounds

Purpose Get current draw window boundary in the display coorinate system

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`<-  rP`	Pointer to rectangular area

Result None

HRWinGetWindowExtent

Purpose Return height and width of draw window

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`<-  extentX`	Width of draw window
`<-  extentY`	Height of draw window

Result None

HRWinGetWindowFrameRect

Purpose Return rectangle defining a window and its frame size and position in the display coordanate system

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> winHandle`	Window handle
`<-  rP`	Pointer to window coordinates

Result None

HRWinInvertChars

Purpose Invert the specified character string in the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> chars`	Pointer to character string to invert
`-> len`	Length of character string to invert (bytes)
`-> x`	x coordinate of first character in string to invert (leftmost side)
`-> y`	y coordinate of first character in string to invert (top)

Result None

HRWinInvertLine

Purpose Invert line in the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> x1`	x coordinate starting point
`-> y1`	y coordinate starting point
`-> x2`	x coordinate end point
`-> y2`	y coordinate end point

Result None

HRWinInvertPixel

Purpose Invert pixel in draw window

Prototype
```
void HRWinInvertPixel ( 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 None

HRWinInvertRectangle

Purpose Invert rectangle in draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to rectangle to invert
`-> cornerDiam`	Corner radius When 0, uses square corners

Result None

HRWinInvertRectangleFrame

Purpose Invert rectangle frame in draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type to draw
`-> rP`	Pointer to frame's rectangular area

Result None

HRWinPaintBitmap

Purpose Draw a bitmap into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> bitmapP`	Pointer to bitmap
`-> x`	x coordinate of upper left corner
`-> y`	y coordinate of upper left corner

Result None

HRWinPaintChar

Purpose Draw a character into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> theChar`	Character to draw
`-> x`	x coordinate of character to draw (leftmost side)
`-> y`	y coordinate of character to draw (top)

Result None

HRWinPaintChars

Purpose Draw character string into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> chars`	Pointer to character string to draw
`-> len`	Length of character string to draw (bytes)
`-> x`	x coordinate of first character of string to draw (leftmost side)
`-> y`	y coordinate of first character of string to draw (top)

Result None

Comments

HRWinPaintLine

Purpose Draw line into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> x1`	x coordinate of starting point
`-> y1`	y coordinate of starting point
`-> x2`	x coordinate of end point

	`-> y2`	y coordinate of end point

Result None

HRWinPaintLines

Purpose Draw several lines into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> numLines`	Number of lines to draw
`-> lines`	Line arrangement

Result None

HRWinPaintPixel

Purpose Draw pixel into the draw window using the current draw state

Prototype
```
void HRWinPaintPixel ( 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 None

HRWinPaintPixels

Purpose Draw several pixels into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> numPoints`	Number of pixels to draw
`-> pts`	Pixel arrangement

Result None

HRWinPaintRectangle

Purpose Draw rectangle into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to rectangular area
`-> cornerDiam`	Corner radius When 0, uses square corners

Result None

HRWinPaintRectangleFrame

Purpose Draw rectangular frame into the draw window using the current draw state

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> frame`	Frame type
`-> rP`	Pointer to frame’s rectangular area

Result None

HRWinRestoreBits

Purpose Copy contents of specified window into the draw window, and delete the original window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> winHandle`	Window handle
`-> destX`	Draw window x coorindate for the copy destination
`-> destY`	Draw window y coordinate for the copy destination

Result None

HRWinSaveBits

Purpose Create off screen window and copy specified area of the draw window into it

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> sourceP`	Pointer to rectangular area to save in display coordinates
`<-  error`	Pointer to errors sourced by this function

Result The window handle including the saved image

0 when an error occurs

HRWinScreenMode

Purpose Set/Get display parameters (display width, height, bit depth, color support)

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> operation`	The operation of this function is decided by the following selectors

`winScreenModeGet`
Returns current display settings

`winScreenModeGetDefaults`
Returns display default settings

`winScreenModeGetSupportedDepths`
Stores and returns the supported screen bit depth in depthP
For details refer to `WinScreenMode` in the SDK

`winScreenModeGetSupportsColor`
When color mode is possible true is stored and returned in enableColorP

`winScreenModeSet`
Set display settings to value specified by other arguments

`winScreenModeSetToDefaults`
Set display settings to default values

`<-> widthP`	Pointer to new/old screen width
`<-> heightP`	Pointer to new/old screen height
`<-> depthP`	Pointer to new/old/possible screen depth
`<-> enableColorP`	If color drawing mode is possible then true

Result If there is no error, the returned value is decided by the operation parameter

When the parameter is invalid, `sysErrParamErr` is returned

When there is a failure in allocating memory, `memErrNotEnoughSpace` is returned

Comments The following shows a comparison between `WinScreenMode()` and `HRWinScreenMode()`.

NOTE: Since there is no compatible mode on Palm OS 5 installed CLIÉ™s, the screen mode cannot be changed.

The values for the parameters (width, height) that can be acquired in each screen mode are as shown below.

Table 6-9 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 6-10 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

The screen mode that changes with respect to current mode and the called API is shown below.

Table 6-11 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 6-12 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 rectangle in the draw window

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to rectangular area to scroll
`-> direction`	Scroll direction (`winUp, winDown, winLeft, winRight`)
`-> distance`	Scroll distance (pixels)
`<- vacatedP`	Pointer to rectangular area that must be redrawn due to its being vacated by the scrolling

Result None

HRWinSetClip

Purpose Set the clipping rectangle of the draw window

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> rP`	Pointer to structure holding the clipping range elements of `rP`, `(topleft.x, topleft.y, extent.x, extent.y)`, must be multiples of 2

Result None

HRWinSetWindowBounds

Purpose Set bounds of window in display coordinates

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> winHandle`	Handle of window to set bounds for
`-> rP`	Pointer to rectangle to use as boundary elements of `rP`, `(topleft.x, topleft.y, extent.x, extent.y)`, must be multiples of 2

Result None

HRWinWindowToDisplayPt

Purpose Convert window coordinate system to display coordinate system

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`<-> extentX`	x coordinate to convert
`<-> extentY`	y coordinate to convert

Result None

Bitmap APIs

HRBmpBitsSize

Purpose Return size of bitmap data

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> bitmapP`	Pointer to bitmap

Result Retunrs size of bitmap data in bytes
Does not include header and color table

HRBmpSize

Purpose Return size of bitmap data

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> bitmapP`	Pointer to bitmap

Result Returns size of bitmap data in bytes
Includes header and color table

HRBmpCreate

Purpose Create a bitmap

Prototype

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> width`	Width of bitmap (pixels) nonzero
`-> height`	Height of bitmap (pixels) nonzero
`-> depth`	Pixel depth of bitmap 1, 2, 4, 8 or 16 This value is used as the the `pixelSize` field of `BitmapType`

`-> colortableP`	Pointer to color table related to bitmap NULL when bitmap does not have a color table The number of colors in the color table must correspond to `depth` parameter value (1-bit = 2, 2-bit = 4, 4-bit = 16, 8-bit = 256)
`<-  error`	Pointer to errors sourced by this function

Result Returns pointer to structure of new bitmap. When there is an error returns NULL.

The Error parameter contains the following values.

`errNone`	success
`sysErrParamErr`	`width, height, depth, colorTableP` is an invalid value
`memErrNotEnoughSpace`	Insuffient memmory to allocate for structure

Font APIs

HRFntGetFontSize

Purpose Return current font ID

Prototype `HRFontID HRFntGetFont ( UInt16 refNum )`

Parameters

`-> refNum`	Reference number of high resolution library

Result Returns current font's font ID

HRFntSetFont

Purpose Set the font

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> font`	Font ID to set

Result Returns font ID before the change

HRFontSelect

Purpose Display a dialog box for user to select a font, and return the selected font ID

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

Parameters

`-> refNum`	Reference number of high resolution library
`-> font`	Highlighted font ID in dialog box Sets one of the following US: `hrStdFont` `hrBoldFont` `hrLargeBoldFont` J: `hrStdFont` `hrBoldFont` `hrLargeFont` `hrLargeBoldFont`

Result Returns the selected font ID

Compatibility with PalmOS 5 installed CLIÉ™s

When operating older high resolution applications on CLIÉs using PalmOS 5, there are some areas where compatibility is not preserved.

This section describes the incompatible areas of HRLib with PalmOS 4.x and earlier, and PalmOS 5.x and later.

Areas incompatible with the past

Incompatibilites due to the change in API

Binary compatibility cannot be guaranteed for applications that fall under this change, but by rewriting them using the new API it is possible to ensure compatibility with older devices and PalmOS 5.

- APIs that handle font size

Incompatibilites with drawing

Due to change in implementation, there are occasions when the display differs than that of older devices

- Drawing fonts similar to SmallFont/TinyFont
- Drawing with patterns
- Drawing under the changed implementation of PalmOS 5

APIs that handle font size

When using the following APIs to calculate high resolution coordinates for existing HighRes applications, if PalmOS 5 is executed in this state there are occations where the characters are not displayed as before, but rather be shifted or corrupted. This is because the following APIs return half the size from before, and they cannot be assimilated into the framework of the former HighRes API.

When using the highres API on PalmOS 5 and later, it is necessary to replace the following APIs with their corresponding newly added APIs to deal with this problem.

Existing API	Corresponding new API
`FntBaseLine`	`HRFntBaseLine`
`FntCharHeight`	`HRFntCharHeight`
`FntLineHeight`	`HRFntLineHeight`
`FntAverageCharWidth`	`HRFntAverageCharWidth`
`FntCharWidth`	`HRFntCharWidth`
`FntWCharWidth`	`HRFntWCharWidth`
`FntCharsWidth`	`HRFntCharsWidth`
`FntWidthToOffset`	`HRFntWidthToOffset`
`FntCharsInWidth`	`HRFntCharsInWidth`
`FntDescenderHeight`	`HRFntDescenderHeight`
`FntLineWidth`	`HRFntLineWidth`
`FntWordWrap`	`HRFntWordWrap`
`FntWordWrapReverseNLines`	`HRFntWordWrapReverseNLines`
`FntGetScrollValues`	`HRFntGetScrollValues`

However, it is not necessary to replace for situations that use the coordinate caluclations of the older coordinates (160x160 system).

Newly added APIs

If the API version returned by HRGetAPIVersion is greater or equal to

```
HR_VERSION_SUPPORT_FNTSIZE
```

the new APIs are available.

Details on the following APIs are noted in the Palm OS Programmer's API Reference, refer to the items for the corresponding exsiting APIs.

HRFntBaseLine

Prototype `Int16 HRFntBaseLine (UInt16 ref)`

Parameters	`-> ref`	Reference number of high resolution library

Comments Refer to `FntBaseLine` in the Palm OS Programmer's API Reference

HRFntCharHeight

Prototype `Int16 HRFntCharHeight (UInt16 ref)`

Parameters	`-> ref`	Reference number of high resolution library

Comments Refer to `FntCharHeight` in the Palm OS Programmer's API Reference

HRFntLineHeight

Prototype `Int16 HRFntLineHeight (UInt16 ref)`

Parameters	`-> ref`	Reference number of high resolution library

Comments Refer to `FntLineHeight` in the Palm OS Programmer's API Reference

HRFntAverageCharWidth

Prototype `Int16 HRFntAverageCharWidth (UInt16 ref)`

Parameters	`-> ref`	Reference number of high resolution library

Comments Refer to `FntAverageCharWidth` in the Palm OS Programmer's API Reference

HRFntCharWidth

Prototype `Int16 HRFntCharWidth (UInt16 ref, Char ch)`

Parameters	`-> ref`	Reference number of high resolution library

Comments Refer to `FntCharWidth` in the Palm OS Programmer's API Reference

HRFntWCharWidth

Prototype `Int16 HRFntWCharWidth (UInt16 ref, WChar iChar)`

Parameters

`-> ref`	Reference number of high resolution library

Comments Refer to `FntWCharWidth` in the Palm OS Programmer’s API Reference

HRFntCharsWidth

Prototype

```
Int16 HRFntCharsWidth (UInt16 ref, Char const *chars,
Int16 len)
```

Parameters

`-> ref`	Reference number of high resolution library

Comments Refer to `FntCharsWidth` in the Palm OS Programmer’s API Reference

HRFntWidthToOffset

Prototype

```
Int16 HRFntWidthToOffset (Char const *pChars, UInt16length,
Int16 pixelWidth, Boolean *leadingEdge, Int16 *truncWidth)
```

Parameters

`-> ref`	Reference number of high resolution library

Comments Refer to `FntWidthToOffset` in the Palm OS Programmer’s API Reference

HRFntCharsInWidth

Prototype

```
void HRFntCharsInWidth (UInt16 ref, Charconst*string,
Int16 *stringWidthP, Int16*stringLengthP,
Boolean *fitWithinWidth)
```

Parameters

`-> ref`	Reference number of high resolution library

Comments Refer to `FntCharsInWidth` in the Palm OS Programmer’s API Reference

HRFntDescenderHeight

Prototype `Int16 HRFntDescenderHeight (UInt16 ref)`

Parameters

`-> ref`	Reference number of high resolution library

Comments Refer to `FntDescenderHeight` in the Palm OS Programmer’s API Reference

HRFntLineWidth

Prototype `Int16 HRFntLineWidth (UInt16 ref, Char const *pChars, UInt16 length)`

Parameters `-> ref` Reference number of high resolution library

Comments Refer to `FntLineWidth` in the Palm OS Programmer's API Reference

HRFntWordWrap

Prototype `UInt16 HRFntWordWrap (UInt16 ref, Char const *chars, UInt16 maxWidth)`

Parameters `-> ref` Reference number of high resolution library

Comments Refer to `FntWordWrap` in the Palm OS Programmer's API Reference

HRFntWordWrapReverseNLines

Prototype

```
void HRFntWordWrapReverseNLines (UInt16 ref,
Char const *const chars, UInt16 maxWidth,
UInt16 *linesToScrollP, UInt16 *scrollPosP)
```

Parameters `-> ref` Reference number of high resolution library

Comments Refer to `FntWordWrapReverseNLines` in the Palm OS Programmer's API Reference

HRFntGetScrollValues

Prototype

```
void HRFntGetScrollValues (UInt16 ref, Char const *chars,
UInt16 width, UInt16 scrollPos, UInt16 *linesP,
UInt16 *topLine)
```

Parameters `-> ref` Reference number of high resolution library

Comments Refer to `FntGetScrollValues` in the Palm OS Programmer's API Reference

New API usage example

```
#include <SonyHRLib.h> // HiReso API Ver 2.0 supported version

/* Global variables */
UInt16 hrRef= sysInvalidRefNum;
```

```
UInt16 gHrV2= true;


/* Sample of initialization processing of HiRes Lib */
{
  if(SysLibFind(sonySysLibNameHR, &hrRef)){
    SysLibLoad('libr', sonySysFileCHRLib, &hrRef);
  }


  if (hrRef != sysInvalidRefNum) {
    UInt32 width= 320, depth= 8;
    Err err;
    UInt16 ver;

    HROpen(hrRef);
    err= HRWinScreenMode( hrRef, winScreenModeSet,
      &width, &width, &depth, NULL);

    HRGetAPIVersion( hrRef, &ver);
    if (ver >= HR_VERSION_SUPPORT_FNTSIZE)
      gHrV2= true;
  }
}


/* Sample of drawing code */
{
  Int16 x= 20, y= 20;
  Int16 w, h;
  RectangleType rect;
  Char *strs= "HR Sample (20,20)";

  HRFntSetFont( hrRef, hrSmallFont);

  /* Before modification
  w = FntCharWidth(strs, StrLen(strs));
  h = FntLineHeight();
  */


  if (gHrV2) { // Necessary to distinguish the old version HRAPI
    w = HRFntCharsWidth(hrRef, strs, StrLen(strs));
    h = HRFntLineHeight(hrRef);
  } else {
    w = FntCharsWidth(strs, StrLen(strs));
    h = FntLineHeight();
  }


  rect.topLeft.x= x;
  rect.topLeft.y= y;
  rect.extent.x= w;
  rect.extent.y= h;

  HRWinSetClip(hrRef, &rect);
  HRWinDrawChars(hrRef, strs, StrLen(strs), x, y);
}
```

Drawing with SmallFont/TinyFont fonts

Since the space between characters in the SmallFont/TinyFont series has enlarged by 1 Pixel horizontally and vertically, there are occasions when the display width is wider than before.

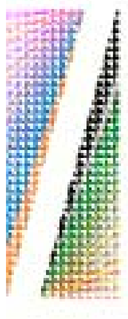
The size of Std/Large series fonts is the same as before.

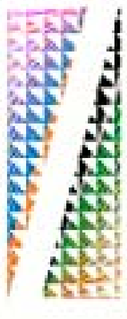
Drawing using Patterns

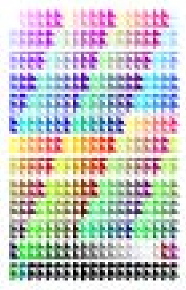
`Drawing in the display window with HRWinFillLine, HRWinFillRectangle, HRWinDrawGrayLine, HRWinPaint`, etc. series APIs is different than before; patterns are drawn such that compatibility is preserved with the large 160x160 system.

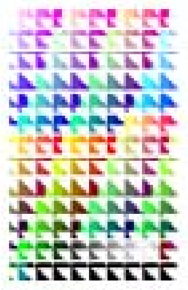
The results of specifying an 8 byte custom pattern and performing consective draws with `HRWinFillLine, HRWinFillRectangle` are shown below.

`HRWinFillLine` `HRWinFillRectangle`


Older system


PalmOS5.x


Older system


PalmOS5.x

The display from drawing to the offscreen is the same as the older HighRes system.

Modified drawing as implemented in PalmOS 5

With PalmOS 5, the frame width drawn by `WinDrawRectange`, etc. has changed. For details, refer to Palm OS Programmer's Companion, Palm OS Programmer's API Reference, etc.

Notes

Sublaunching

Take care of the screen mode when sublaunching other applications while inside an application, or when a program is sublaunched from another application.
Occasionally the display may become corrupted unless the screen mode is changed after menus, comand bars, pop up windows, etc, are deleted.

When sublaunching

When launching another application from inside of an application in high resolution mode, if that application does not support high resolution mode, the application will sublaunch after switching to normal mode.

When sublaunched

When a high resolution supporting application is launched from an application running in compatibility mode, the sublaunched application should first save the screen (using `WinSaveBits()`), and then switch to high resolution mode. When closing, switch to compatibility mode and redraw the saved screen (using `WinRestorBits()`).

Switching screen mode

Switching the screen mode takes a little time. Program to reduce the amount of screen mode switching.

NOTE: Since there is no compatibility mode on PalmOS 5 installed CLIÉ™s, the screen mode cannot be changed.

BmpCompress

For information on dealing with bitmaps larger than 64KB, refer to the Palm OS Programmer’s API Reference.

About HighRes Assist

NOTE: There is no HighRes Assist feature on CLIÉ™s installed with PalmOS 5.

The HighRes Assist feature does not depend on the availability of the high resolution API; as such existing appliations will run in high resolution mode. By using this feature, for applications that run on older devices the characters, etc. will simply become high resolution and the display will be better looking.
However, when using the HighRes Assist feature, applications may exhibit the following behavior.

- Applications will run significantly slower than before
  This is common with applications such as games, etc.
- Applications will not run correctly
  The display will appear shrunken and split into 2 into the upper half of the screen, characters will be shrunken and be unreadable, etc.

Especially, when applications run slowly, from the user's standpoint the application will occasionally appear to run correctly and be difficult to distinguish as slow. To prevent the above conditions from happening, and to run applications in compatibility mode without using HighRes Assist, study the code shown below. However, on occasion applications that offer the same features without using HighRes Assist can not be run in compatibility mode.

CASE 1: When screen mode is not changed outside of application startup

```
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: When screen mode changes occur within the applicaton at times other than at startup

```
#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);
  }

  ...

}
```

7 Virtual Silkscreen: Virtual Silkscreen Manager

Some CLIÉ handhelds replace the physical silkscreen traditionally seen in other Palm OS devices with an extended drawing area. This chapter describes how to take advantage of this extended drawing area and how to use it effectively.

Overview

The virtual silkscreen manager (hereafter referred to as the Silk Manager) manages and controls drawing to the extended drawing area on 320x480 and 480x320 CLIÉ handhelds and provides functions to control Silk Plug-ins.

Note that the Silk Manager is a CLIÉ proprietary feature and applications that utilize it may not be compatible with other Palm OS devices.

Feature Specifications

This section explains the display structure of the extended drawing area and related terminology. Refer to Figure 7-1 for graphical illustration.

Terminology

Wide-Resolution Screen	A screen with 320x480 (portrait) or 480x320 (landscape) display capability. The resolution on older Palm OS devices is 160x160 or 320x320.
Extended Drawing Area	The portion of a wide-resolution screen that extends upon older, square screens. On high-resolution devices, this area has dimensions of 320x160 or 160x320 pixels.
Application Drawing Area	The area where an application draws. Its size is normally 160x160 or 320x320, but on the devices with wide-resolution screens, it can include the extended drawing area by using the Silk Manager, increasing the area to to 320x480 (portrait) or to 480x320 (landscape).

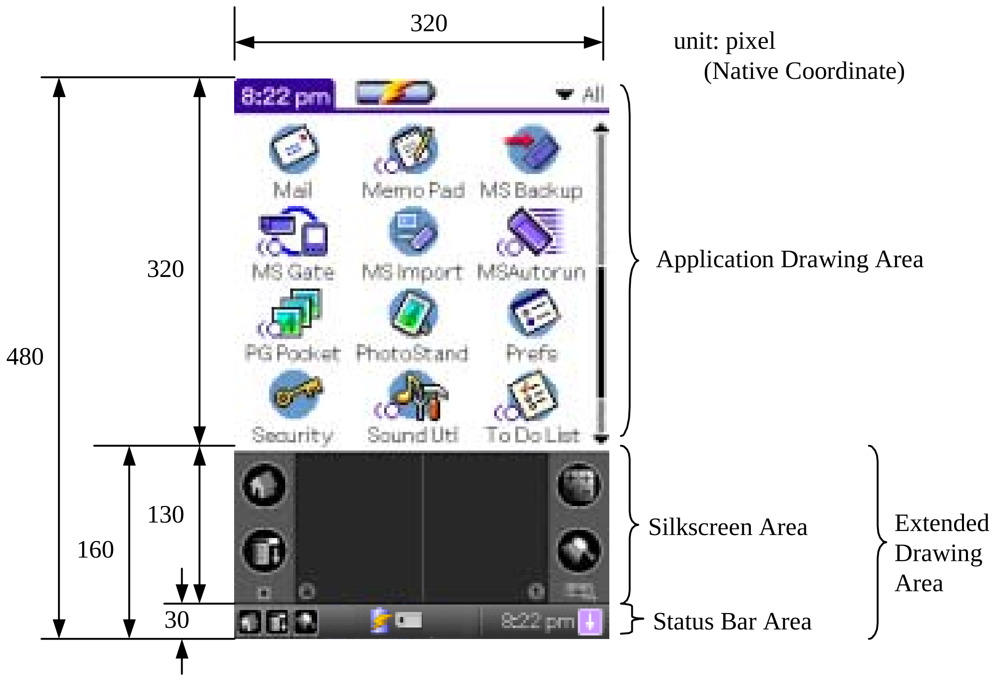
Silkscreen Area	The 320x130 pixel area in the upper part of the extended drawing area on wide-resolution, portrait-mode screens, or the corresponding 130x320 pixel area on wide-resolution, landscape-mode screens. While in older devices this area is fixed as the Graffiti input area.
Virtual Silkscreen	The silkscreen displayed by software into the silkscreen area.
Soft Graffiti	The virtual Graffiti area drawn by software in the silkscreen area. The printed, fixed Graffiti area in older devices is called Hard Graffiti.
Silk Plug-in[1]	Customized plugin software that runs on the silkscreen area.
Standard Input Plug-in	The Silk Plug-in that uses the silkscreen area for character input methods. As a standard feature of the CLIÉ, soft Graffiti and soft keyboard are offered as a type of Silk Plug-in[2].
Status Bar Area	The 320x30 pixel area in the bottom part of the extended drawing area on portrait-mode screens, or the corresponding 30x320 area on landscape-mode screens. Used by the system to display icons, system status, etc.
Maximizing	Maximizing the size of the silkscreen area.
Minimizing	Minimizing the size of the silkscreen area.

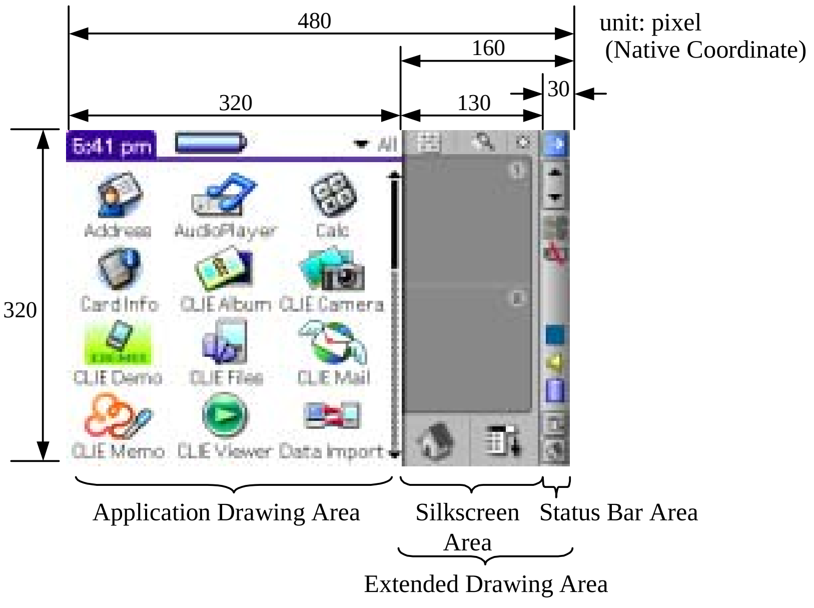
1. Silk Plug-ins are not supported on wide-resolution, landscape-mode devices.
2. Only Soft Graffiti is supported on wide-resolution, landscape-mode devices.

Display Structure

This shows the display structure of the extended drawing area.

Figure 7-1 Display Structure





User Interface Specifications

Resize

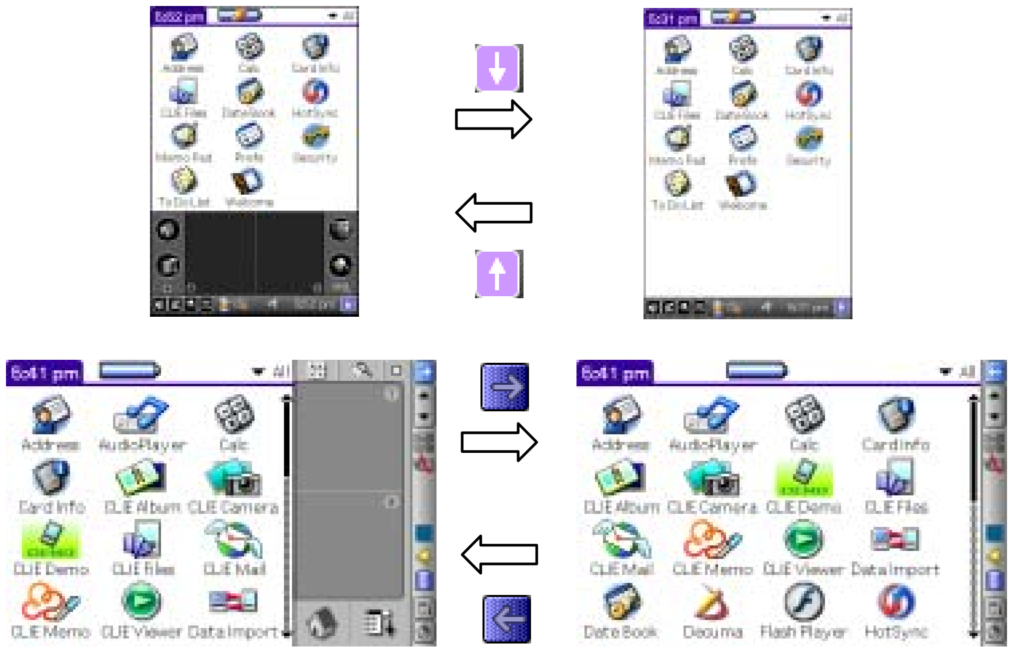
In general, applications that enable the extended drawing area allow the user to switch display size between 320x320 and 320x450 (or 450x320) sizes (refer to Figure 7-2) by tapping the resize button on the status bar area.

Some applications may change the display size automatically.

The status bar is always shown and cannot be minimized by the resize button.

Depending on the implementation of the application, resizing behavior may differ.

Figure 7-2 Changing Display Size



When the currently running application does not support the extended drawing area, the resize icon is disabled.

When moving from applications that support the extended drawing area and are drawing with a size of 320x450 (or 450x320) to applications that do not support the extended drawing area, the application drawing area automatically returns to 320x320, and the virtual silkscreen is displayed in the silkscreen area.

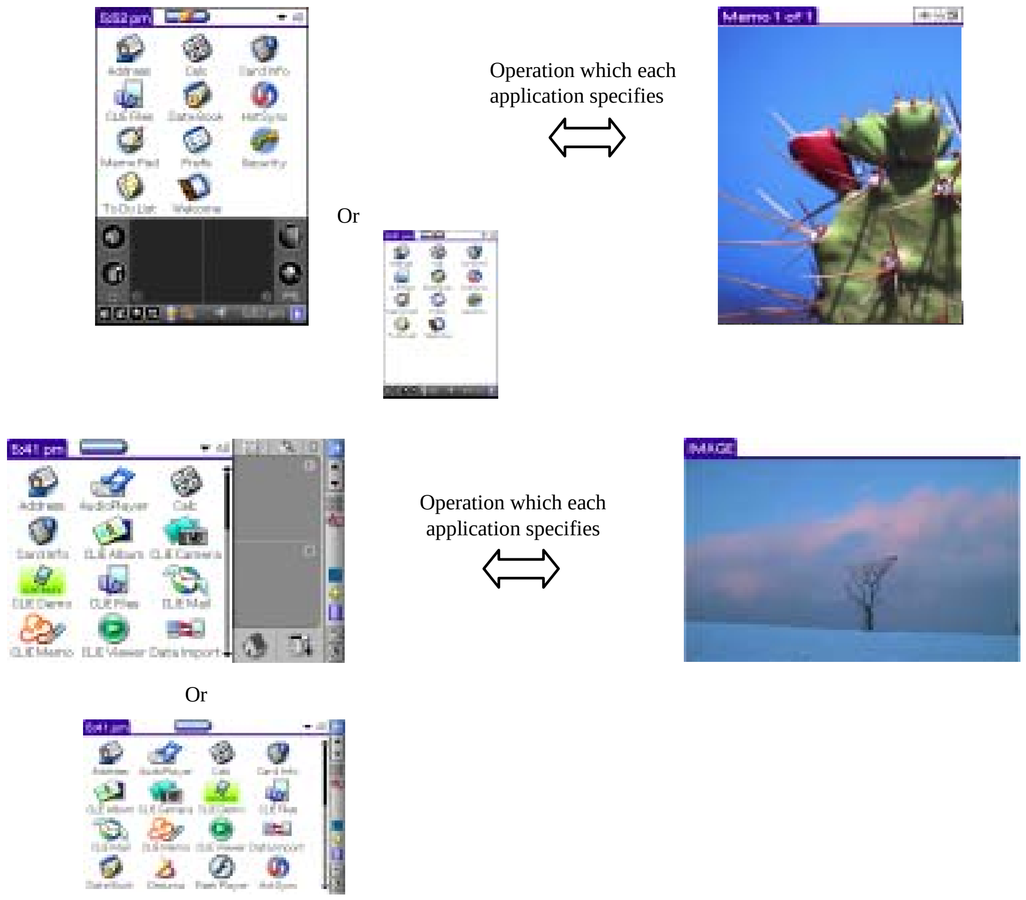
When menus and popups area displayed, the resizing occurs after they are cleared.

Full Screen Display

It is possible to program an application to utilize the full screen display, including the status bar area. However, there is no built-in resize icon to support changing the display to the full screen.

The method of switching between full screen and other screen sizes can differ from one application to another application. In general, it is recommended to use the back button to end the full screen display.

Figure 7-3 Full Screen Display


Operation which each application specifies
Or
Operation which each application specifies
Or

Using the Virtual Silkscreen

Functional Environment

The Silk Manager only supports the devices with a wide-resolution screen (native pixel dimensions of 320x480 or 480x320).

It is also recommended that an application checks the availability of the Virtual Silk Screen feature first before using it (refer to the following codes). It is also necessary to check the Silk Manager version because some features may not be available in earlier versions. Version 3 of the Silk Manager is required to support landscape-mode screens.

The Silk Manager API is offered as a shared library. To use the library, acquire the library reference number with `SysLibFind`.

Functions from version 1 of the Silk Manager API should be used only on models that support only that version (such as the PEG-NR70 series).

To determine the size of the application drawing area of a wide-resolution deviec, use `WinGetDisplayExtent` or `VskGetState`.

```
#include <SonyCLIE.h>

Err error = errNone;
UInt16 refNum;
UInt32 vskVersion;

if ((error = SysLibFind(sonySysLibNameSilk, &refNum))){
  if (error == sysErrLibNotFound) {
    /* couldn't find lib */
    error = SysLibLoad( 'libr', sonySysFileCSilkLib, &refNum );
  }
}

if (!error ) {
  error = FtrGet(sonySysFtrCreator, sonySysFtrNumVskVersion, &vskVersion);
  if (error) {
    /* Version 1 is installed
     only resize is available */
    if(SilkLibOpen (refNum)==errNone) {
      SilkLibEnableResize(refNum);
    }
  } else if (vskVersion == vskVersionNum2) {
    /* Version 2 is installed */

    if(VskLibOpen(refNum) == errNone) {
      VskSetState(refNum, vskStateEnable, vskResizeVertically);
    }
  } else {
     /* Version 3 or up is installed
        Horizontal screen is available */
     if(VskLibOpen(refNum) == errNone) {
       VskSetState(refNum, vskStateEnable, vskResizeHorizontally);
}
```

Resize

For an application to use the Silkscreen area, it must first declare that resizing the extended drawing area is supported by using the Silk Manager API (`VskSetState`) (See “Application Switching”). When resizing support is declared, Silk Manager enables the resize icon on the status bar.

The resize direction can be determined with `VskGetState` with version 3 or higher of the Silk Manager.

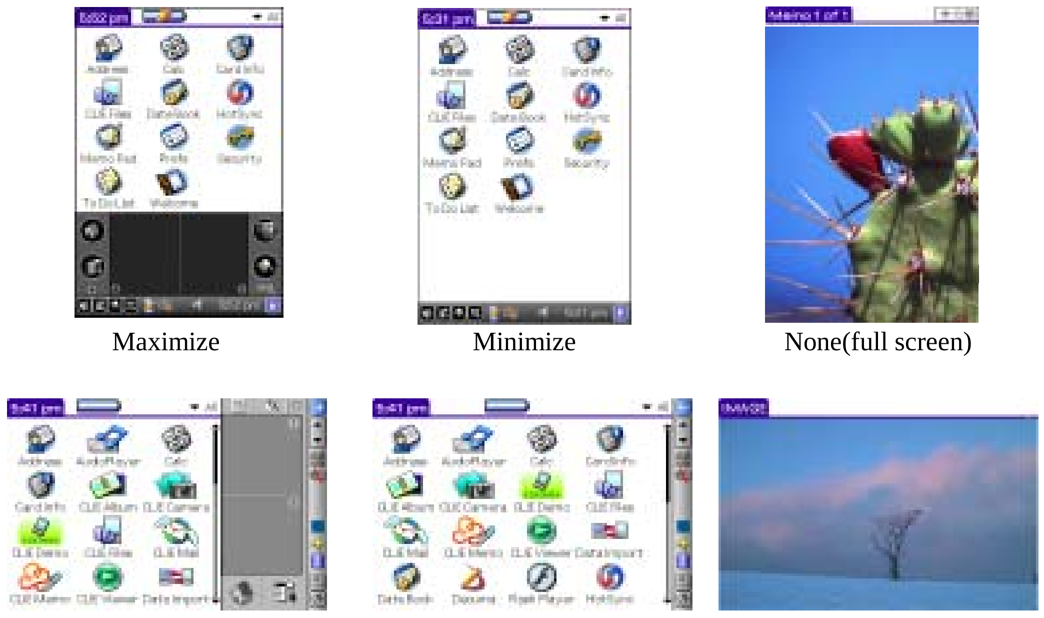
Minimizing and Maximizing the Silkscreen area is possible not only by tapping the resize icon, but also by applications using the Silk Manager API.

Full screen display is also possible by API calls.

The system changes the resize icon automatically based on the current silkscreen state.

A notification `(sysNotifyDisplayChangeEvent)` is issued when the screen size changes. Upon receiving such notification, an application has to manually change the size and/or position of forms, to reposition widgets, etc. (Refer to "Notifications".)

Figure 7-4 Maximize/Minimize /Full Screen Display



Maximize Minimize None(full screen)

Maximize Minimize None

Application Switching

When swtiching to another UI application, the Extended Drawing Area may be initialized as maximized or as unsupported. The Silk Manager detects the `sysAppLaunchFlagUIApp` flag in the `SysAppLaunch` API passed when the application is switched.

The actual drawing of the silkscreen , however, is delayed until the first `FrmDrawForm` call in the new application. If an application wants to start with the same silkscreen state as the previous application, it should call `VskSetState(...,vskStateEnable,vskResizeVertically)` (or `...,vskResizeHorizontally`) before the first `FrmDrawForm` call. This method prevents the flickering that occurs when the Silkscreen area is temporarily maximized

When it is necessary to know the state of the silkscreen directly from the previous application, call `WinGetDisplayExtent` before initializing the Silk Manager library and before the first `FrmDrawForm` call, or call `VskGetState(...,vskStateResize,...)` after `VskOpen`.

The Extended Drawing Area is not initialized when sublaunching applications. When sublaunching an application when in the minimized state, if the current application does not know whether the sublaunched application supports extended area drawing or not, it is recommended to maximize the silkscreen and to disable resizing before sublaunching and to reenable resizing and to restore the silkscreen state after sublaunched application returns.

The above silkscreen initialization principles from the application switching also apply to the full screen display.

The currently active Silk Plug-In is not changed when switching applications.

When the device is locked, the device switches to the Security application. Because of this, Extended Drawing Area initialization is performed.

Guidelines

The following guidelines are defined so that there will be common user operations and feels among applications regarding changing of the silkscreen area size. Unless there is a special reason, we recommend following these guidelines.

Do not only support minimizing

It is assumed that character input on Palm OS devices is performed via Graffiti on the virtual silkscreen. Because of this, applications should support displaying the silkscreen area at any time.

However, there is no restriction against starting the application with the minimized silk screen area.

Don’t prevent the user from maximizing

For the same reason as above, an application should not temporarily disable maximizing the silkscreen area.

To avoid confusing users by continually enabling or disabling the silkscreen area, it is recommneded to allow users to be able to minimize/maximize whenever they want.

This restriction does not mean that features must be offered the same when in minimized state and maximized state. In other words, there is no rule against partially limiting features offered in minimized state from those offered when the silkscreen area is maximized.

Although it is not recommended that the minimize feature be dynamically restricted, it is sometimes necessary to disable the minimize feature temporarily because not all the forms in an application may support resizing the virtual silkscreen.

Don’t rashly use full screen display

The status bar is displayed even when the silkscreen area is in the minimized state. This is necessary to enable Palm OS devices’ basic operations, such as “one tap return to home,” “global search,” etc., no matter what kind of application is currently running. For this reason, it generally is recommended to avoid using full screen display.

Full screen display should be used only for cases, such as when size calculations become more difficult with 320x450 than in 320x480, when the presence of the status bar significantly impacts aesthetics, etc.

Obey the guide for recovery methods from full screen display

Generally, since the basic operational buttons are not displayed while in the full screen display, it is important to offer a unified interface among applications on how the user can end full screen display state and return to a state with the status bar.

Although there are many ways of returning from the full screen display, we recommend an application to implement the below methods.

- When the back button is pressed, full screen display ends and the status bar displayed state returns.
- If the area normally occupied by the status bar is tapped, the status bar should be displayed.

Also, always respond to pen taps and show information on display updates including how to end the full screen display. Use sound cues only if no alternatives are not possible. To improve user friendliness, provide visual cues to indicate how to restore the status bar, and do not rely on experimentation from the user.

Special issues

List

When the silkscreen area is maximized/minimized while a list is being displayed, the Silk Manager erases the list. This occurs only when an application uses `LstPopupList` to display the list. The list is not automatically erased if drawn by other methods. Consequently, there are occasions in which the silkscreen area will overwrite the list.

Saving Silkscreen drawing area

If the silkscreen area is maximized/minimized when a modal form (e.g., system Find Dialog) is the current active drawing window, the underneath application form should not be resized right away because it will overwrite the top modal form. As a result, when there is a modal form on top of an application, when minimizing the silkscreen area from a maximized state, there are occasions where this area will be blanked out. To avoid this, the Silk Manager saves this area (`WinSaveBits`) when maximized, and restores it (`WinRestoreBits`) after minimizing. The requirement for this feature to be enabled is that the lower bound of the 2nd and higher forms on the top can not exceed 320. Moreover, applications need to call `FrmEraseForm` before re-drawing a form. When applications continue a `FrmDrawForm` without performing a `FrmEraseForm`, this feature does not work effectively.

Pen events

When the pen is put over the Graffiti input area when the Graffiti recognition feature is disabled, the penup event is issued to the current running application. It is the same for pen buttons. This is a PalmOS specification.

When a UI application switching is performed when the silkscreen area is not in the maximized state, (when the `SysAppLaunch` API's `sysAppLaunchFlagUIApp` flag is passed through), `EvtFlushPenQueue` is executed so that pen events tapped into the silkscreen area do not have an effect on the next application. However, pen events are not flushed during a sublaunch.

Silk Manager Reference

Virtual Key

The Silk Manager receives the following virtual keys and performs various corresponding operations.

`vchrSilkResize`	Alternately performs a maximize/minimize.
`vchrSilkLoader`[1]	Starts the Silk Plug-in loader.
`vchrSilkChangeSlkw`[1]	When the keyCode is `keyCodeSilkPrev`, starts the previous Silk Plug-in, and when the keyCode is `keyCodeSilkInputDef`, starts the default Silk Plug-in for character input.

Normally, the status bar issues these virtual keys and the Silk Manager processes them appropriately.

Notifications

Silk Manager issues the following notifications.

sysNotifyDisplayChangeEvent

When the size of the application drawing area changes Silk Manager broadcasts `sysNotifyDisplayChangeEvent`.

After an application receives this Notification, it can determine the display window size and redraw accordingly. Also be aware of the following items:

- This Notification is also issued for various cases such as depth and color palette change, do the redawing only when it is necessary. Refer to the PalmOS documentation for more details on this Notification.
- Silk Manager always minimizes/maximizes when there is a request from the applications that support the extended drawing area (Resize Enabled state). This is the same for when the PalmOS is displaying popup dialogs in the current application. In this case, the resizing cannot be performed until the popup dialog closes. It is necessary to set an internal flag when receiving the resizing request while a popup dialogs is active , and based on this flag state, perform the redraw processing after the dialog is dismissed and the application returns to its own

1. Not supported on wide-resolution, landscape-mode devices.

event loop. This method prevents the undesired drawing over the popup dialog (the currently active drawing window). For the extended drawing area, Silk Manager handles the redrawing process automatically.

IMPORTANT: On some PalmOS4.x(68K) devices the system sometimes automatically sets ResizeDisable. This is mainly so that when the PalmOS occasionally lays forms across the extended drawing area, the soft graffiti is not overwritten.

IMPORTANT: The PalmOS5 Silk Manager moves all the forms displayed by the PalmOS so that their Y coordinate is 320 and lower (with PalmOS5 there are no system forms beyond 320 vertically, so this problem does not occur).

- The size of the application drawing area can be acquired with `VskGetState or with WinGetDisplayExtent in the` PalmOS API.
- Applications should not depend on having the extended drawing area size be 320x450 pixels. Always use the API to acquire the current draw window size, and perform draw processing based on that size.
- By the `VskSetState(..., vskStatusResize, ...)` APIs, when the application itself changes the drawing area size, have the application redraw after receiving this Notification.
- Silk Manager does not automatically change Forms, Windows, and Bitmaps managed by the application. When unable to draw to the silkscreen area, check the size of Forms or the Windows it contains. Consequently, with the above check, an application may overwrite the virtual silkscreen (Silk Plug-in). The only Window changed by the Silk Manager is the one returned by `WinGetDisplayWindow`.

<Example: changing the size of myForm>

```
RectangleType rect;
FormPtr frmP = myForm;

rect.topLeft.x = 0;
rect.topLeft.y = 0;
WinGetDisplayExtent(&rect.extent.x, &rect.extent.y);
WinSetBounds(FrmGetWindowHandle(frmP), &rect);
```

- Silk Manager receives this notification with a priority of -126. Generally, have applications receive this Notification with priority `sysNotifyNormalPriority`.
- When the application exits, make sure to unregister the notification.

`SysNotifyParamType` attribue values

`notifyType`	`sysNotifyDisplayChangeEvent`

`broadcaster`	`sysNotifyBroadcasterCode`
`notifyDetailsP`	`SysNotifyDisplayChangeDetailsType*`

sysNotifyPalmSilkChangeEvent

When the graffiti screen is changed, `sysNotifyPalmSilkChangeEvent` is broadcast. The graffiti screen change indicates whether the soft graffiti is displayed in the silkscreen area or not. However, this notification does not correspond to the redrawing due to minimization/maximization of the silkscreen area.

In general, applications do not need to receive this Notification. It is mainly for the utilities that assume the graffiti is being displayed.

`SysNotifyParamType` attribute values

`notifyType`	`sysNotifyPalmSilkChangeEvent`
`broadcaster`	`sysNotifyBroadcasterCode`

`notifyDetailsP`	Pointer to UInt32 type. The value pointed to is one of the following 2.
	`sysNotifyPalmSilkAppeared(1)`
	When the graffiti screen has appeared.
	`sysNotifyPalmSilkDisappeared(0)`
	When the graffiti screen has disappeared.

Silk Manager API

VskOpen

Purpose Begin usage of the virtual silkscreen manager

Prototype `Err VskOpen(UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No error
`vskErrNotAvailable`	Virtual silkscreen manager can not be used

Comments Equivalent to `SilkLibOpen`.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskClose

Purpose End usage of the virtual silkscreen manager

Prototype `Err VskClose(UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No error
`vskErrNotOpen`	The library is not open
`vskErrStillOpen`	The library is still Open

Comments Equivalent to `SilkLibClose`.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskGetAPIVersion

Purpose Get the API version for the library

Prototype `UInt32 VskGetAPIVersion(UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`0x00000002`	API version for CLIÉ handhelds running PalmOS5
`0x00000003`	API version for CLIÉ handhelds that support landscape-mode screens

VskSetCurrentSlkw[1]

Purpose Switch from the current Silk Plug-in to the specified one

Prototype

```
Err VskSetCurrentSlkw(UInt16 refNum, UInt16 slkwType,
UInt32creator)
```

Parameters

`-> refNum`	Reference number for the library
`-> slkwType`	reserved. Set as 0.

[1] Not supported on wide-resolution, landscape-mode devices.

	`-> creator`	Creator ID of the Silk Plug-in
Result	`errNone`	
	`vskErrNotOpen`	The library is not open
	`vskErrCannotFind`	

Comments Use when changing the Silk Plug-in. Usage of the Silk Plug-in loader is assumed.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskGetCurrentSlkw[1]

Purpose Get the Creator ID of the current Silk Plug-in

Prototype
```
Err VskGetCurrentSlkw(UInt16 refNum, UInt16 slkwType,
UInt32* creator)
```

Parameters	`-> refNum`	Reference number for the library
	`-> slkwType`	reserved. Set to 0.
	`<-  creator`	Creator ID of the current Silk Plug-in.
Result	`errNone`	
	`vskErrNotOpen`	The library is not Open

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskGetState

Purpose Get the state of the Silk Manager

Prototype
```
Err VskGetState(UInt16 refNum, UInt16 type, UInt16* state)
```

Parameters	`-> refNum`	Reference number for the library
	`-> type`	The state type to get. See **Comments**.

[1] Not supported on wide-resolution, landscape-mode devices.

`<- state`	The value of the specified state. See **Comments**.

Result

`errNone`	
`vskErrNotOpen`	The library is not Open.
`vskErrParamErr`	

Comments The state types and values supported are specified in the table below:

	type	state	Explanation
The current resize state of the silkscreen	`vskStateResize`	`vskResizeMax`	The silkscreen is maximized
		`vskResizeMin`	The silkscreen is minimized
		`vskResizeNone`	The silkscreen is completely hidden
Whether silkscreen resizing is enabled	`vskStateEnable`	`vskResizeDisable`	Resizing is disabled
		`vskResizeVertically`	Vertical resizing is allowed
		`vskResizeHorizontally`[a]	Horizontal resizing is allowed
The supported silkscreen resizing direction	`vskStateResizeDirection`[a]	`vskResizeDisable`	Resizing is disabled
		`vskResizeVertically`	Vertical resizing is supported
		`vskResizeHorizontally`	Horizontal resizing is supported
Whether Silk Plug-ins are supported	`vskStateSilkPlugInAvailable`[a]	`vskSilkPlugInNotAvailable`	Silk Plug-ins are not supported
		`vskSilkPlugInAvailable`	Silk Plug-ins are supported

a. Available only if `VskGetAPIVersion()` returns 0x3 or higher

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskSetState

Purpose Set the state of the Silk Manager

Prototype `Err VskSetState(UInt16 refNum, UInt16 type, UInt16 state)`

Parameters

`-> refNum`	Reference number for the library

`-> type`	The state type to set. See **Comments**.
`-> state`	The new value for the state. See **Comments**.

Result

`errNone`	
`vskErrNotOpen`	The library is not Open
`vskErrParamErr`	

Comments As the result of specifiying `vskStateResize` when calling this API, `sysNotifyDisplayChangeEvent` is issued.

The state types and values supported are specified in the table below:

	type	state	Explanation	
			Portrait-mode device	**Landscape-mode device**
Set the resize state of the silkscreen	`vskStateResize`	`vskResizeMax`	Maximizes the silkscreen	
		`vskResizeMin`	Minimizes the silkscreen	
		`vskResizeNone`	Completely hides the silkscreen	
Enables or disables silkscreen resizing	`vskStateEnable`	`vskResizeDisable`	Disables resizing	Disables resizing
		`vskResizeVertically`	Enables vertical resizing	No effect
		`vskResizeHorizonally`[a]	No effect	Enables horizontal resizing

a. Available only if `VskGetAPIVersion()` returns 0x3 or higher

If an application supports both vertical and horizontal resizing, it can set the value for `vskStateEnable` to `(vskResizeVertically|vskResizeHorizontally)`.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskGetPalmSilkEnabled

Purpose Return whether the Graffiti interpretation engine and pen buttons are enabled/disabled.

Prototype

```
Err VskGetPalmSilkEnabled(UInt16 refNum, Boolean *graffiti,
Boolean *penButton)
```

Parameters

`-> refNum`	Reference number for the library

	`<- graffiti`	Whether Graffiti interpretation engine and also pen buttons are enabled or disabled.
	`<- penButton`	Whether pen buttons only is enabled or disabled.
Result	`vskErrNotOpen`	The library is not Open

Comments Currently, only returns the `VskEnablePalmSilk()` setting and whether or not pen buttons is enabled.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

VskDoCommand

Purpose Issue a command to the active Silk Plug-in

Prototype

```
Err VskDoCommand(UInt16 refNum, UInt32 creator,
UInt16command, UInt32 data1, UInt32 data2)
```

Parameters	`-> refNum`	Reference number for the library
	`-> creator`	Creator ID of the Silk Plug-in
	`-> command`	Command (depends on Silk Plug-in)
	`-> data1`	Data (Depends on the command)
	`-> data2`	Data (Depends on the command)
Result	`errNone`	
	`vskErrNotOpen`	The library is not Open
	`vskErrParamErr`	

Comments The data is interpreted as a string of bytes. When exchanging information between 68K code and ARM code, take care as the Endian/Byte Alignment is different.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x02 or higher

Silk Manager API (For Version 1 compatibility)

IMPORTANT: The API below cannot be used if the value returned by `VskGetAPIVersion()` is 0x3 or higher. Even if the value of `VskGetAPIVersion()/SilkLibGetAPIVersion()` is 0x2, developers are recommended to use the Vsk... functions.

SilkLibOpen

Purpose Begin usage of the virtual silkscreen manager

Prototype `Err SilkLibOpen ( UInt16 refNum )`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No error
`silkLibErrNotAvailable`	The virtual silkscreen manager can not be used

Comments Performs processing for using this library.

Compatibility If the value for `VskGetAPIVersion()` is 0x02, usage of `VskOpen()` is recommended.

SilkLibClose

Purpose Ends usage of this library

Prototype `Err SilkLibClose ( UInt16 refNum )`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No error
`silkLibErrNotOpen`	The library is not open
`silkLibErrStillOpen`	The library is still Open

Comments Performs processing for closing the library.

Compatibility If the value for `VskGetAPIVersion()` is 0x02, usage of `VskClose()` is recommended.

SilkLibEnableResize

Purpose Enable resizing the height of the application drawing area and the up/down arrows (resize icon) in the status bar.

Prototype `Err SilkLibEnableResize (UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No error
`silkLibErrResizeDisabled`	Resizing is not allowed.

Comments It is recommended to call this API at application initialization (before receiving `frmOpenEvent`, or executing `FrmDrawForm()`).

When this API is executed, the minimize/maximize icons in the status bar become enabled.

Compatibility If the value for `VskGetAPIVersion()` is 0x02, usage of `VskSetState()` is recommended.

SilkLibDisableResize

Purpose Prevent resizing of the application drawing area's height and disable the up/down arrow buttons on the status bar.

Prototype `Err SilkLibDisableResize (UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`errNone`	No errors

Comments Normally, used as the opposite of `SilkLibEnableResize ()`.

Compatibility If the value for `VskGetAPIVersion()` is 0x02, usage of `VskSetState()` is recommended.

SilkLibResizeDispWin

Purpose Change the height of the application drawing area. At the same time, display or hide the active input area and the status bar area. The height can be set to 3 values.

Prototype `Err SilkLibResizeDispWin( UInt16 refNum, UInt8 pos )`

Parameters

`-> refNum`	Reference number for the library
`-> pos`	Size after application drawing area change. Only the following values area allowed.

`silkResizeNormal`
Normal application drawing area height

`silkResizeToStatus`
Height where only the status bar is displayed

`silkResizeMax`
Height of entire display (extended drawing area is hidden)

Result

`errNone`	No error
`silkLibErrResizeDisabled`	Shows that change could not be performed

Comments To use this API, applications must have notified the library that changing the drawing area size is possible. For the notification method refer to `SilkLibEnableResize()`. Since when the application drawing area is maximized the status bar area is hidden, applications must offer to the user a way to return the drawing area to its previous state. Since the notification that the drawing area has actually change via this API is performed by the `sysNotifyDisplayChangeEvent` Notification, perform any post-change drawing after this Notification. For details, refer to Notifications.

Compatibility Can be used only if `VskGetAPIVersion()`returns 0x01 or higher.

SilkLibGetAPIVersion

Purpose Get the API version for the library

Prototype `UInt32 SilkLibGetAPIVersion(UInt16 refNum)`

Parameters

`-> refNum`	Reference number for the library

Result

`0x00000001`	API version for CLIÉ equipped with PalmOS4.x
`0x00000002`	API version for CLIÉ equipped with PalmOS5

8

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.

9

JPEG Utility: Sony JpegUtil Library

The Sony JpegUtil Library offers to applications functions for the handling of JPEG images. By using the Sony JpegUtil Library, JPEG images taken with digital cameras and other devices can be converted to bitmap format, displayed on the screen, etc. A utility API to convert images from the PictureGearPocket format, currently one of the image formats used on the CLIÉ™, into JPEG images has also been provided.

Also, there is an built-in camera on some CLIÉ™s. For devices such as these, it is possible to use the built-in camera as a digital camera to take still images and store them on a Memory Stick in JPEG format (DCF format: digital camera standard format).

Through the offering functions such as these, better, more visually oriented applications can be realized.

Function specifications

This section explains details regarding the functions offered by Sony Jpeg Util Library.

Function list

The following utility APIs for decoding/encoding JPEGs are offered.

- Encoding screen data or bitmap data into JPEG images.
- Decoding JPEG data into bitmap data, or displaying on the screen.(considers version 2.0 and later Exif Orientation tag)
- Acquiring JPEG image information (image size, date or other Exif main information)(Supports acquisition of ver.2.0 and later Orientation information)
- Converting (encoding) from PGPF (Picture Gear Pocket Format) database into JPEG images.
- Acquiring the progress when decoding/encoding, as well as supporting in progress cancel.
- Setting of ver.2.0 and later JPEG Orientation information is possible (if Exif Orientation tag is provided)
- Subject JPEGs are those stored in a Memory Stick or in Memory.

However, Progressive JPEGs are not supported.

When dealing with the DCF format that is used as the save format in digital cameras, please support this through the application.

Using the JPEG utility

Loading the library

To use the library, it is necessary to have the library loaded and then acquire a library reference number. An example of the process for this is shown below.
However, `sonySysFileCJpegUtilLib` et al, are defined in `SonySystemResources.h`.

```
#include <SonyCLIE.h>

UInt16 refNum; /* Library Reference Number */
Err err;

/* Checking if library is loaded and acquiring the number */
err = SysLibFind(sonySysLibNameJpegUtil, &refNum);
if (err)
{
  /* If the library is not loaded */
  err = SysLibLoad(sonySysFileTJpegUtilLib,
                   sonySysFileCJpegUtilLib, & refNum);
  if (err)
  {
    /* Failure in loading the Sony JpegUtil Library */
  }
}
```

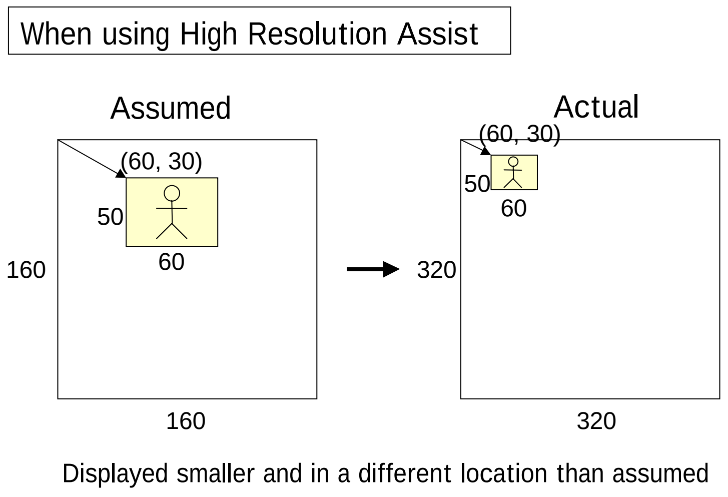
Relation of JPEG utility to Resolution

NOTE: There is no HighRes assist feature on CLIEs installed with PalmOS 5.

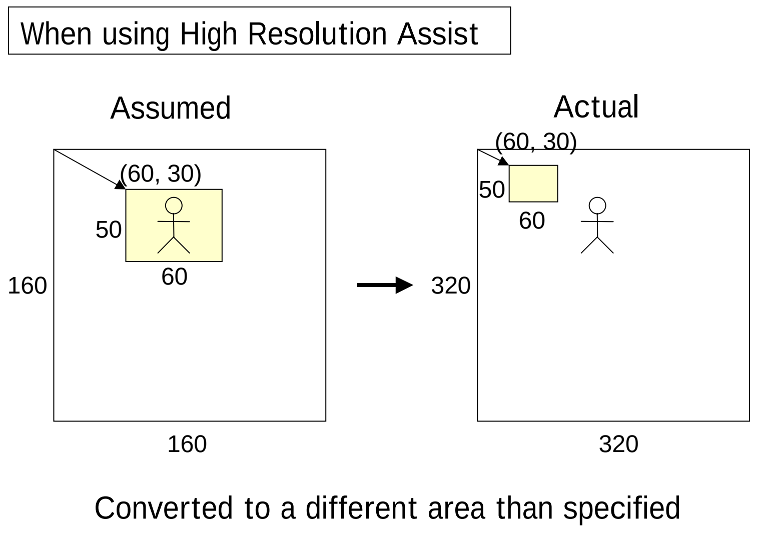
When displaying in high resolution mode using HighRes Assist, etc. and not using the Sony HR Library, on CLIÉ™s that support high resolution mode, the following 2 APIs in the Sony JpegUtil Library will not operate properly.

```
jpegUtilLibDecodeImageToWindow()
jpegUtilLibEncodeImageFromWindow()
```

In the case of jpegUtilLibDecodeImageToWindow()


When using High Resolution Assist
Assumed
Actual
(60, 30)
50
60
160
160
(60, 30)
50
60
320
320
Displayed smaller and in a different location than assumed

In the case of jpegUtilLibEncodeImageFromWindow()


When using High Resolution Assist
Assumed
Actual
(60, 30)
50
60
160
160
(60, 30)
50
60
320
320
Converted to a different area than specified

Encoding/decoding progress display and cancel notification

With the Sony JpegUtil Library, a system for acquiring the progress of the encoding/ decoding and for canceling the encoding/decoding in progress is provided. To take advantage of this system, use Sony's `PrgInfoType` structure. Note that this structure is not part of the standard Progress Manager API provided by Palm OS.

Utilizing the PalmOS standard ProgressDialog

The application should use the `PrgInfoType` structure and perform the following before starting the encode/decode.

```
PrgInfoType prgInfo;
prgInfo.prgCbFunc = NULL;
// When using the OS standard ProgressDialog set to NULL.
prgInfo.prgP = PrgStartDialog(“Decode”, textCb, NULL);
```

Here, the first argument is the Dialog Title (`“Decode”`), the second argument is the `TextCallback` function (`textCb`), and the third argument represents a pointer to the data used by `textCb`; in cases where this is not needed, set to NULL.

An example of the `textCallback` function is shown here.

```
Boolean textCb(PrgCallbackDataPtr cbP)
{
  StrPrintF(cbP->textP, “%d%% done”, cbP->stage);
  return true;
}
```

If this `prgInfo` is specified and the API for encoding/decoding is called, with the Sony JpegUtil Library, `txtCb` is called at certain intervals during the encode/decode. `PrgCallbackDataPtr` is specified when calling the Callback function and in a member of this structure, `stage(UInt16)`, a value representing the executed percentage of the encode/decode is substituted.
It is possible to display into the Dialog by inserting the text to be viewed into `textP` of the same structure.
Call `PrgStopDialog` when the encode/decode finishes.

For canceling of the encode/decode, when the Cancel button in the displayed Dialog is tapped, the cancel process is automatically performed.

For further details, refer to the description of the Progress Manager in the PalmOS Programmer's Companion, PalmOS Programmer's API Reference.

Utilizing the original Callback function

The application should use the `PrgInfoType` structure and, for example, perform something similar to the following before starting the encode/decode.

```
PrgInfoType prgInfo;
prgInfo.prgP = NULL;
prgInfo.prgCbFunc = jpegCallbackFunc;
                        // original callback function
```

```
    prgInfo.prgCbData = &(prgInfo.percent);
                               // data used in callback function
```

If this `prgInfo` is specified and API for encoding/decoding is called, with the Sony JpegUtil Library, `jpegCallbackFunc` is called at certain intervals during the encode/decode.
Applications can provide a system for displaying the progress and at the same time making cancel possible within `jpegCallbackFunc`. Progress information can be acquired from `prgInfo.percent`.
Also, it is necessary to perform the handling of canceling with `jpegCallbackFunc`. To inform the Sony JpegUtil Library of a cancel, set the returned value of `jpegCallbackFunc` to true. Oppositely, to continue encoding/decoding, return a false.

Also, for the system to be able to handle Events during JPEG encoding/decoding, it is recommended to describe the Callback function as in the following example.

```
Boolean jpegCallbackFunc(void *prgCbData)
{
  UInt16 percent = (UInt16) *(UInt16 *)prgCbData;
  EventType ev;
  Char s[20];

   if(EvtSysEventAvail(true)) {
     return true;
  }

  MemSet(s, sizeof(s), 0);
  StrPrintF(s, "%d%% done", percent);
  WinDrawChars(s, StrLen(s), 10, 150); // Progress display
  return false;
}
```

Not utilizing progress display and cancel

If displaying progress and allowing the user to cancel the operation are not necessary, pass NULL as the `prgInfoP` argument.

JPEG Utility API

Data Structures

This section lists the data structures defined by the Sony JpegUtil Library.

JpegUtilLibErr

Errors for the Sony JpegUtil Library module.

Value Descriptions

`jpegUtilLibErrNone`	Success.
`jpegUtilLibErrBadParam`	Parameters are incorrect.
`jpegUtilLibErrNotOpen`	Library is not open.
`jpegUtilLibErrStillOpen`	Library is still open.
`jpegUtilLibErrNoMemory`	Insufficient memory.
`jpegUtilLibErrNotSupported`	Unsupported function.
`jpegUtilLibErrNotJpegFormat`	Not JPEG format.
`jpegUtilLibErrNotExifFormat`	Not Exif format.
`jpegUtilLibErrEncDecCanceled`	Encode/decode cancelled.
`jpegUtilLibErrResourceBusy`	Resource is busy.

JpegImageType

Type of JPEG image.

```
typedef enum {
  jpegDecModeNormal = 0,
  jpegDecModeThumbnail
} JpegImageType;
```

Value Descriptions

`jpegDecModeNormal`	Main image.

`jpegDecModeThumbnail`	Thumbnail image (Exif compliant JPEG file).

Comments The requirements for an image to be a Thumbnail are as shown below.

- Exif2.1 specification
- JPEG format thumbnail
- Size is 160x120

JpegImageRatio

Linear scaling factor of JPEG image.

```
typedef enum {
  jpegDecRatioNormal = 0,
  jpegDecRatioHalf,
  jpegDecRatioQuarter,
  jpegDecRatioOctant
} JpegImageRatio;
```

Value Descriptions

`jpegDecRatioNormal`	1:1
`jpegDecRatioHalf`	2:1
`jpegDecRatioQuarter`	4:1
`jpegDecRatioOctant`	8:1

JpegDetailInfoCapabilityType

Structure that shows capability information of the JPEG file.

```
typedef struct {
  UInt16 softName:1;
  UInt16 gpsInfo:1;
  UInt16 orientation:1;
  UInt16 reserved:13;
} JpegInfoCapabilityType;
```

Field Descriptions

`softName`	Software name Capability
`gpsInfo`	GPS information Capability
`orientation`	Image orientation (valid with ver.2.0 and later)
`reserved`	reserved

Comments Various flag information for `JpegInfoCapabilityType`:
Information set by Callee (function side).
Setting by the Caller (calling side) is undesired.

JpegDetailInfoType

JPEG file information structure.

```
typedef struct {
  JpegDetailInfoCapabilityType jpegDetailInfoCapability;
  Char dateTime[20];
  Char *softName;
  GPSInfoP gpsInfoP;
  Char dateTimeOriginal[20];
  Char dateTimeDigitized[20];
  UInt16 orientation;
} JpegDetailInfoType, *JpegDetailInfoP;
```

Field Descriptions

`jpegDetailInfoCapability`	JpegDetailInfo capability.
`dateTime`	Date and Time information (ASCII) (e.g., 2001:10:23:21:03:45)
`softName`	Software name.
`gpsInfoP`	Pointer to GPS information.
`dateTimeOriginal`	Date and time original image was taken (e.g.: 2001:10:23:21:03:45) (valid with ver.2.0 and later)
`dateTimeDigitized`	Date and time image was digitized after being taken (e.g.: 2001:10:23:21:03:45) (valid with ver.2.0 and later)
`orientation`	Image orientation (valid with ver.2.0 and later)

RationalType

JPEG and Exif parameter fraction structure.

```
typedef struct {
  UInt32 numerator;
  UInt32 denominator;
} RationalType;
```

Field Descriptions

`numerator`	Numerator.
`denominator`	Denominator.

GPSInfoCapabilityType

Structure that shows GPS information capabilities.

```
typedef struct {
  UInt16 version:1;
  UInt16 latitudeRef:1;
  UInt16 latitude:1;
  UInt16 longitudeRef:1;
  UInt16 longitude:1;
  UInt16 altitudeRef:1;
  UInt16 altitude:1;
  UInt16 mapDatum:1;
  UInt16 reserved:8;
} GPSInfoCapabilityType;
```

Field Descriptions

`version`	Version Capability.
`latitudeRef`	Latitude reference Capability as North(N) or South(S).
`latitude`	Latitude information Capability.
`longitudeRef`	Longitude reference Capability as East(E) or West(W).
`longitude`	Longitude information Capability.
`latitudeRef`	Altitude reference (0:sea level) Capability.
`latitude`	Altitude information Capability.
`mapDatum`	Survey name Capability (“TOKYO” or “WGS-84”)
`reserved`	reserved

GPSInfoType

GPS information structure.

```
typedef struct {
  GPSInfoCapabilityType gpsInfoCapability;
  Char version[4];
  Char latitudeRef[2];
  RationalType latitude[3];
  Char longitudeRef[2];
  RationalType longitude[3];
  Char altitudeRef;
  RationalType altitude;
  Char *mapDatum;
} GPSInfoType, *GPSInfoP;
```

Field Descriptions

`gpsInfoCapability`	GpsInfo member Capability.

`version`	Version.
`latitudeRef`	Latitude reference as North(N) or South(S).
`latitude`	Latitude information.
`longitudeRef`	Longitude reference as East(E) or West(W).
`longitude`	Longitude information.
`latitudeRef`	Altitude reference (0:sea level).
`latitude`	Altitude information.
`mapDatum`	Survey name (“TOKYO” or ”WGS-84”).

JpegPrgCallbackFunc

Pointer to the Callback function used to acquire the progress.

```
typedef Boolean (*JpegPrgCallbackFunc)(void *);
```

PrgInfoType

Encode/Decode progress structure.

```
typedef struct {
  UInt16 percent;
  ProgressPtr prgP;
  JpegPrgCallbackFunc prgCbFunc;
  void *prgCbData;
} PrgInfoType, *prgInfoP;
```

Field Descriptions

`percent`	Encode/Decode progress (%).
`prgP`	Pointer acquired with PrgStartDialog (refer to Progress.h).
`prgCbFunc`	Callback function pointer.
`PrgCbData`	Pointer to data used by the Callback function.

JpegImageOrientation

Structure that shows the image orientation

```
typedef enum {
  JpegImageOriNotSupported = 0,
  JpegImageOriNormal,
  JpegImageOriNormalR,
  JpegImageOriCW180,
  JpegImageOriCW180R,
  JpegImageOriCW90R,
  JpegImageOriCW90,
  JpegImageOriCCW90R,
     JpegImageOriCCW90
   } JpegImageOrientation;
```

Field descriptions

`JpegImageOriNotSupported`
Unsupported orientation

`JpegImageOriNormal`
Normal

`JpegImageOriNormalR`
Reversed from normal (NotSupported)

`JpegImageOriCW180` 180º

`JpegImageOriCW180R`
reversed 180º (NotSupported)

`JpegImageOriCW90R` 90º `clockwise rotation and reversed` (NotSupported)

`JpegImageOriCW90` 90º clockwise rotation

`JpegImageOriCCW90R`
90º Counterclockwise rotation and reversed (NotSupported)

`JpegImageOriCCW90` 90º counterclockwise rotation

Comments Currently, reversed images are not supported (valid with ver.2.0 and later)

System I/F API

jpegUtilLibOpen

Purpose Open the Sony JpegUtil Library.

Prototype `Err jpegUtilLibOpen( UInt16 jpegUtilLibRefNum );`

Parameters `-> jpegUtilLibRefNum`
Sony JpegUtil Library reference number.

Result Please refer to `JpegUtilLibErr`.

jpegUtilLibClose

Purpose Close the Sony JpegUtil Library.

Prototype

```
Err jpegUtilLibClose( UInt16 jpegUtilLibRefNum );
```

Parameters

`-> jpegUtilLibRefNum`	Sony JpegUtil Library reference number.

Result Please refer to `JpegUtilLibErr`.

jpegUtilLibGetAPIVersion

Purpose Return the Sony JpegUtil Library version.

Prototype

```
UInt32 jpegUtilLibGetAPIVersion( UInt16 jpegUtilLibRefNum );
```

Parameters

`-> jpegUtilLibRefNum`	Sony JpegUtil Library reference number.

Result

`version`	Sony JpegUtil Library version.

Please refer to `JpegUtilLibErr`.

Utility API

jpegUtilLibDecodeImageToBmp

Purpose Decode JPEG data and return the results in bitmap format.

Prototype

```
Err jpegUtilLibDecodeImageToBmp( UInt16 jpegUtilLibRefNum,
FileRef fileRef, MemPtr inBufP, JpegImageType imageType,
JpegImageRatio ratio, BitmapPtr *bitmapPP,
PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
`-> fileRef`	File reference number of the JPEG file.
`-> inBufP`	Memory address where JPEG data is stored.
`-> imageType`	Image type (thumbnail or real image).
`-> ratio`	Image scaling factor.
`<-  bitmapPP`	Pointer to the Bitmap output by the Sony JpegUtil Library.

`<-> prgInfoP`	Indicates decoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

When `fileRef` is not 0, decodes JPEG data from the file specified by `fileRef`.

When `fileRef` is 0, decodes JPEG data from the memory area specified by `inBufP`.

[Output]

The Sony JpegUtil Library allocates a Bitmap and returns a pointer to it to the application. The application can use `BmpGetDimentions()` to acquire the Bitmap width and height and can use `WinDrawBitmap()`, etc. to display into the draw window. It is the application's responsibility to call `BmpDelete()` to release the memory when done.

When Orientation information exists in the Exif header, the orientation of the image is considered when output. (0 , 90 , 180 , -90 is supported)

- Because 16bpp is assumed for bitmaps, this is only supported with OS4.0 and later.
- The decoded results are held as a bitmap suitable for use. It is also possible to temporarily store received mail attachments in memory, and then use the API to decode from memory.

See Also `jpegUtilLibDecodeImageToBmpForFS`

jpegUtilLibDecodeImageToWindow

Purpose Decode JPEG data and then draw into the specified DrawWindow location.

Prototype

```
Err jpegUtilLibDecodeImageToWindow(
UInt16 jpegUtilLibRefNum, FileRef fileRef, MemPtr inBufP,
JpegImageType imageType, RectangleType *rP,
PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
`-> fileRef`	File reference number of the JPEG file.
`-> inBufP`	Memory address where JPEG data is stored.
`-> imageType`	Image type (thumbnail or real image).
`-> rP`	Decoded image rectangular display area.
`<-> prgInfoP`	Indicates decoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

When `fileRef` is not 0, decodes JPEG data from the file specified by `fileRef`.

When `fileRef` is 0, decodes JPEG data from the memory area specified by `inBufP`.

[Output]

Draws the specified JPEG data to the current draw window in the rectangle specified by `rP`. If the JPEG image is larger than this rectangle, the JpegUtil Library automatically scales down the input image size by the minimum amount necessary to fit the rectangle. Only linear reductions of 2, 4, or 8 times are supported. Images with larger dimensions more than 8 times those of `rP` will not be displayed. If the input image is smaller than `rP` after reduction, the surrounding area will be filled with black.

When Orientation information exists in the Exif header, the orientation of the image is considered when output. (0 , 90 , 180 , -90 is supported)

- Because 16bpp is assumed for Bitmaps, this is only supported with OS4.0 and later.
- This is suited for displaying image after image into a location on the display, without handling any bitmaps. It is also possible to temporarily store received mail attachments in memory, and then use the API to decode from memory.
- Care concerning the display’s resolution is required when specifying the rectangular area.

See Also `jpegUtilLibDecodeImageToWindowForFS`

jpegUtilLibEncodeImageFromBmp

Purpose Encode bitmap data into JPEG data.

Prototype

```
jpegUtilLibErr jpegUtilLibEncodeImageFromBmp(
UInt16 jpegUtilLibRefNum, Boolean isExif, Char *dateTime,
Char *softName, GPSInfoP gpsInfoP, UInt8 quality,
BitmapPtr bitmapP, FileRef fileRef, MemPtr *outBufPP,
UInt32 *outBufSizeP, PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
`-> isExif`	Whether or not to encode as Exif compliant JPEG.
`-> dateTime`	Date and time picture was taken. (e.g., 2001:11:08 13:00:00)
`-> softName`	Software name.
`-> gpsInfoP`	GPS information.
`-> quality`	Quality (1..100 : higher is better).
`-> bitmapP`	Starting address of image when encoding from bitmap.
`-> fileRef`	File reference number of the saved JPEG file.

<-	`outBufPP`	Memory address of JPEG output.
<-	`outBufSizeP`	Size of memory allocated for output.
<->	`prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Encodes the bitmap at the set `bitmapP`.
When specifying as Exif, can record information into the Exif header by specifying `dateTime`, `softName`, and `gpsInfoP`.
Use when wanting to save the information, such as date and time of picture or GPS information, acquired with `jpegUtilLibGetJpegInfo()`.

[Output]

When `fileRef` is not 0, outputs JPEG data into the set `fileRef`.

When `fileRef` is 0, the Sony JpegUtil Library allocates memory buffer and returns its start address in `outBufPP` and its size in `outBufSizeP`.
This memory buffer is maintained by the system and is freed when the library closes. Do not free this memory manually.

- Because 16bpp is assumed for Bitmaps, this is only supported with OS4.0 and later.
- Encodes a JPEG from bitmap data. Can also be used for temporary applications such as outputting to memory and attaching to mail, etc.
- If `fileRef` is specified, sets `outBufPP` and `outBufSizeP` to NULL.
- If the file specified by `fileRef` already exists, the file is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.
- If `dateTime` is not specified, the time of the API call is written.

See Also `jpegUtilLibEncodeImageFromBmpForFS`

jpegUtilLibEncodeImageFromWindow

Purpose Encode JPEG data from a rectangular area of the Display Window.

Prototype

```
Err jpegUtilLibEncodeImageFromWindow(
UInt16 jpegUtilLibRefNum, Boolean isExif, Char *dateTime,
Char *softName, GPSInfoP gpsInfoP, UInt8 quality,
RectangleType *rP, FileRef fileRef, MemPtr *outBufPP,
UInt32 *outBufSizeP, PrgInfoP prgInfoP );
```

Parameters

->	`jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
->	`isExif`	Whether or not to encode as Exif compliant JPEG.

`->`	`dateTime`	Date and time picture was taken. (e.g., 2001:11:08 13:00:00)
`->`	`softName`	Software name.
`->`	`gpsInfoP`	GPS information.
`->`	`quality`	Quality (1..100 : higher is better).
`->`	`rP`	Rectangular area of the Display Window.
`->`	`fileRef`	File reference number of the saved JPEG file.
`<-`	`outBufPP`	Memory address of JPEG output.
`<-`	`outBufSizeP`	Size of memory allocated for output.
`<->`	`prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Encodes a rectangular area of the Display Window specified by `rP` using the display window coordinate system.
When specifying as Exif, can record information into the Exif header by specifying `dateTime`, `softName`, and `gpsInfoP`.
Can save the information, such as date and time of picture or GPS information, acquired with `jpegUtilLibGetJpegInfo()`.

[Output]

When `fileRef` is not 0, outputs JPEG data into the set `fileRef`.

When `fileRef` is 0, the Sony Jpeg Library allocates a memory buffer and returns its start address in `outBufPP` and its size in `outBufSizeP`.
For memory allocation about 400KB StrageHeap is necessary. When memory cannot be allocated, JPEG data output to memory cannot be performed.
This memory buffer is maintained by the system, and is freed when the library closes. Do not free this memory manually.

- Because 16bpp is assumed for the rectangular area image, this is only supported with OS4.0 and later.
- Suited to clipping part of image shown in the display and encoding into JPEG. Can also be used for temporary applications such as outputting to memory and attaching to mail, etc.
- Care concerning the display's resolution is required when specifying the rectangular area.
- When `fileRef` is specified, sets `outBufPP` and `outBufSizeP` to NULL.
- If the file specified by `fileRef` already exists, the file is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.
- If `dateTime` is not specified, the time of the API call is written.

See Also `jpegUtilLibEncodeImageFromWindowForFS`

jpegUtilLibEncodeImageFromPGP

Purpose Encode JPEG data from a PGP format image database in the CLIÉ™.

Prototype

```
Err jpegUtilLibEncodeImageFromPGP( UInt16 jpegUtilLibRefNum,
Boolean isExif, Char *softName, GPSInfoP gpsInfoP,
UInt8 quality, DmOpenRef dRef, FileRef inFileRef,
FileRef outFileRef, MemPtr *outBufPP, UInt32 *outBufSizeP,
PrgInfoP prgInfoP );
```

Parameters

->	`jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
->	`isExif`	Whether or not to encode as Exif compliant JPEG.
->	`softName`	Software name.
->	`gpsInfoP`	GPS information.
->	`quality`	Quality (1..100 : higher is better).
->	`dRef`	Database reference number of a PGP database in the CLIÉ™.
->	`inFileRef`	File reference number of a PGP file in the MS.
->	`outFileRef`	File reference number of the saved JPEG file.
<-	`outBufPP`	Memory address of JPEG output.
<-	`outBufSizeP`	Size of memory allocated for output.
<->	`prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Converts a PGP stored in the database into JPEG.
When specifying as Exif, can record information into the Exif header by specifying `dateTime`, `softName`, and `gpsInfoP`.
Useful for saving the information, such as date and time of picture or GPS information, acquired with `jpegUtilLibGetJpegInfo()`.

The JPEG uses the date information recorded in the PGP database.

[Output]

When `fileRef` is not 0, outputs JPEG data into the set `fileRef`.

When `fileRef` is 0, the Sony Jpeg Library allocates a memory buffer and returns its start address in `outBufPP` and its size in `outBufSizeP`.
For memory allocation about 400KB StrageHeap is necessary. When memory cannot be allocated, JPEG data output to memory cannot be performed.
This memory is maintained by the system and is freed when the library closes. Do not free this memory manually.

- Because 16bpp is assumed for the image, this is only supported with OS4.0 and later.
- Suited to converting PGP data into JPEG.
- If `outFileRef` is specified, sets `outBufPP` and `outBufSizeP` to NULL.
- If the file specified by `outFileRef` already exists, the file is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.

See Also `jpegUtilLibEncodeImageFromPGPForFS`

jpegUtilLibGetJpegInfo

Purpose Retrieves information for a specified JPEG image.

Prototype

```
Err jpegUtilLibGetJpegInfo( UInt16 jpegUtilLibRefNum,
FileRef fileRef, MemPtr inBufP, UInt32 *imgHeightP,
UInt32 *imgWidthP, Boolean *isThumbnailP,
JpegDetailInfoP jpegDetailInfoP );
```

Parameters

->	`jpegUtilLibRefNum`	Sony JpegUtil Library reference number.
->	`fileRef`	File reference number of the JPEG file.
->	`inBufP`	Memory address where JPEG data is stored.
<-	`imgHeightP`	Height of JPEG image.
<-	`imgWidthP`	Width of JPEG image.
<-	`isThumbnailP`	Whether or not a thumbnail image is included in the JPEG data.
<-	`jpegInfoP`	Pointer to JPEG data information.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

When `fileRef` is not 0, acquires JPEG data information from the set `fileRef`.

When `fileRef` is 0, acquires JPEG data information from the memory area specified by `inBufP`.

- Some information can only be acquired if the JPEG image is compliant with the Exif specification.
- The Sony JpegUtil Library allocates memory for `softName`, `gpsInfoP`, and `gpsInfoP->mapDatumP`, but does not free it automatically. It is the application's responsibility to manage and free these memory buffers. If necessary, applications can use the following macro.

```
#define FreeJpegDetailInfo(jpegDetailInfoP) \
do {  \
  if((jpegDetailInfoP)->jpegDetailInfoCapability.softName) {  \
    MemPtrFree((jpegDetailInfoP)->softName);  \
  }  \
  if((jpegDetailInfoP)->jpegDetailInfoCapability.gpsInfo) {   \
    if((jpegDetailInfoP)->gpsInfoP->gpsInfoCapability.mapDatum) {  \
      MemPtrFree((jpegDetailInfoP)->gpsInfoP->mapDatum); \
    }  \
    MemPtrFree((jpegDetailInfoP)->gpsInfoP);  \
  }  \
} while (0)
```

See Also jpegUtilLibGetJpegInfoForFS

Utility API (Parts added from Ver.2.0)

jpegUtilLibDecodeImageToBmpForFS

Purpose Decode JPEG data and return the results as Bitmap format.

Prototype

```
Err jpegUtilLibDecodeImageToBmpForFS(
UInt16 jpegUtilLibRefNum, FileHand stream,
JpegImage TypeimageType, JpegImageRatio ratio,
BitmapPtr *bitmapPP, PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Reference number for JpegUtilLib
`-> stream`	FileHandle for FileStream format
`-> inBufP`	Memory address where JPEG data is stored
`-> imageType`	Image type (thumbnail or real image)
`-> ratio`	Image Scaling factor (1/1 or 1/2 or 1/4 or 1/8)
`<-  bitmapPP`	Pointer to Bitmap that JpegUtilLib will output
`<-> prgInfoP`	Indicates decoding progress

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

FileHandle of FileStream format database

[Output]

JpegUtilLib allocates a Bitmap area and returns a pointer to it to the application. The application can use `BmpGetDimentions()` to acquire the Bitmap width and height and can use `WinDrawBitmap()`, etc. to display into the draw window. Have the application call `BmpDelete() to release the memory.`

- Because 16bpp is assumed for bitmaps, this is only supported with OS4.0 and later.
- The decoded results are held as a bitmap suitable for use. It is also possible to temporarily store received mail attachments in memory, and then use the API to decode from memory.
- When Orientation information exists in the Exif header, the orientation of the image is considered when output. (0 , 90 , 180 , -90 is supported)

See Also `jpegUtilLibDecodeImageToBmp`

jpegUtilLibDecodeImageToWindowForFS

Purpose Decode JPEG data and then draw into the specified DrawWindow location.

Prototype

```
Err jpegUtilLibDecodeImageToWindowForFS(
UInt16 jpegUtilLibRefNum, FileHand stream,
JpegImageType imageType, RectangleType *rP,
PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Reference number for JpegUtilLib
`-> stream`	FileHandle of FileStream format
`-> inBufP`	Memory address where JPEG data is stored.
`-> imageType`	Image type (thumbnail or real image).
`-> rP`	Decoded image rectangular display area.
`<-> prgInfoP`	Indicates decoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

FileHandle of FileStream format database

[Output]

Draws the JPEG data into the rectangular area specified by rP, in the Window specified by `WinSetDrawWindow()`. Scaling and fitting into the rectangular area is performed by the Sony JpegUtil Library. (With Fitting the surrounding area will be filled with black)

- Because 16bpp is assumed for Bitmaps, this is only supported with OS4.0 and later.
- This is suited for displaying image after image into a location on the display, without handling any bitmaps. It is also possible to temporarily store received mail attachments in memory, and then use the API to decode from memory.

- Care concerning the display's resolution is required when specifying the rectangular area.
- When Orientation information exists in the Exif header, the orientation of the image is considered when output. (0 , 90 , 180 , -90 is supported)

See Also `jpegUtilLibDecodeImageToWindow`

jpegUtilLibEncodeImageFromBmpForFS

Purpose Encode bitmap data into JPEG data.

Prototype

```
Err jpegUtilLibEncodeImageFromBmpForFS(
UInt16 jpegUtilLibRefNum, Boolean isExif,
Char *dateTimeOriginal, Char *softName, GPSInfoP gpsInfoP,
UInt8 quality, BitmapPtr bitmapP, FileHand stream,
PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Reference number for JpegUtilLib
`-> isExif`	Whether or not to encode as Exif format JPEG.
`-> dateTimeOriginal`	Date and time picture was taken. (e.g., 2001:11:08 13:00:00)
`-> softName`	Software name.
`-> gpsInfoP`	GPS information.
`-> quality`	Quality (1..100).
`-> bitmapP`	Starting address of image when encoding from bitmap.
`-> stream`	FileHandle of FileStream format
`<-> prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Encodes the bitmap at the set `bitmapP`. When specifying as Exif, can record information into the Exif header by specifying `dateTimeOriginal`, `softName`, and `gpsInfoP`. Use when wanting to save the information, such as date and time of picture or GPS information, acquired with `jpegUtilLibGetJpegInfo()`.

[Output]

Outputs JPEG data into the FileStream format database.

- Because 16bpp is assumed for Bitmaps, this is only supported with OS4.0 and later.

- Suitable for usage when encoding a JPEG image from bitmap data. Can also be used for temporary applications such as outputting to memory and attaching to mail, etc.
- When the FileStream format database specified by stream already exists, it is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.
- If `dateTimeOriginal` is not specified, the time of the API call is written.

See Also `jpegUtilLibEncodeImageFromBmp`

jpegUtilLibEncodeImageFromWindowForFS

Purpose Encode JPEG data from a rectangular area of the Display Window.

Prototype

```
Err jpegUtilLibEncodeImageFromWindowForFS(
UInt16 jpegUtilLibRefNum, Boolean isExif,
Char *dateTimeOriginal, Char *softName, GPSInfoP gpsInfoP,
UInt8 quality, RectangleType *rP, FileHand stream,
PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Reference number for JpegUtilLib
`-> isExif`	Whether or not to encode as Exif compliant JPEG.
`-> dateTimeOriginal`	Date and time picture was taken. (e.g., 2001:11:08 13:00:00)
`-> softName`	Software name.
`-> gpsInfoP`	GPS information.
`-> quality`	Quality (1..100).
`-> rP`	Rectangular area of the Display Window.
`-> stream`	FileHandle of FileStream format.
`<-> prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Encodes a rectangular area of the Display Window specified by `rP` using the display window coordinate system.
When specifying as Exif, can record information into the Exif header by specifying `dateTimeOriginal`, `softName`, and `gpsInfoP`. Can save the information, such as date and time of picture or GPS information, acquired with `jpegUtilLibGetJpegInfo()`.

[Output]

Outputs JPEG data into the FileStream format database.

- Because 16bpp is assumed for the rectangular area image, this is only supported with OS4.0 and later.
- Suited to clipping part of image shown in the display and encoding into JPEG. Can also be used for temporary applications such as outputting to memory and attaching to mail, etc.
- Care concerning the display's resolution is required when specifying the rectangular area.
- When the FileStream format database specified by stream already exists, it is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.
- If `dateTimeOriginal` is not specified, the time of the API call is written.

See Also `jpegUtilLibEncodeImageFromWindow`

jpegUtilLibEncodeImageFromPGPForFS

Purpose Encode JPEG data from a PGP format image database in the CLIE.

Prototype

```
Err jpegUtilLibEncodeImageFromPGPForFS(
UInt16 jpegUtilLibRefNum, Boolean isExif, Char *softName,
GPSInfoP gpsInfoP, UInt8 quality, DmOpenRef dRef,
FileRef inFileRef, FileHand stream, PrgInfoP prgInfoP );
```

Parameters

`-> jpegUtilLibRefNum`	Refernece number for JpegUtilLib.
`-> isExif`	Whether or not to encode as Exif compliant JPEG.
`-> softName`	Software name.
`-> gpsInfoP`	GPS information
`-> quality`	Quality (1..100)
`-> dRef`	Database reference number of a PGP database in the CLIE.
`-> inFileRef`	File reference number of a PGP file in the MS.
`-> stream`	FileHandle of FileStream format.
`<-> prgInfoP`	Indicates encoding progress.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Convers PGP inside DB (specified with `dRef`)/MS (specified with `inFileRef`) into a JPEG. When specifying as Exif, can record information into the Exif header by specifying `softName` and `gpsInfoP`. As stated below, useful for saving GPS information acquired with `JpegUtilGetJpegInfo()`. The date and time information, `DateTimeOriginal`, recorded in the PGP database is reflected into the JPEG.

[Output]

Outputs JPEG data into the FileStream format database.

- Because 16bpp is assumed for the image, this is only supported with OS4.0 and later.
- Suited to converting PGP data into JPEG.
- When the FileStream format database specified by stream already exists, it is overwritten.
- `softName` and `gpsInfoP` are not stored in the Exif header if they are NULL.

See Also `jpegUtilLibEncodeImageFromPGP`

jpegUtilLibGetJpegInfoForFS

Purpose Retrieves information for a specified JPEG image.

Prototype

```
Err jpegUtilLibGetJpegInfoForFS( UInt16 jpegUtilLibRefNum,
FileHand stream, UInt32 *imgHeight, UInt32 *imgWidth,
Boolean *isThumbnail, JpegDetailInfoP jpegDetailInfoP );
```

Parameters

->	`jpegUtilLibRefNum`	Reference number for JpegUtilLib
->	`stream`	FileHandle of FileStream format
<-	`imgHeight`	Height of JPEG image.
<-	`imgWidth`	Width of JPEG image
<-	`isThumbnail`	Whether or not a thumbnail image is included in the JPEG data.
<-	`jpegInfoP`	Pointer to JPEG data information.

Result Please refer to `JpegUtilLibErr`.

Comments

[Input]

Acquires JPEG data information of FileStream format database.

- Some members `JpegInfoType` can only be acquired if JPEG is Exif compliant.
- Since JpegUtilLib allocates memory for `softName`, `gpsInfoP`, and `gpsInfoP->mapDatumP`, if the memory is not needed it is necessary for the application to free it. On occasions such as these, the following macro can be used.

```
#define FreeJpegDetailInfo(jpegDetailInfoP) \
{ \
     if((jpegDetailInfoP)->jpegDetailInfoCapability.softName) { \
          MemPtrFree((jpegDetailInfoP)->softName); \
     } \
     if((jpegDetailInfoP)->jpegDetailInfoCapability.gpsInfo) { \
          if((jpegDetailInfoP)->gpsInfoP->gpsInfoCapability.mapDatum) { \
       MemPtrFree((jpegDetailInfoP)->gpsInfoP->mapDatum); \
  } \
  MemPtrFree((jpegDetailInfoP)->gpsInfoP); \
     } \
}
```

See Also `jpegUtilLibGetJpegInfo`

jpegUtilLibSetJpegOrientation

Purpose Sets the orientation of a JPEG image.

Prototype

```
Err jpegUtilLibSetJpegOrientation (
UInt16 jpegUtilLibRefNum, FileRef fileRef, MemPtr inBufP,
JpegImageOrientation orientation );
```

Parameters

-> `jpegUtilLibRefNum`	Reference number for the Sony JpegUtil Library.
-> `fileRef`	File reference number of JPEG file
-> `inBufP`	Memory address where JPEG data is stored
-> `orientation`	JPEG image orientation to set

Result `jpegUtilLibErrExifParamNotFound`

When Exif format but `Orientation` tag does not exist, this error is returned.

Please refer to `JpegUtilLibErr`.

Comments **[Input]**

When `fileRef` is not 0, sets the image orientation of the set `fileRef`'s JPEG data. When `fileRef` is 0, sets the image orientation of the JPEG data in the memory area specified by `inBufP`.

- Sets both the main image and the thumbnail image to the same orientation.
- When there is no Orientation tag in the Exif for both the main image and the thumbnail, an error is returned. Determining if Orientation tags exist for both can

be accomplished by acquiring `jpegUtilLibGetJpegInfo` and examining the capability information.

Notes

Usage example

The following shows an usage example of the Sony JpegUtil Library.

For simplicity, error handling is omitted.

```
/* Function to decode a JPEG image in the Memory Stick to a Bitmap */
#include <SonyCLIE.h>

void jpegDecodeToBmp()
{
  Err err;
  UInt32 volIterator = vfsIteratorStart;
  UInt16 jpegUtilLibRefNum;
  UInt16 HRrefNum;
  UInt16 volRefNum;
  FileRef fileRef;
  JpegImageType imageType = jpegDecModeNormal; // Specify the main image
  JpegImageRatio ratio = jpegDecRatioNormal; // ratio
  BitmapPtr bitmapP;
  Coord width, height;

  // Find Hireso Library
  err = SysLibFind(sonySysLibNameHR, &HRrefNum);
  if (err) {
    // Load Hireso Library
    SysLibLoad( sonySysFileTHRLib, sonySysFileCHRLib, &HRrefNum );
    if (err) {
      return;
    }
  }

  err = HROpen(HRrefNum);
  if(err) {
    return;
  }

  // Find Sony JpegUtil Library

  err = SysLibFind(sonySysLibNameJpegUtil, &jpegUtilLibRefNum);
  if (err) {
    // Load Sony JpegUtil Library
    err = SysLibLoad( sonySysFileTJpegUtilLib, sonySysFileCJpegUtilLib,
                      & jpegUtilLibRefNum );
    if (err) {
      return;
    }
  }

  // Open Sony JpegUtil Library
  err =jpegUtilLibOpen(jpegUtilLibRefNum);
  if (err) {
    return;
  }

  // Get volRefNum
  err = VFSVolumeEnumerate(&volRefNum, &volIterator);
  if (err) {
    return;
  }

  // Open the jpeg file you want to decode
  err = VFSFileOpen( volRefNum, "/DCIM/100MSDCF/DSC00001.JPG", vfsModeRead,
                      &fileRef );

  // Call Utility API
  err = jpegUtilLibDecodeImageToBmp( jpegUtilLibRefNum, fileRef, NULL,
                      imageType, ratio, &bitmapP, NULL );

  // Close Sony JpegUtil Library
  jpegUtilLibClose(jpegUtilLibRefNum);

  if (!err) { // Display the decoded image
    BmpGetDimensions(bitmapP, &width, &height, NULL);

    // Draw bitmap
    HRWinDrawBitmap( HRrefNum, bitmapP, rec.topLeft.x, rec.topLeft.y );
    // Delete bitmap
    BmpDelete(bitmapP);
  }

  // Close File
  VFSFileClose(fileRef);

  HRClose(HRrefNum);
}
```

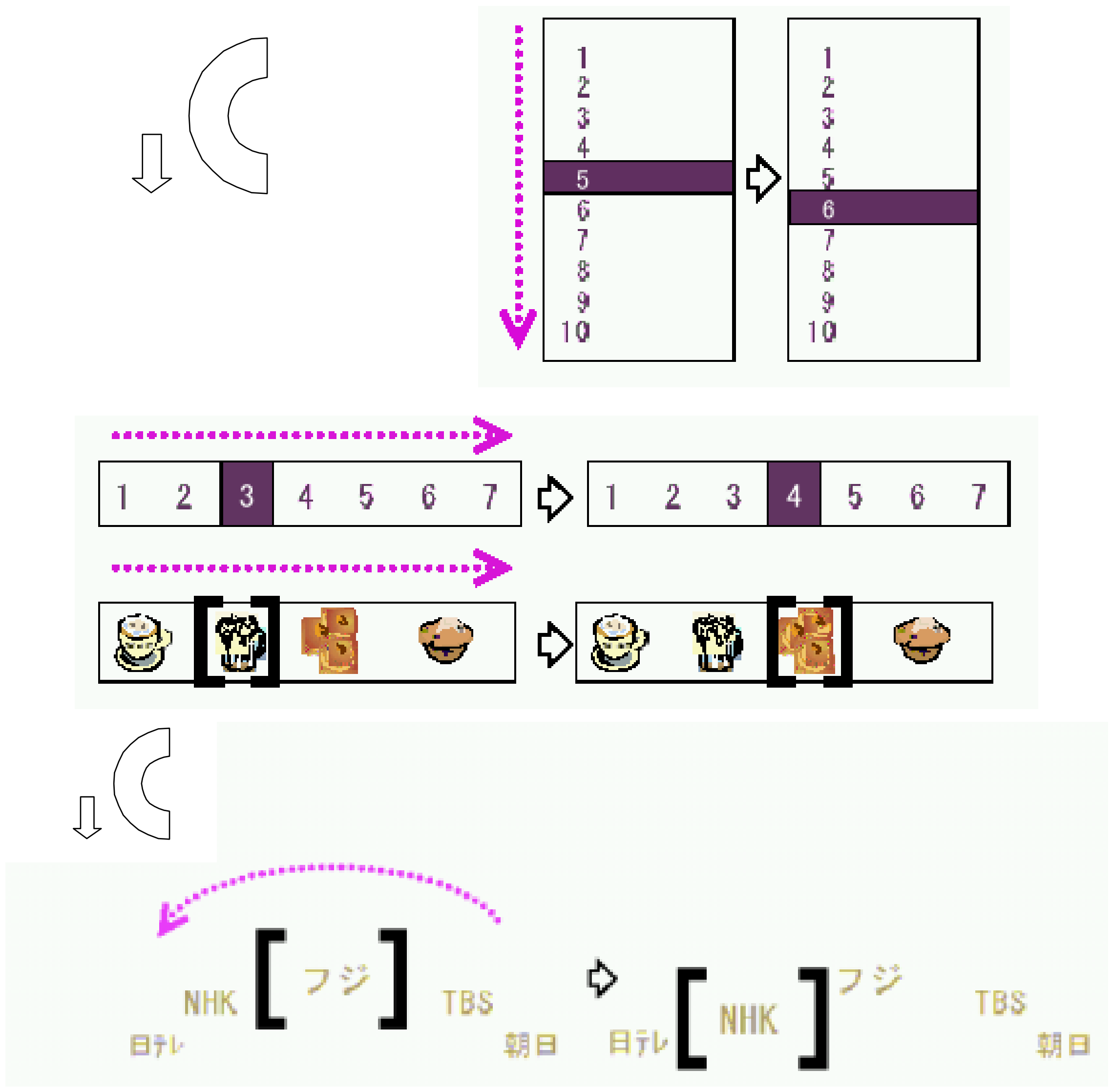
A

Jog Dial navigator User Interface Guideline

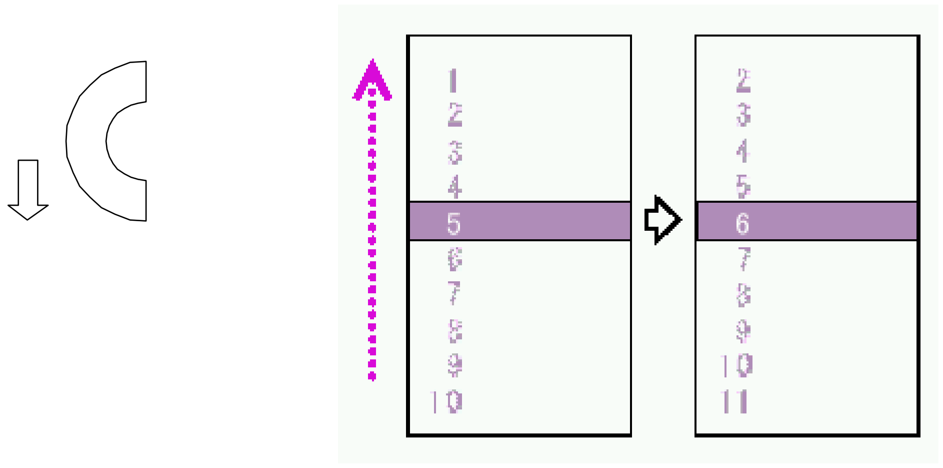
This is a guideline for developers who want to use the Jog Dial navigator in their applications. Users should expect the Jog Dial navigator to influence programs in similar ways. By following these guidelines, developers can ensure that their application's user interface responds to the Jog Dial navigator appropriately.

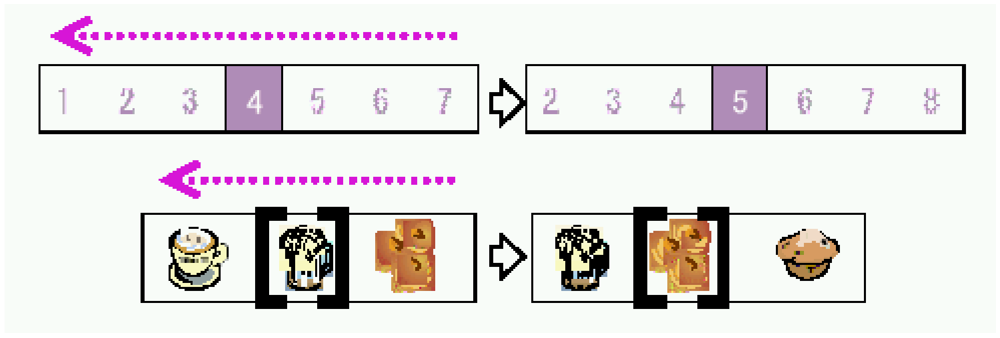
- When continuing to press the Jog Dial navigator and then releasing, `vchrJogPush` is executed with the initial first press and `vchrJogRelease` is executed upon release. Unless the application is a kind of launcher, both actions are basically considered as an Enter function, however it is recommended 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`

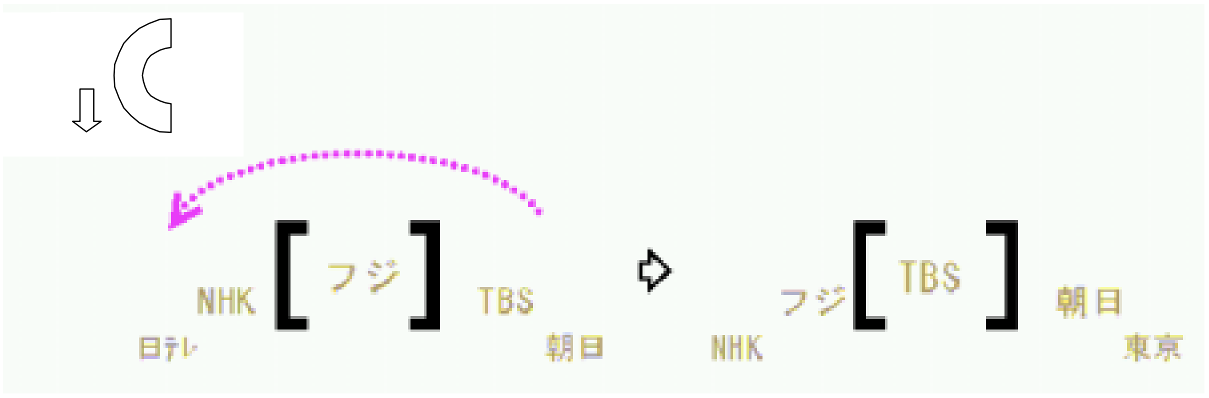
call. In the case of a circular list, the cursor should be moved in the same direction as the Jog Dial navigator while the list/wheel holds its position.



- 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.







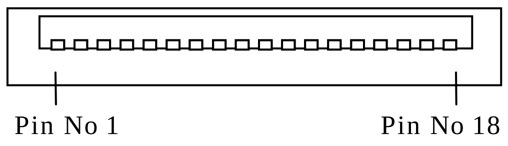
B

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.

Interface Connector

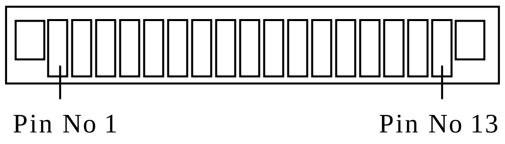
Pin Specification(PEG-NZ, NX, TH, TG, TJ Series)



Pin No	Name	In brief
1	GND	Ground for Signal, Power
2	USB D+	USB Data+
3	USB D-	USB Data-
4	USB_GND	Ground for USB
5	VBUS	VBUS for USB
6	Reserved	
7	DC+B	Power terminal post
8	CHARGE	Charge
9	Reserved	
10	UNREG_OUT	Power Supply

Pin No	Name	In brief
11	HOT_SYNC	HotSync
12	DTR	UART(Data Terminal Ready)
13	RXD	UART(Receive Data)
14	TXD	UART(Transmit Data)
15	CTS	UART(Clear to Send)
16	RTS	UART(Request to Send)
17	CNT	Accessory detection
18	GND	Ground for Signal, Power

Pin Specification(PEG-S, PEG-N Series)

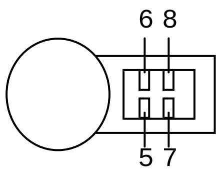


Pin No	Name	In brief
1	USB D-	USB Data-
2	USB D+	USB Data+
3	DTR	Data Terminal Ready
4	RXD	Receive Data
5	RTS	Request to Send
6	TXD	Transmit Data
7	CTS	Clear to Send
8	NC	-
9	DC_B+	Power terminal post
10	HOT SYNC	Hot Sync
11	UNREG OUT	Power supply

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)

C

CF Memory Card

Some CLIÉ™ Handheld devices support CF memory cards. This chapter describes how to use CF memory cards and Memory Stick® media on CLIÉ Handheld devices effectively.

Using VFS Manager

Volume reference numbers and volume information of expansion cards can be obtained with appropriate VFS Manager API calls or in VFS notification events. To identify the expansion card type, use the media type field in the volume information structure.

Please refer to the following sample code for details.

Listing 1 Using VFS Manager API

```
static Err AppStart(void)
{
  UInt32 volIterator = vfsIteratorStart;
  UInt16 volRefNum;
  VolumeInfoType volInfo;
  Err err;

  while(volIterator != vfsIteratorStop) {
    /* obtain Volume Reference Number */
    err = VFSVolumeEnumerate(&volRefNum, &volIterator);
    if(err) {
      FrmCustomAlert(ErrOKAlert, "VFSVolumeEnumerate failed", "", "");
      break;
    }

    /* obtain Volume Information */
    err = VFSVolumeInfo(volRefNum, &volInfo);
    if(err) {
      FrmCustomAlert(ErrOKAlert, "VFSVolumeInfo failed", "", "");
      break;
    }

    /* identify Media Type */
    switch(volInfo.mediaType) {
      case expMediaType_MemoryStick:
        /* Process for Memory Stick media */
        break;
      case expMediaType_CompactFlash:
        /* Process for CF memory card */
        break;
      default:
        break;
    }
  }
}
```

Listing 2 Using notifications (`sysNotifyVolumeMountedEvent`)

```
static Err PrvGetMediaTypeFromVolRefNum(UInt16 volRefNum, UInt32*mediaTypeP)
{
  VolumeInfoType volInfo;
  Err err;

  /* obtain Volume Information */
  err = VFSVolumeInfo(volRefNum, &volInfo);
  if(err == errNone) {
    /* return media type */
    *mediaTypeP = volInfo.mediaType;
  }
  return err;
}

static Err PrvVolumeMountedHandler( SysNotifyParamType *notifyParamsP )
{
  /* NOTE: applications should avoid using globals in notification callbacks
   */

  VFSAnyMountParamType *vfsMountParamP =
    (VFSAnyMountParamType*) notifyParamsP->notifyDetailsP;
  UInt32 mediaType;
  Err err;

  err = PrvGetMediaTypeFromVolRefNum(vfsMountParamP->volRefNum, &mediaType);
  if(err == errNone) {
    switch(mediaType) {
      case expMediaType_MemoryStick:
        /* Process for Memory Stick media */
        break;
      case expMediaType_CompactFlash:
            /* Process for CF memory card */
            break;
         default:
            break;
      }
   }
   return errNone;
}

static Err AppStart(void)
{
   UInt16 cardNo;
   LocalID dbID;

   SysCurAppDatabase(&cardNo, &dbID);
   SysNotifyRegister(cardNo, dbID, sysNotifyVolumeMountedEvent,
      PrvVolumeMountedHandler, sysNotifyNormalPriority, NULL);
   return errNone;
}

static void AppStop(void)
{
   UInt16 cardNo;
   LocalID dbID;

   SysCurAppDatabase(&cardNo, &dbID);
   SysNotifyUnregister(cardNo, dbID, sysNotifyVolumeMountedEvent,
      sysNotifyNormalpriority);
}
```

Notes

Usage notes

- VFS Manager is a feature of Palm OS®. For more details, please refer to the Palm OS SDK supplied by PalmSource, Inc.

- Please refer to CLIÉ customer support website for information about CF memory card compatibility on CLIÉ Handheld devices.

  Customers in USA, Canada:

  `http://www.ita.sel.sony.com/support/clie/`

  Customers in European Countries:

  `http://www.clie-link.com`

  Customers in Asia-Pacific Countries and Mexico:

  `http://vaio-online.sony.com/clie`

- Capabilities such as the read-and-write speeds of CF memory cards vary for each product. This SDK makes no guarantees about the performance of memory cards.

Index

A

Application Drawing Area 99

C

Compatibility Mode 43

E

Existing API 44
Existing Applications 44
Extended Drawing Area 99

F

Feature number 13

G

GPSInfoCapabilityType 134
GPSInfoType 134

H

High Density API 44
High Resolution API 44
High resolution mode 44
HRBmpBitsSize 86
HRBmpCreate 86
HRBmpSize 86
HRClose 64
HRFntAverageCharWidth 90
HRFntBaseLine 90
HRFntCharHeight 90
HRFntCharsInWidth 91
HRFntCharsWidth 91
HRFntCharWidth 90
HRFntDescenderHeight 91
HRFntGetFontSize 87
HRFntGetScrollValues 92
HRFntLineHeight 90
HRFntLineWidth 92
HRFntSetFont 87
HRFntWCharWidth 91
HRFntWidthToOffset 91
HRFntWordWrap 92
HRFntWordWrapReverseNLines 92
HRFontSelect 88
HRGetAPIVersion 64
HROpen 64
HRWinClipRectangle 65
HRWinCopyRectangle 65
HRWinCreateBitmapWindow 66
HRWinCreateOffscreenWindow 66
HRWinCreateWindow 67
HRWinDisplayToWindowPt 67
HRWinDrawBitmap 68
HRWinDrawChar 68
HRWinDrawChars 68
HRWinDrawGrayLine 69
HRWinDrawGrayRectangleFrame 69
HRWinDrawInvertedChars 70
HRWinDrawLine 70
HRWinDrawPixel 70
HRWinDrawRectangle 71
HRWinDrawRectangleFrame 71
HRWinDrawTruncChars 71
HRWinEraseChars 72
HRWinEraseLine 72
HRWinErasePixel 72
HRWinEraseRectangle 73
HRWinEraseRectangleFrame 73
HRWinFillLine 73
HRWinFillRectangle 74
HRWinGetClip 74
HRWinGetDisplayExtent 74
HRWinGetFramesRectangle 75
HRWinGetPixel 75
HRWinGetPixelRGB 75
HRWinGetWindowBounds 76
HRWinGetWindowExtent 76
HRWinGetWindowFrameRect 76
HRWinInvertChars 77
HRWinInvertLine 77
HRWinInvertPixel 77
HRWinInvertRectangle 78
HRWinInvertRectangleFrame 78
HRWinPaintBitmap 78
HRWinPaintChar 79
HRWinPaintChars 79
HRWinPaintLine 79
HRWinPaintLines 80
HRWinPaintPixel 80
HRWinPaintPixels 80
HRWinPaintRectangle 81

HRWinPaintRectangleFrame 81
HRWinRestoreBits 81
HRWinSaveBits 82
HRWinScreenMode 82
HRWinScrollRectangle 84
HRWinSetClip 85
HRWinSetWindowBounds 85
HRWinWindowToDisplayPt 85

J

JpegDetailInfoCapabilityType 132
JpegDetailInfoType 133
JpegImageRatio 132
JpegImageType 131
JpegPrgCallbackFunc 135
jpegUtilLibClose 137
jpegUtilLibDecodeImageToBmp 137
jpegUtilLibDecodeImageToBmpForFS 144
jpegUtilLibDecodeImageToWindow 138
jpegUtilLibDecodeImageToWindowForFS 145
jpegUtilLibEncodeImageFromBmp 139
jpegUtilLibEncodeImageFromBmpForFS 146
jpegUtilLibEncodeImageFromPGP 142
jpegUtilLibEncodeImageFromPGPForFS 148
jpegUtilLibEncodeImageFromWindow 140
jpegUtilLibEncodeImageFromWindowForFS 147
JpegUtilLibErr 131
jpegUtilLibGetAPIVersion 137
jpegUtilLibGetJpegInfo 143
jpegUtilLibGetJpegInfoForFS 149
jpegUtilLibOpen 136
jpegUtilLibSetJpegOrientation 150

M

Maximizing 100
Minimizing 100

P

PrgInfoType 135

R

RationalType 133
RmcDisableKeyHandler 123
RmcEnableKeyHandler 124
RmcGetStatus 124
RmcKeyCodeEnum 121
RmcKeyHandleProcPtr 125
RmcKeyRates 124
RmcLibClose 122
RmcLibOpen 122
RmcRegEnum 120
RmcRegister 122
RmcStatusType 121

S

Silk Plug-in 100
SilkLibClose 117
SilkLibDisableResize 118
SilkLibEnableResize 118
SilkLibGetAPIVersion 119
SilkLibOpen 117
SilkLibResizeDispWin 119
Silkscreen Area 100
Soft Graffiti 100
sonySysFtrNumJogAstControlP 23
sonySysFtrNumJogAstMaskP 23
sonySysFtrNumJogAstMOCardNoP 23
sonySysFtrNumJogAstMODbIDP 24
sonySysFtrNumSysInfoP 20
Standard Input Plug-in 100
Status Bar Area 100
sysNotifyDisplayChangeEvent 109
sysNotifyPalmSilkChangeEvent 111

V

vchrJogBack 16
vchrJogBack Assist 25
vchrJogDown 15
vchrJogLeft 19
vchrJogPush 15
vchrJogPush/PushRepeat/Release Assist 28
vchrJogPushedDown 16
vchrJogPushedUp 15
vchrJogPushedUp/PushedDown Assist 28
vchrJogPushRepeat 15
vchrJogRelease 15
vchrJogRight 19
vchrJogUp 15
vchrJogUp/Down Assist 26
vchrRmcKeyPush 35
vchrRmcKeyRelease 35
Virtual Silkscreen 100

VskClose 112
VskDoCommand 116
VskGetAPIVersion 112
VskGetCurrentSlkw 113
VskGetPalmSilkEnabled 115
VskGetState 113
VskOpen 111
VskSetCurrentSlkw 112
VskSetState 114

W
Wide-Resolution Screen 99