

Developer's Guide



Programming RIM Palm-Sized
Wireless Handhelds

Version 2.0



BlackBerry – Programming RIM Palm-Sized Wireless Handhelds , Version 2.0
Last revised 4/27/00

Part Number: PDF-02764-001

© 2000 RESEARCH IN MOTION LIMITED.

RIM, Research In Motion, and the RIM logo are trademarks of Research In Motion Limited. Research In Motion and RIM are registered, U.S. Patent and Trademark Office.

Complies with software O/S version 2.0 and applications version 2.0.

Windows is a trademark of Microsoft Corporation. Intel is a registered trademark of Intel Corporation. All other brands, product names, company names, trademarks, and service marks mentioned herein are registered trademarks or trademarks of their respective holders.

Warning: This document is for the use of licensed users only. Any unauthorized copying, distribution or disclosure of information is a violation of copyright laws.

While every effort has been made to ensure technical accuracy, information in this document is subject to change without notice and does not represent a commitment on the part of Research In Motion Limited.

Research In Motion Limited

295 Phillip Street
Waterloo, Ontario
Canada N2L 3W8
Tel. (519) 888-7465
Fax (519) 888-6906
Web site: www.rim.net
Email: info@rim.net

Printed in Canada

KM0200/proton info.doc

Contents

- 1. Introduction..... 5**
 - About this guide 5
- 2. Programming features 7**
 - Using the Simulator and OS..... 7
 - Writing portable code 8
- 3. Additional API functions 11**
- Index of API functions 15**
- Index 17**

1

Introduction

Version 2.0 of the BlackBerry SDK also supports the new RIM 957 Wireless Handheld. The RIM 957 features a palm-sized form factor, a large 160x160 screen, rechargeable battery, 5Mb of flash memory, and RIM's renowned "always on, always connected" wireless technology and QWERTY keyboard design.

From a programming standpoint, the palm-sized wireless handheld supports a superset of the features provided by the pager-sized BlackBerry handhelds. The larger handhelds sport an enhanced interface, including a notification LED, two shift keys and an escape key.

This document describes some of the additional information needed to develop applications for the palm-sized Wireless Handheld. It complements the existing BlackBerry Operating System API Developer's Guide.

About this guide

The SDK includes Developer's Guides to assist you in creating your applications. This guide includes general information on the features that are specific to the RIM 957 Wireless Handheld.

Throughout this guide, the simulator will be referred to as the 'simulator.'

Note Notes provide additional information to help complete a task.

Tip Tips offer an alternative method of performing an action

WARNING

Warnings follow any procedure or paragraph containing instructions that, if followed improperly, could result in damaging the handheld or software.

The names of program groups, icons, folders, windows, etc. will appear in **bold** text. An example would be the **Research In Motion** program group.

Menu options, menu names, fields, tabs, checkboxes etc. will appear in Arial font. References to buttons will also appear in this fashion, with the hot key underlined where applicable. An example would be the Next> button.

All programming elements such as returns, events, constants, structures, parameters, etc. will appear in Courier New font. An example would be the `COMM_RX_ERROR` error code.

2

Programming features

Because the RIM 957 Wireless Handheld is a BlackBerry, all of the code used for other BlackBerry Wireless Handhelds will still work. The 957 has a few additional APIs (described in more detail in the next chapter).

This chapter discusses additional features of RIM's palm-sized wireless handhelds, how to simulate them, and how to code for or around them.

Using the Simulator and OS

The `OSLOADER.EXE` executable starts the simulator. The first argument to the command is the name of the OS DLL file. For simulating a pager-sized wireless handheld, the OS DLL is `OSPGRMB.DLL` (as described in the SDK User's Guide).

To simulate RIM 957, a palm-sized handheld, use the `OSHHMB.DLL` file as an argument to the `OSLOADER.EXE` executable.

After setting up the project, go to Project -> Settings and select the Debug tab.

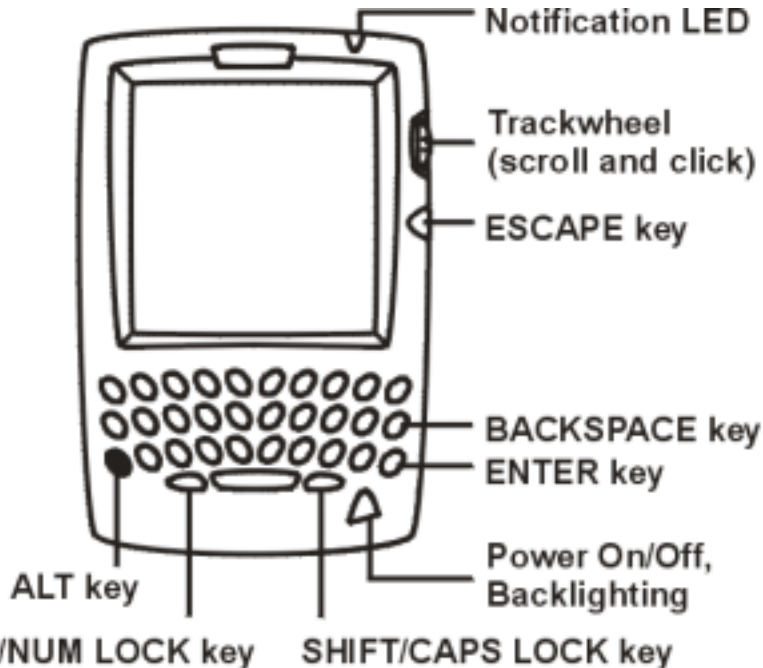
- Set the executable for the Debug Session as `OSLOADER.EXE`.
- In the Program arguments field put a line in the form:
`[OS DLL] [OS parameters] [application DLLs]`
`[OS DLL]` should be `OSPHHMB.DLL` if you are simulating a palm-sized RIM wireless handheld. The `[OS parameters]` are any command line switches you need to pass to the simulator.
- List all of the application DLLs at the end of the line, making sure your application is the last one listed.

All other settings for the project are the same as described in the SDK User's Guide.

Writing portable code

RIM strongly recommends that you write your applications so they can run on either the 950 or the 957. To take full advantage of the additional APIs, you must be able to identify the handheld device at runtime.

The RIM 957 Wireless Handheld has a different user interface than the RIM 950. This diagram shows the additional features in the display and interface:



New APIs are provided for controlling the Notification LED and for a new battery storage mode. Additional `SubMSG` values are provided for the additional keys.

Identifying a palm-sized wireless handheld

To write portable code that also takes advantage of the additional APIs of the palm-sized wireless handheld, your code must be able to distinguish between handheld devices at runtime.

Information about the type of the device is stored in the `DEVICE_INFO` structure (defined in `<RIM.H>`). Query the `RimGetDeviceInfo(...)` function to fill a `DEVICE_INFO` structure.

Possible values for the `deviceType` member are:

Value	Meaning
<code>DEV_PAGER</code>	A small RIM pager-sized device, typically a RIM 950.
<code>DEV_OEM</code>	A RIM OEM modem, such as the R902M.
<code>DEV_HANDHELD</code>	A RIM palm-sized wireless handheld, such as the RIM 957.

(The `DEVICE_INFO.networkType` member can be used to distinguish between 850 and 950-style BlackBerry wireless handhelds.)

Resolving unknown OS APIs

In order to allow your code to include references to operating system APIs that exist on only one model of wireless handheld, the linking behavior has changed slightly.

If your application calls an operating system API that doesn't exist, the linker resolves it to an internal call. Your code will still compile and link, but when that API is invoked, the application will display an error message.

Programming the display

The display for the palm-sized wireless handheld is significantly larger than the display for the pager-sized wireless handheld. This difference is hidden beneath the abstraction of the UI Engine.

Display code written using the UI Engine is portable between pager-sized and palm-sized wireless handhelds.

If your application requires low-level LCD access, you need to be concerned about the differences in screen real estate.

Information about the display can be determined by querying `LcdGetConfig(...)`. The width and height of the screen are stored in the `LcdConfig` structure (defined in `<LSC_API.H>`).

Interpreting additional SubMsgs

The keyboard of the palm-sized wireless handheld has more keys. This means that additional `SubMsg` values can occur for `KEY_DOWN`, `KEY_REPEAT`, and `KEY_UP` events:

Information about shift and alt status is still stored in the `Data[0]` field of the , but there are additional constants to determine which key was pressed: `SHIFT_STATUS_L` (the left Shift key) and `SHIFT_STATUS_R` (the right Shift key).

For example, if the left shift key is pressed down, the value stored in `Data[0]` is `0x22`: both the `SHIFT_STATUS` and `SHIFT_STATUS_L` constants are represented.

There are also new key names to represent the additional keys, as shown in the following table.

Constant	Value	Meaning
<code>KEY_ESCAPE</code>	<code>0x001b</code>	The escape key.
<code>KEY_SHIFT</code>	<code>0x0100</code>	On the palm-sized handhelds, this key refers to the right Shift key.
<code>KEY_SHIFT2</code>	<code>0x0102</code>	The left Shift key.
<code>KEY_BKLITE</code>	<code>0x0103</code>	The backlight key.

The constants are defined in `<KEYPAD.H>`.

3

Additional API functions

The following is a list of additional API functions which are available when developing an application for a RIM palm-sized wireless handheld:

RimConfigureLEDs.....	12
RimRequestStorageMode.....	13
RimSetLed	12

RimConfigureLEDs

Configure the way LEDs light.

```
void RimConfigureLEDs( int LED_On_Time,  
                      int LED_Off_Time, int DutyCycle )
```

Parameters:

LED_On_Time

Length of time the LED should remain on in milliseconds (will be rounded to nearest 8 ms)

LED_Off_Time

Length of time the LED should remain off in milliseconds (will be rounded to nearest 64 ms)

DutyCycle

Set this parameter to one of LED_DUTY_12 for 12%, LED_DUTY_25 for 25%, LED_DUTY_50 for 50%, or LED_DUTY_100 for 100%. During the period of LED_On_Time, the LED will light for the given percentage of time during each 8 ms period. Thus, DutyCycle sets the brightness of the LED during the LED_On_Time.

Return Value:

No return value

RimSetLed

Set the state of a LED.

```
void RimSetLed( int LedNumber, int LedState )
```

Parameters:

LedNumber

One of LED_COVERAGE for the coverage LED or LED_MESSAGE for the message LED.

LedState

One of LED_OFF, LED_ON, or LED_BLINK.

Return Value:

No return value

RimRequestStorageMode

Puts the handheld into a storage state to preserve the battery.

```
BOOL RimRequestStorageMode( void )
```

Return Value:

No return value

Remarks:

This function electronically disconnects the Lithium battery to prevent it from draining over a long period of inactivity. Note that the real time clock will likely drift during the time the handheld is in this state. The only way to power up the device after this call is to press the reset button on the back of the device or place it in a charging cradle.

Index of API functions

RimConfigureLEDs, 12
RimRequestStorageMode, 13
RimSetLed, 12

Index

- about this guide, 5
- additional functions, 11
- configuring LEDs, 12
- display, 9
- low-level LCD access, 9
- programming hints, 7
- requesting storage mode, 13
- RIM 957 Wireless Handheld, 5
- RIM palm-sized wireless handheld
 - identifying device, 8
 - identifying network type, 9
 - programming, 7
- RIM palm-sized wireless handheld information, 5
- screen width and height, 9
- setting LED, 12
- shift keys, 10
- SubMsg values, 10
- UI engine, 9