Software Developer's Reference

for the

Palm® and Stowaway™ Portable Keyboards

Palm OS® Platform

Revision 1.10

Table of Contents

I. Introduction

Think Outside

Think Outside is the inventor of the Stowaway Portable Keyboard, the world's first 100% full-size keyboard that folds to pocket-size. Versions of our keyboard that support Palm handhelds are distributed by Palm, Inc. under the name ***Palm Portable Keyboard***. Keyboards that support Handspring Visor handhelds, the HP Jornada 540 series, and the Compaq iPaq PocketPC are distributed by Targus Group International under the name ***Targus Stowaway Portable Keyboard***.

Stowaway-Friendly Software for Palm OS

This document is a reference for software developers who wish to write ***Stowaway-Friendly*** software for the ***Palm OS*** platform. In the simplest sense, a Stowaway-Friendly application is software designed to work well with our keyboard. Users may navigate and select menus, buttons, spreadsheet cells, etc. with the keys, and they should not have to interrupt typing to pick up the stylus.

Our larger motivation behind our "Stowaway-Friendly" campaign is to broaden the appeal of portable devices. While we can provide a good keyboard, and developers can provide great software, it's clear to us that the value of the software and keyboard ***together*** would be worth dramatically more if they worked well with each other. For this reason, we created the Stowaway Developer's Kit, including this reference for software developers.

In this document, we cover the built-in features of Think Outside's Palm OS driver and guidelines for keyboard-enhanced Palm applications. We also provide some sample code that demonstrates how to implement basic keyboard navigation and selection. If you're interested in building a connector or writing your own software driver, please take a look at the Hardware and Electronics Reference, also included in the Developer's Kit.

<u>Where is the latest Developer Support information?</u>
The latest developer information is available on our web site:

> ***http://www.thinkoutside.com/devsupport/***

When you sign up, you may download the latest developer materials. You may also specify that you wish to receive email notice when the materials are updated.

Questions and suggestions about developer support may be sent to:

> ***devsupport@thinkoutside.com***.

<u>How can I contact Think Outside?</u>
Think Outside is headquartered in Carlsbad, California, about 30 miles north of downtown San Diego.

> **Think Outside**
> **5790 Fleet Street Suite 130**
> **Carlsbad, California 92008**
> **(760) 431-9090**
> ***www.thinkoutside.com***

We welcome questions, comments, and other feedback regarding our developer support. Please send email to ***devsupport@thinkoutside.com***.

II. Document Overview

This document is organized into four sections:

- **Keyboard Driver for Palm OS Platform** - covers the behavior of the keyboard driver written by Think Outside.
- **Basic Keyboard Functionality** - reference for the functionality you automatically get when you install the Think Outside driver.
- **International and Custom Layouts** - reference for creating international and custom keyboard layouts, compatible with Think Outside's Palm OS driver versions 1.5 and above. The Sample Layout project, included in the Developer's Kit, will allow you to build custom keyboard layouts.
- **Enhanced Keyboard Functionality** - covers aspects of keyboard navigation that require some software implementation in your applications. The Keyboard Demo application, included in the Developer's Kit, gives some basic examples on how to implement enhanced keyboard navigation.

III. Document Conventions

The keyboards supported in this document are sold as the *Palm Portable Keyboard*, and the *Targus Stowaway Portable Keyboard for Handspring Visor*. In this document, we will refer to either keyboard as the **Stowaway Portable Keyboard**.

Some formatting conventions:

- The names of variables are set in *italics*.
- Function, type, class, and method names are in **`bolded courier`**.
- Form names within the user interface are ***bolded italics***.
- User interface object names are **bolded**.
- Table headings are **bolded and underlined**.
- Lines of code, file and directory names, values for variables and numerical values, e.g. `43`, `12.5`, etc. are set to `courier`.
- For variables that have only a limited number of values, the values of these variables are in *`italicized courier`*.
- String values for variables are in `"quoted courier"`.
- References to other sections in the document are in Arial font.
- Keyboard keys are shown in `CAPITALIZED COURIER`, e.g. `ENTER`

IV. Keyboard Driver for Palm OS Platform

1. Traps

The stowaway driver for the Palm OS Platform is an application that traps three system event handlers:

- *`sysTrapEvtGetEvent`*
- *`sysTrapFldHandleEvent`*
- *`sysTrapDmDeleteDatabase`*

Of these, we will discuss *`sysTrapEvtGetEvent`* in detail, as it is the most significant.

On pressing a key, the driver converts the keyboard scan code into an appropriate key event or series of pen events. The events are then pumped into the application's event queue during the processing of the *`sysTrapEvtGetEvent`* callback function.

Once the traps are installed, the keyboard driver waits for one of two events depending on the platform the driver is installed on.

For **Handspring Visor** handhelds, the driver waits for a power up. (You will notice that you need to power cycle the Visor handhelds once you connect a Stowaway keyboard.) When the driver detects the power up signal, it opens the serial port and looks for the keyboard. If it finds the keyboard, then the serial port stays open until the power is turned off, or until the keyboard is detached from the Visor. The driver checks for a detached keyboard every 30 seconds.

For **Palm** handhelds, the driver uses a HotSync signal to detect when the keyboard is present. Connecting the Palm handheld to the keyboard triggers this signal. In this way, the keyboard driver can stay in a "sleep" state until the HotSync signal is detected. If the driver sees a HotSync, it opens the serial port and checks for the keyboard. If a keyboard is found, the serial port remains open and the driver eats the HotSync event. If the keyboard is not found, the serial port is closed and the HotSync is passed onto the next handler in the *`sysTrapEvtGetEvent`* chain.

To save power on the Palm device, the serial port is closed every 3 seconds. Closing the serial port allows the Palm device to have access to the port for beaming and modem operations. Once the port is closed, the keyboard can be redetected through the HotSync, as described above, once the user presses a key on the Stowaway.

2. Key Events

Whenever a key is pressed on the keyboard, the driver pumps an event into the application through the use of either the **`EvtEnqueueKey`** or **`EvtAddEventToQueue`** system calls. The type of event generated depends on the key pressed. Regular alphabetic and numeric keys use **`EvtEnqueueKey`** and generate a *`keyDownEvent`* event with the **`EventType`** structure filled in with the *`chr`* or *`vchr`* value appropriate to the key pressed.

The keyboard also has a series of special keys that perform functions that are outside the scope of the *`keyDownEvent`*, e.g. pressing the Done key causes a similar action of the **Done** button on the screen. These special keys generate events that are specific to the type of key being pressed. Most use a series of calls to **`EvtAddEventToQueue`** that emulate a stylus tap on the button, with follow on events similar to what the O/S would generate if the button were tapped with the stylus.

> Note: There are no special keyboard events generated in lieu of the stylus tap events. In future versions of the driver, we will broadcast a special *`keyDownEvent`* event for each of the specialized keys. If the event is handled by the application, the stylus events won't be generated. If the application doesn't handle the special *`keyDownEvent`* event, the stylus events will be pumped in.

The driver determines which buttons are valid buttons to be pressed using the following logic:

```
if ((ctrl_ptr->style == buttonCtl) && (ctrl_ptr->attr.frame !=
noButtonFrame) &&
     (!ctrl_ptr->attr.graphical) && (ctrl_ptr->text!=NULL) &&
     (ctrl_ptr->attr.usable != 0)) && (StrLen(ctrl_ptr->text) > 1))
{
   // Check for a button match
   …
}
```

In the above code snippet, *ctrl_ptr* is of type **`ControlType*`**, and is a pointer to one of the active Form's U/I control objects.

V. Basic Keyboard Functionality

If you install Think Outside's driver for Palm OS on a Palm compatible device, you will automatically enable the keyboard functionality in this section. No programming is necessary!

1. Always Available

This functionality is always available, unless a 3rd-party application takes over handling the key event:

Key or Key Combination	Use and Description
FN - UP ARROW (SCROLL)	Equivalent to Palm scroll up hard button.
FN - DOWN ARROW (SCROLL)	Equivalent to Palm scroll down hard button.
DATE	Equivalent to Palm Date Book hard button.
PHONE	Equivalent to Palm Address Book hard button.
TO DO	Equivalent to Palm To Do hard button.
MEMO	Equivalent to Palm Memo hard button.
FN - DATE (APPS)	Equivalent to Palm Application Launcher silkscreened icon.
FN - PHONE (MENU)	Equivalent to Palm Menu silkscreened icon.
FN - TO DO (CALC)	Equivalent to Palm Calculator silkscreened icon.
FN - MEMO (FIND)	Equivalent to Palm Find silkscreened icon.
FN - + (✸)	Equivalent to Palm Contrast hard button.

2. Buttons and Menus

This functionality is available if the appropriate buttons or menu choices exist:

Key or Key Combination	Use and Description
DONE	Equivalent to tapping on a button labeled "Done".
FN - SPACE (NEW)	Equivalent to tapping on a button labeled "New".
FN - DONE (CANCEL)	Equivalent to tapping on a button labeled "Cancel".
FN - DEL (DELETE)	Equivalent to tapping on a button labeled "Delete".
FN - LEFT ARROW (SHOW)	Equivalent to tapping on a button labeled "Show".
FN - RIGHT ARROW (DETAILS)	Equivalent to tapping on a button labeled "Details".
FN - ENTER (OK)	Equivalent to tapping on a button labeled "OK".
CMD - <LETTER>	Menu shortcut, equivalent to Graffiti command-stroke + letter.
CMD - CNTL - <LETTER>	Taps a button that starts with that letter. For example, Cmd-Cntl-O would "tap" an "OK" button. Does not work if there are two buttons that start with the same letter.
FN - SPACE (NEW)	Equivalent to tapping on a button labeled "New".

3. Standard Palm Text Fields

This functionality is available within a STANDARD Palm text field:

Key or Key Combination	**Use and Description**
LEFT ARROW	Moves the cursor right one character.
RIGHT ARROW	Moves the cursor left one character.
CTRL – LEFT ARROW	Moves the cursor left one word. The cursor is placed at the beginning of the current or previous word.
CTRL – RIGHT ARROW	Moves the cursor right one word. The cursor is placed at the beginning of the next word.
CTRL – SHIFT – LEFT ARROW	Selects text from the current cursor position to the start of the current or previous word.
CTRL – SHIFT – RIGHT ARROW	Selects text from the current cursor position to the start of the next word.
SHIFT – LEFT ARROW	Selects one character of text to the left of the cursor
SHIFT – RIGHT ARROW	Selects one character of text to the right of the cursor
UP ARROW	Moves the cursor up one line, or to the previous Field if at the top of the current Field.
DOWN ARROW	Moves the cursor down one line, or to the next Field if at the bottom of the current Field.
CTRL – UP ARROW	Moves the cursor up one Field, or to the top of the current Field
CTRL – DOWN ARROW	Moves the cursor down one Field, or to the bottom of the current Field
CTRL – SHIFT – UP ARROW	Selects text from the current cursor position to the beginning of the Field
CTRL – SHIFT – DOWN ARROW	Selects text from the current cursor position to the end of the Field
SHIFT – UP ARROW	Selects text from the cursor position to the same cursor position in the previous line of text
SHIFT – DOWN ARROW	Selects text from the cursor position to the same cursor position in the next line of text
CMD – LEFT ARROW	Moves the cursor to the beginning of the line
CMD – RIGHT ARROW	Moves the cursor to the end of the line
CMD – UP ARROW	Scrolls up one page
CMD – DOWN ARROW	Scrolls down one page
FN – UP ARROW	Scrolls up one page
FN – DOWN ARROW	Scrolls down one page

WARNING! - If you do NOT use Palm's standard text field and create your own (for rich text or HTML, for example), you may lose these built-in features.

4. Usage Notes

Tab key

Most applications have a number of different Forms, each with a specific purpose. Each Form normally falls into one of four different categories: 1) data entry, 2) record list, 3) option selection, or 4) informative display. The TAB key should have a slightly different behavior depending on the type of Form currently displayed.

On a data entry Form, most of the screen is comprised of a single text Field or series of text Fields. Pressing TAB on this type of Form should move the focus incrementally between each of the different Fields on the Form. An example would be the address book. Once a record is selected for editing, the Form that appears has a number of different text Fields: name, address, city, etc. Pressing the TAB key should move from one Field to the next Field in the series. At the end of the Form, pressing TAB should set the focus on the first button. Pressing TAB again should move to the next button, and so on. At the end of the button series, pressing TAB should return to the first edit Field on the Form.

The record list Form is a reduced version of the data entry Form. This Form is used to select a record to be edited in the data entry Form. The focus should move between the list and the buttons whenever the TAB key is pressed. The arrow keys are used to select an actual record for edit.

Option selection Forms are slightly different. Most often users are not trying to enter data or make a selection for every object on the Form. On a data entry screen, TAB should move to each Field, because it is likely that text will be entered in each of them. On an option selection Form, most of the options are already set to the default values that users most often use. Only one or two items may need to be changed. In this case, the TAB key should move around the screen quickly. Pressing TAB should move from one group of similar objects to the next group of similar objects. An example would be a series of Push-Buttons. TAB should move into and out of the entire set, and not move within the set.

The informative display Form has little use for the TAB. Most often this type of screen is for an ***About*** box or something similar. The TAB, on this type of Form, should move the focus between the buttons on the Form, if more than one button; otherwise pressing it should simply highlight the single button.

For each of the Form types, the CTRL-TAB key should be used to move directly to the buttons. In this case, the event (*vchrNextField*) doesn't change, just the modifier flags do. Similarly, if the Form has a category list control in the upper right corner, TAB should move to it just after the last button is tabbed off of, and prior to moving back to a record list, text Field, or option control object.

Arrow keys

The arrow keys should be used to move within a Field or group of control objects. There are a few small nuances that are not covered in the table:

- CTRL-RIGHT-ARROW and CTRL-LEFT-ARROW moves a word at a time; the cursor should stop at the beginning of each word, regardless of the line breaks or amount of white space between words. Pressing CTRL-RIGHT-ARROW on the last word of a single Field Form should put the cursor at the end of the file.
- Pressing DOWN-ARROW when at the end of a Field should move to the next Field, in a series of Fields. The same applies when pressing UP-ARROW at the beginning of a Field. The cursor should move to the last line of the previous Field.
- Once a button has focus, the arrow keys should move the focus between the different buttons.
- The arrow keys should not move out of a group of like controls, e.g. a series of text controls, a set of buttons, a group of push buttons, etc.

Ctrl-Cmd keys

The driver supports a combination of keys (`CTRL-CMD`) that are used to select buttons on the screen, e.g. press `CTRL-CMD-O` to tap the **OK** button on a Form. Because of this capability, you should start every button on a Form with a different character, e.g. don't have an **Open** button and **Options…** button on the same Form.

Menu Shortcuts

Avoid numbers as menu shortcuts, since shortcuts are accessed through the `CMD` key, and `CMD-0` through `CMD-9` are predefined for launching applications. If you use numbers for some of the menu shortcuts, then these shortcuts may not be accessible using the keyboard.

VI. International and Custom Keyboard Layouts

Starting with **version 1.5**, Think Outside's Palm OS driver supports custom keyboard layouts.

The mapping for the layout is stored in a .pdb database. A sample CodeWarrior R6 project for generating a database from an .r file is included in the Sample Code directory of the Developer's Kit. The project is named "KbdLayout".

To **LOAD** a database, sync the .pdb file as normal.

To **DELETE** a database, use a file utility tool such as Insider.

To **SELECT** your layout, launch the Keyboard application on your device. Make sure it's version 1.5 or above. The layouts included in your database will appear in the pop-up list under the "`Layout`" section of the Keyboard application.

Here's a summary of steps to customize your keyboard layout:

- Customize the output file name (for example, "`mykeymap.pdb`") and the database name. In CodeWarrior, these may be changed in the PalmRez Post Linker settings dialog.
- In the `keymap.r` file, pick an offset number for the resource ID and a resource name:
  ```
  resource 'map_' (MAP_EXT_START + MyOffset, "MyMapExt")
  ```
  Note: *`MyOffset`* should be in the range 0 to 50.
- Customize the layout and give yourself credit:
  ```
  "American Southern",          // Appears in Layout pop-up list.
  langEnglish,                      // Defined in keymap.h.
  LANG_ENGLISH_UNITED_STATES,       // Defined in languages.h.
  "Copyright (c) 2001 My Company", // Appears in Layout Info box.
  "My Company and Me",              // Appears in Layout Info box.
  ```
- Customize the keys in the array.
- Build your `.pdb` from the `.r` file.

NOTE: DOUBLE-BYTE CHARACTER MAPS

The keymap files we have included in our sample code use single-byte characters, as denoted by:

```
#define NON-INTERNATIONAL 1
```

We are currently developing our driver to be compatible with double-byte characters. If you are creating double-byte character mappings, currently, there is a known issue where the Function (Fn) key may no longer respond properly. We are interested in your progress and your specific issues. Please write to ***devsupport@thinkoutside.com*** and let us know how we can better support you.

VII. Enhanced Keyboard Functionality

The keyboard functionality described in this section is not automatically provided by the Palm OS or by Think Outside's driver. You will have to provide the support within your application while you are developing your software.

The sample code / demo application included in the Developer's Kit gives some examples of the functionality in this section.

First, let's clarify what we mean by the term "object". An "object" is a User Interface element on a form such as a field, button, table, or checkbox. Objects can be grouped together, such as the set of buttons at the bottom of a form or a set of push button controls. An object can have also parts, such as a table with row and column entries.

1. Basic Rules of Keyboard Navigation

- Use TAB to move between objects or groups of objects.
- Use arrow keys to move within an object or object group.
- Make it obvious which object has the focus.

2. Common Sense Rule of Keyboard Navigation

- Pressing an arrow or TAB should move you in a *reasonable* direction.

3. Object Behavior

Button

Focus Indicator

When a button has focus it should be set to reverse video.

Movement

Use TAB to move into and out-of this Form object. TAB is also used to move between the buttons. Pressing the RIGHT and LEFT-ARROW keys should move the focus from one button to the next also. The difference being that with TAB, the focus can move out of the group of buttons and into another Form object. The arrow keys simply move the focus within the group of buttons. Pressing RIGHT-ARROW on the last button moves the focus back to the first button. Pressing TAB on the last button moves focus to the next object on the Form (normally located somewhere near the top of the Form).

Selection

Once a Button has focus, pressing either the SPACE bar or ENTER key should execute a Button tap.

Checkbox

Focus Indicator

A Checkbox normally has either a label, or a text value associated with it. Having both is uncommon. The label is placed on the left side of the Checkbox, and the text is placed on the right. To show that the Checkbox has focus, set either the label or the text value to reverse video.

Movement

Use the TAB-key to move into and out-of this control. When a Form has a series of Checkboxes, then the up and down arrow keys should move between them, and the TAB should move to the next Form object that is not a Checkbox. The implementation of a series of Checkboxes is normally done with a Table object. So, the use of the arrow keys to move between the checkboxes is consistent with the Table object's type.

Selection

Check and uncheck the control using either the SPACE bar or the ENTER key.

Field (Text)

Focus Indicator

A blinking cursor indicates that the text Field has focus. When the text Field does not have focus, the cursor should be hidden. If the text Field has a label, then the label should be set in reverse video.

Movement

Use the TAB key to move into and out-of this Form object. Once in the object, the arrow keys should be used to move around it.

If the Field is placed in series with a number of other text Fields, e.g. when editing an address, the UP and DOWN-ARROW keys should move between Fields when the cursor reaches the edge of the Field. TAB and SHIFT-TAB should jump between Fields also, but the difference between this and the regular arrow keys is TAB jumps immediately to the next Field, where DOWN-ARROW jumps to the next Field only if the cursor is on the last line of the Field. Pressing Tab in the last Field of the series should move out of the series and to the next U/I control object. Normally, this is a Button on the bottom of the Form.

CTRL-LEFT-ARROW should go back one word, with the cursor placed at the beginning of the word. CTRL-RIGHT-ARROW should go forward one word, again with the cursor placed at the beginning of the word. White space and punctuation should be used as delimiters and skipped accordingly.

If there is only a single Field on the Form, e.g. as in Memo Pad, CMD-UP and CMD-DOWN should go to the top and bottom of the Field, respectively. CTRL-UP and CTRL-DOWN should move to the top and bottom of the current page.

Selection

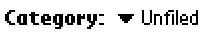
N/A

List with Popup Trigger

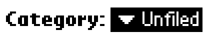
Focus Indicator

The Popup Trigger along with its associated text should be set to reverse video when focus is set to this Form object. This can be done by sending a `ctlSelectEvent` event to the control:

```
MemSet(&event, sizeof(EventType), 0);
event.eType = ctlSelectEvent;
// Highlight the button
event.data.ctlSelect.controlID = FormButtonOfInterestButton;
EvtAddEventToQueue (&event);
```


Category: ▼ Unfiled

changes to


Category: ▼ Unfiled

after the code above is executed.

Movement

TAB should move into and out-of this control. The UP and DOWN-ARROW keys choose the previous or next item in the List and change the associated text to the new value for the Popup Trigger. The arrow keys should not open the List window.

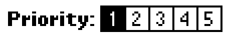
Selection

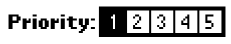
Pressing ENTER when the Popup Trigger has focus should open the List window. The UP and DOWN-ARROW keys can then be used to select an item from the List. Once an item has been selected, ENTER should make the highlighted item the current selection and close the List.

Push Button

Focus Indicator

Push Buttons are normally grouped into a set of available values for a particular software setting. For example, you could have a set of Push Buttons for selecting the priority for an email or ToDo item. When the group of Push Buttons has focus, the entire group should be outlined in a bold-lined box.



would change to


.

Another option for displaying focus on the Push Button control is to flash the current selection while the control group has focus.

Movement

Movement into the Push Button control group is done through the TAB key. Once the group has focus, the LEFT-ARROW and RIGHT-ARROW keys should be used to change the current selection.

Selection

No special selection key is needed by this control. The arrow keys are all that is required to make a new selection.

Selector Trigger

Focus Indicator

The Popup Trigger along with its associated text should be set to reverse video when focus is set to this Form object. This can be done by sending a *`ctlSelectEvent`* event to the control:

```
MemSet(&event, sizeof(EventType), 0);
event.eType = ctlSelectEvent;
// Highlight the button
event.data.ctlSelect.controlID = FormButtonOfInterestButton;
EvtAddEventToQueue (&event);
```

Movement

Use the TAB-key to move into and out-of this U/I object. Since Selector Triggers are normally not tied to other Selector Trigger objects, the arrow keys can be used to change the value of the control. For example, pressing down arrow on a date Selector Trigger could increment the date by one day.

Selection

Once a Selector Trigger has focus, pressing either the SPACE bar or ENTER key should execute a tap on the Trigger.

Table

Focus Indicator

How focus is shown depends on the objects stored in the Table. For text Fields, a blinking cursor indicates that the text Field has focus. When it does not have focus, then the cursor should be hidden. The important thing here is to not show focus for the entire table. Only the current cell should have focus indicated.

Movement

Movement through the table should be done with the arrow keys. Pressing TAB should move out of the table and into the next control. Normally this would be a button at the bottom of the Form.

Selection

The selection method depends on the type of U/I object that is present in the active cell.