handspring

Event Handlers

Application Note AN-04 Revision 1.11 Document No. 80-0120-00

Table of Contents

1. Overview

Handspring has added a system software extension to the Palm OS core to support the Springboard expansion slot. The Handspring extension includes several APIs for implementing various event handlers to facilitate system event processing. This system event processing includes the application-level software events and the kernel-level interrupt events. The different usage and relationships of the various event handler APIs are described and clarified in this application note.

API	Description
`HsAppEventHandlerSet( )`	Register application event handler procedure
`HsCardAttrSet(… hsCardAttrIntHandler ...)`	Register card interrupt handler procedure
`HsCardAttrSet(… hsCardAttrEvtHandler ...)`	Register card event handler procedure
`HsCardAttrSet(… hsCardAttrPwrHandler ...)`	Register power (management) handler procedure
`HsAppEventPost( )`	Post application event message to application event handler
`HsCardEventPost( )`	Post card event message to card event handler

2. Event Handler Processing

2.1. Interrupt Handler and Card Event Handler

To process the hardware interrupts from a module, the `CardSetup` application registers an interrupt handler with `HsCardAttrSet(… hsCardAttrIntHandler ...)`. With the Palm OS single-threaded architecture, it is very important that the interrupt handler processing time be kept to a minimum to avoid an unresponsive system. Usually, the interrupt handler is limited to copying data buffers and the manipulation of some register controls. Thus, the Handspring extension provides an application-level event handler called the *card event handler* to defer the heavy processing to the application level, if necessary.

The card event handler is provided for applications to handle *card-specific* events that require a certain amount of data processing. These events commonly require system responses or user interfaces available only at the application level (i.e., unavailable at the kernel level of the interrupt handler). If used, the card event handler is registered by the `CardSetup` or the main application using `HsCardAttrSet(… hsCardAttrEvtHandler ...)`. When the interrupt handler receives an event-triggered interrupt that requires processing deferral to the card event handler, the interrupt handler calls the `HsCardEventPost()` API to post a message to the card event handler's message queue. The Palm OS notifies the card event handler of a message to process.

Note: If the Springboard module is designed to work with third-party applications, the interrupt handler and card event handler should be implemented in the `CardSetup` application or the module's shared library. This encapsulates the interrupt and card handlers with the rest of the module's system software. Thus, third-party applications are removed from these low-level software details; third-party applications should use the application event handler to implement event processing.

Note: If the card event handler is implemented in the `CardSetup` application or in a shared library, the `CardSetup` application or shared library must temporarily open its resource database to make user interface calls. Opening the resource database enables the system to use the correct resource database. When user interface calls are no longer needed, the resource database should be closed.

In addition to the interrupt handler and the supporting card event handler, the Handspring extension also provides support for handling power management through the power handler. If used, the `CardSetup` or main application can register the power handler using `HsCardAttrSet(… hsCardAttrPwrHandler ...)`. When the system detects power events such as sleep, wake, or low battery, the system will call the event handler to let the power handler manage the card's power control. The power handler routine must observe the same restrictions as an interrupt handler.

2.2. Application Event Handler

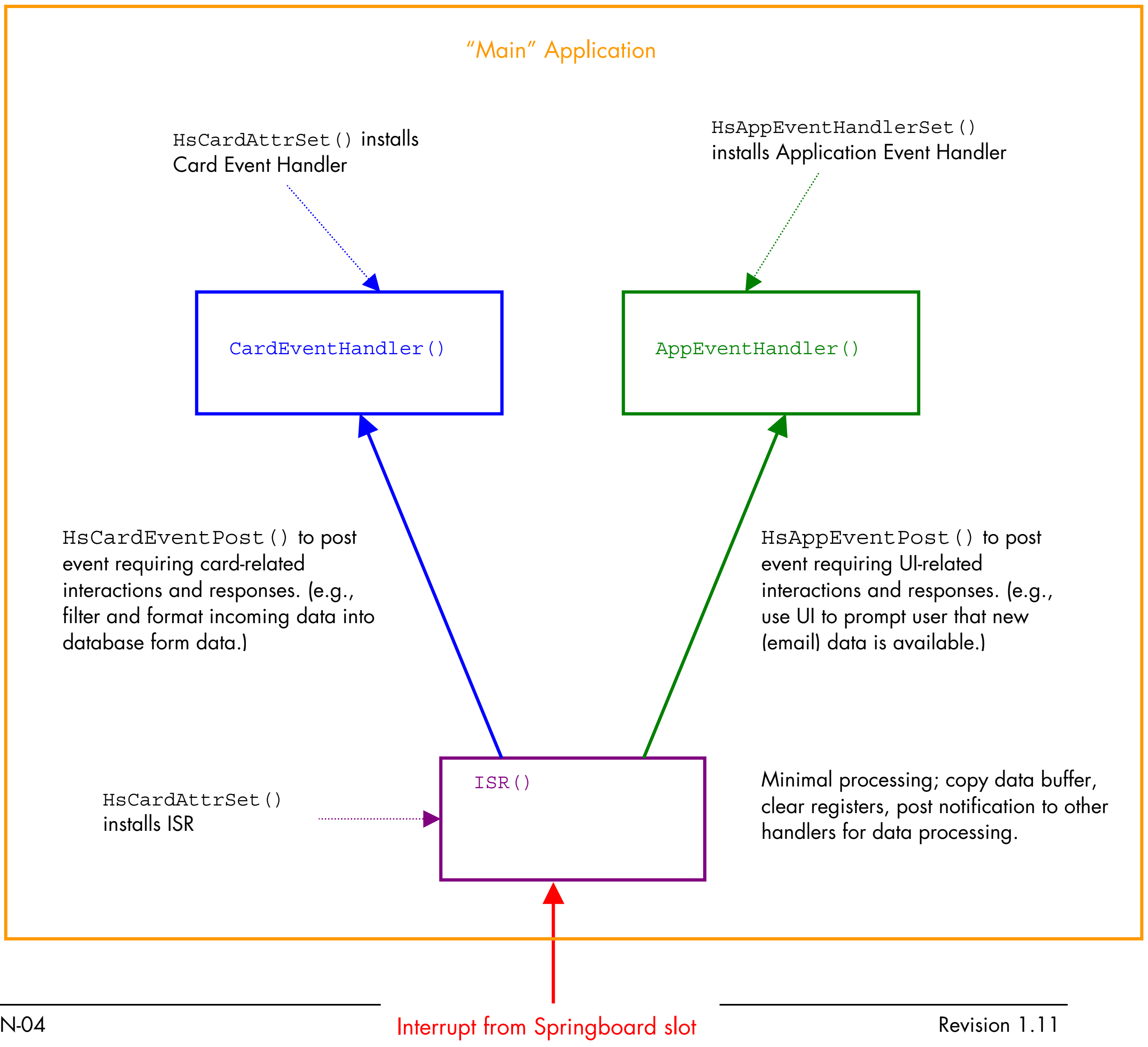
A Springboard software application can consist of multiple applications and multiple libraries (such as the Setup application, Welcome application, "main" applications, and serial library). This assimilation of multiple code resources into a unified application solution neccesitates a facility for the different code resources to communicate various event conditions to the application. To satisfy this requirement, the Handspring system extension provides the application event handler to receive application event postings.

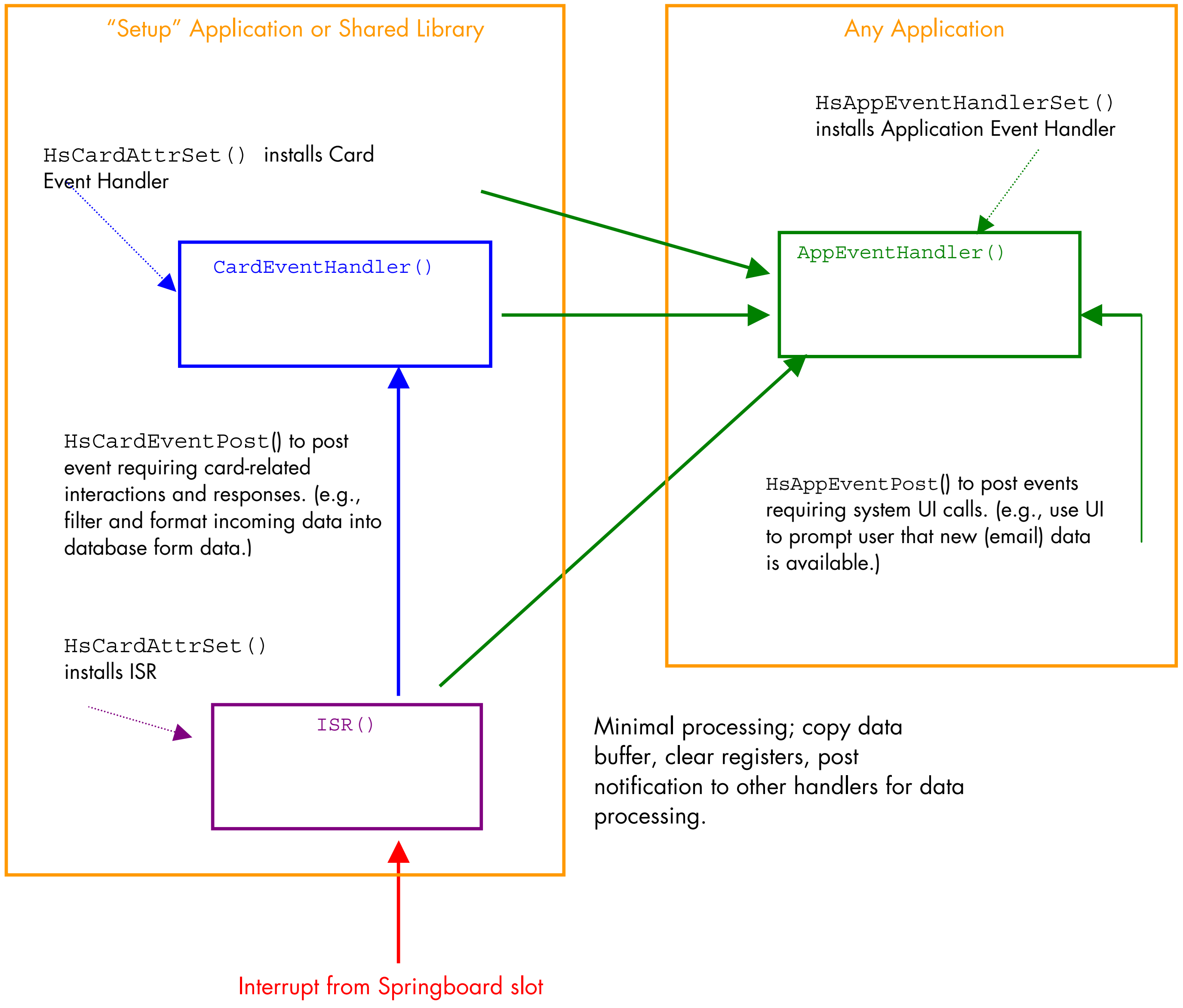
The application event handler is provided for applications to handle *general application* events that require some processing. (The card module needn't be present to post event messages to the application event handler.) If used, the application event handler is registered by the main application using the `HsAppEventHandlerSet()`. When the interrupt handler receives an event-triggered interrupt that requires processing deferral to the application event handler, the interrupt handler calls the `HsAppEventPost()` API to post a message to the application event handler's message queue. The Palm OS will notify the application event handler of a message to process.

Note: Each application may have its own unique application event handler. Only the currently running application's application event handler will receive the `post` message from `HsAppEventPost()`.

2.3. Event Handler Interaction

The following diagrams illustrate the event messaging and processing between the interrupt handler to the card event handler and the application event handler. The diagrams demonstrate a sample application as it processes a wireless email application.


"Main" Application
HsCardAttrSet() installs Card Event Handler
HsAppEventHandlerSet() installs Application Event Handler
CardEventHandler()
AppEventHandler()
HsCardEventPost() to post event requiring card-related interactions and responses. (e.g., filter and format incoming data into database form data.)
HsAppEventPost() to post event requiring UI-related interactions and responses. (e.g., use UI to prompt user that new (email) data is available.)
HsCardAttrSet() installs ISR
ISR()
Minimal processing; copy data buffer, clear registers, post notification to other handlers for data processing.
Interrupt from Springboard slot


"Setup" Application or Shared Library
HsCardAttrSet() installs Card Event Handler
CardEventHandler()
HsCardEventPost() to post event requiring card-related interactions and responses. (e.g., filter and format incoming data into database form data.)
HsCardAttrSet() installs ISR
ISR()
Interrupt from Springboard slot
Any Application
HsAppEventHandlerSet() installs Application Event Handler
AppEventHandler()
HsAppEventPost() to post events requiring system UI calls. (e.g., use UI to prompt user that new (email) data is available.)
Minimal processing; copy data buffer, clear registers, post notification to other handlers for data processing.

3. History

Date	Revision #	Description of changes
11 Dec 00	1.11	Minor formatting changes to improve readability.
11 May 00	1.10	Added restriction for UI call for Card Event Handler in Setup app. Added diagram showing multi-apps interaction.
4 Apr 00	1.00	Initial release.