Handspring Product Guide: Handspring Visor Platinum

Release 1.1

Information herein is preliminary and subject to change without notice.

TRADEMARK ACKNOWLEDGMENT

Document number: 80-0095-00

Handspring, Inc.

189 Bernardo Ave.

Mountain View, CA 94043-5203

TEL: (650) 230-5000

FAX: (650) 230-2100

www.handspring.com

Table of Contents

1 Product Specifications 6

1.1 Introduction 6

2 Implementation Details 7

2.1 Module Access Time and Wait States 7

2.2 Interrupt Latency 7

2.3 Battery Voltage Thresholds 8

2.4 Springboard Expansion Slot Signal Specification 8

2.4.1 Address Bus 8

2.4.2 Microphone 8

2.5 DC Characteristics 8

2.5.1 Power Supply (Vcc) 8

2.5.2 Battery Voltage 8

3 Cradle Interface 9

3.1 Functional Description 10

3.1.1 Serial Communications 10

3.2 Pinout 12

3.3 Signal Descriptions 12

3.4 Keyboard Support via Remote UI 14

3.4.1 Remote UI Packet Description 14

3.4.2 Remote UI Packet Header 15

3.4.3 Remote UI Packet Body 16

3.4.4 CRC Computation 17

3.4.5 Remote UI Packet Example 18

4 Mechanical Information 20

4.1 Handspring Handhelds 20

4.2 Springboard Modules 21

4.3 Cradle Base 22

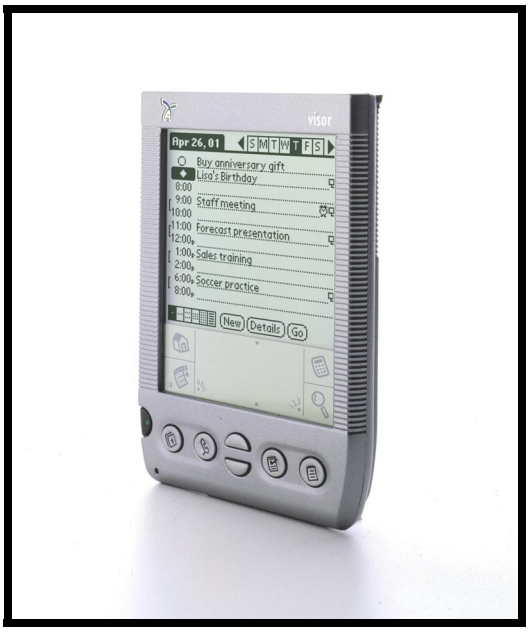
4.4 Non-Encroachment Zones and Backward Compatibility 23

5 Environmental Test Specifications 24

6 Handspring Developer Agreement 25

Change History		
Revision	**Date**	**Description**
1.1	10/16/00	**Cradle Interface – TXD/Power Pin Specification**: The power available through the TXD pin on the cradle interface has changed. The specification is 2.7V, 3mA maximum. The short circuit current is 6mA minimum.

1 Product Specifications



1.1 Introduction

This guide contains information specific to the Handspring Visor Platinum handheld computers. Specifications for the cradle interface are included in this guide. For details about the Springboard Expansion Slot, refer to the Springboard Development Guide for Handspring Handheld Computers.

Table 1. Product Specifications

Features	Specifications
CPU	Motorola MC68VZ328 Dragonball-VZ (33MHz)
Internal Memory	8MB
Software in ROM	Standard Palm OS applications, plus DateBook+, Advanced Calculator, CityTime, MathLib, and Remote UI
Operating System	Palm OS 3.5.1 plus Handspring Extensions
Springboard	Springboard 1.1
Springboard Expansion Slot Memory	32MB Maximum. 16MB maximum per chip select.
LCD	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)
LCD Controller	Integrated onto CPU. Supports 16 level grayscale.
UART	Integrated onto CPU
IR	Standard Palm OS Infrared
Cradle Interface	USB, Serial (TX/RX, TTL level)
Battery	Two Standard AAA batteries

2 Implementation Details

2.1 Module Access Time and Wait States

The Visor Platinum handheld uses a Dragonball VZ processor running at ~33 MHz. The CPU can be setup to insert zero to six wait states when accessing the Springboard Expansion Slot. The system initially sets the slowest access speed possible to read the header information on the Springboard module. The "access time" (in nanoseconds) supplied in the header is then used to calculate the appropriate number of wait states needed.

The minimum access time for Springboard modules on Visor Platinum is 130ns. Measured throughput for the Springboard bus on Visor Platinum is:

- Write: ~1.5Mbytes/second
- Read: ~1.8Mbytes/second

2.2 Interrupt Latency

Table 2. Interrupt Latencies

Condition	Measured Latency	Maximum Springboard Latency Specification
Device already awake	0.033 ms	0.15 ms
Device asleep (*sysAwakeP == false)	3.07 ms	4 ms
Device asleep (*sysAwakeP == true)	4.38 ms	10 ms

This section describes the interrupt latency times for the Visor Platinum handheld computer. The latency times were measured on an actual unit and include all exception processing and function call overhead to get to the first instruction of the module interrupt handler.

The three conditions include: 1) when the device is already awake, 2) the first time the interrupt handler is called after coming out of sleep mode, but before the rest of the system is awake (i.e., the `*sysAwakeP` parameter is false), and 3) the second time the interrupt handler is called after coming out of sleep mode, which is after the rest of the system has awakened (i.e., the `*sysAwakeP` parameter is true).

In the third case, the designer should consider adding an *additional* 20 milliseconds if the operating system needs to open the USB library. Note that this case happens only if the USB library was open when the device went to sleep. This situation is rare, because most applications will close the USB library before the device goes to sleep.

Note that the `HsCardErrTry/HsCardErrCatch` macros will generate additional delays. These delays should be considered when working with timing-sensitive code such as interrupt handlers.

2.3 Battery Voltage Thresholds

The Palm OS specifies two voltage thresholds which can be retrieved through the `SysBatteryInfo()` function:

> **Warning Threshold:** When the battery voltage drops below the Warning Threshold, the system puts a `lowBatteryChr` key event in the queue. Applications call `SysHandleEvent()`, which in turn calls `SysBatteryDialog()` in response to this event.
>
> **Critical Threshold:** When the battery voltage drops below the Critical Threshold the system turns itself off without warning in order to protect user data.

The LOWBAT* signal on the Springboard Expansion Slot is asserted when the battery voltage falls below a second Critical Threshold. This signal is only asserted for several milliseconds before power to the Springboard slot is removed and the buffers driving the Springboard signals are disabled. This action is implemented in hardware rather than through the operating system.

Developers should check that sufficient power is available before putting a module into a high power state.

Table 3. Battery Voltage Thresholds

Threshold	System Action	Voltage
Battery at full capacity	Battery at full capacity. No action.	3.0V
Warning Threshold	The operating system begins to warn the user that batteries are low.	2.1V
Critical Threshold – Software	The operating system puts the system into sleep mode to protect user data.	1.6V
Critical Threshold - Hardware	A hardware comparator removes power to Springboard Slot.	1.6V

2.4 Springboard Expansion Slot Signal Specification

The Visor Platinum Handheld incorporates a Springboard 1.1 Expansion Slot.

2.4.1 Address Bus

A0 is not used (pulled low). All memory accesses are 16 bits wide and conducted on even-byte boundaries.

2.4.2 Microphone

The Visor Platinum handheld uses a microphone with a 2.2KΩ impedance, a standby operating voltage of 2.0V, and a maximum current consumption of 0.5mA.

2.5 DC Characteristics

2.5.1 Power Supply (Vcc)

The Springboard specification for Vcc is 3.0V to 3.6V. Nominal Vcc on Visor Platinum is 3.1V.

2.5.2 Battery Voltage

The maximum voltage allowed through the battery contacts is 3.2V.

3 Cradle Interface

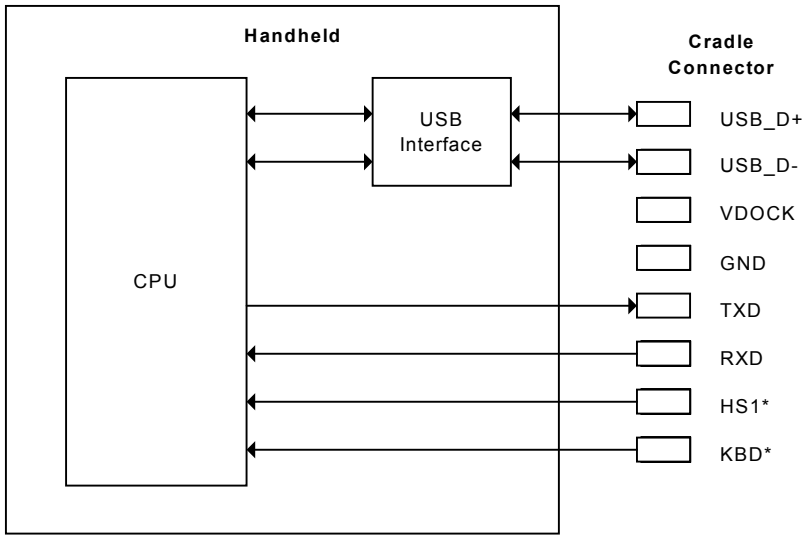
The cradle connector provides serial communications with Handspring's family of handheld computers. The cradle connector is located at the bottom of the handheld device. It is typically used for communicating with a PC or Mac for data synchronization; however, a variety of peripherals such as a keyboard, pager, or modem can also be interfaced to the handheld unit through this port.

Figure 1: Cradle view

3.1 Functional Description

The Cradle Connector contains two serial buses that communicate with external devices: a USB bus and a simple serial bus. The high-speed USB interface is a slave-only interface, providing a payload bandwidth between the host PC or Mac and the handheld device. The serial port provides asynchronous capability to low-speed devices (9600 bps or less), however, higher speeds may be observed. Figure 2 shows how the Cradle Connector interfaces with the handheld.

Figure 2: Cradle Connector Interface



3.1.1 Serial Communications

The serial interface on the Visor Platinum handheld cradle connector is implemented with only TXD and RXD signals. Unlike other Palm device, there is no hardware handshaking provided. Additionally, the TXD (transmit data) and RXD (receive data) signals implement TTL logic levels based on system Vcc (3.1-volts in Visor Platinum). The serial cradle contains level-shifting circuitry that enables the handheld to work with RS-232 logic levels required on a desktop or laptop computer.

Figure 3: View of Visor Platinum handheld bottom

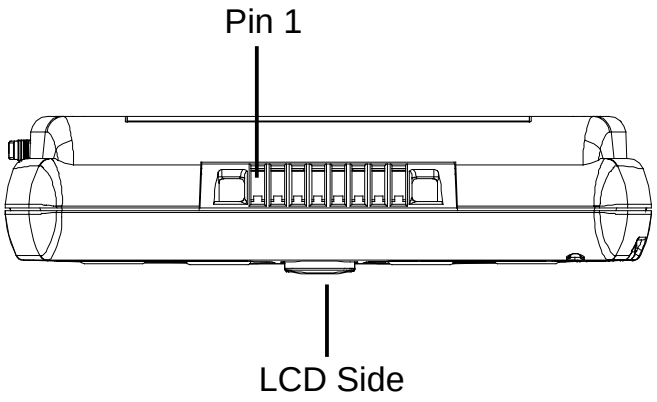
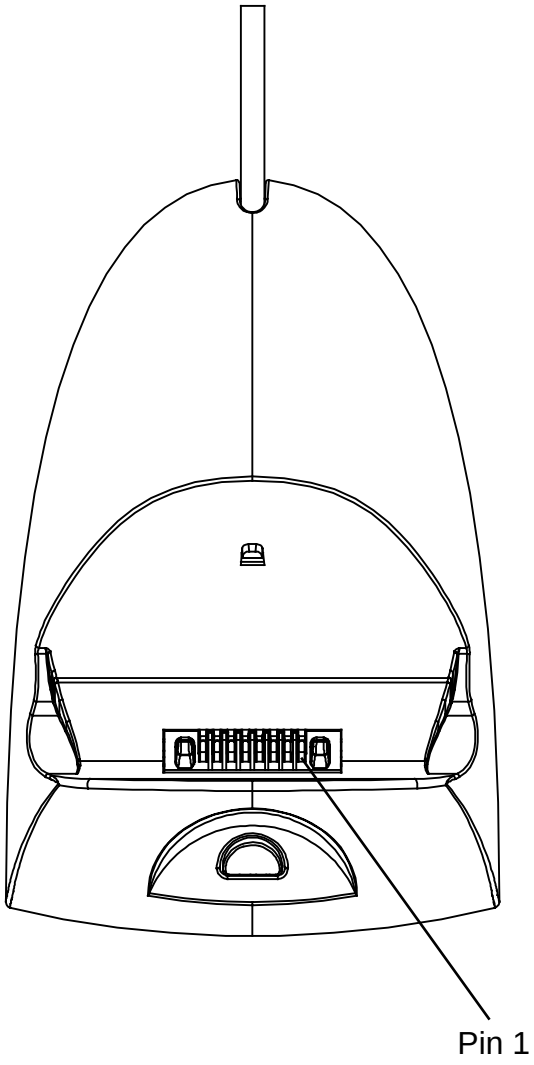


Figure 4: View of Eight-Pin Cradle Connector



3.2 Pinout

This section defines the functions of the signals on the eight-pin Cradle Connector. Figure 3 is a view of the Visor Platinum base, showing the eight pins. Table 4 lists the signals with their respective pin assignments. The signals are described in alphabetical order following the table. Active-low signals are identified with a "*" at the end of their names.

Table 4: Cradle Connector Pin Summary

Pin	Name	I/O/P[1]	Function
1	RXD	I	Receive Data
2	KBD*	I	Keyboard Detect
3	HS1*	I/PU	HotSync Interrupt
4	GND	P	Ground
5	USB_D-	I/O	USB Data
6	USB_D+	I/O	USB Data
7	VDOCK	P	Cradle Power
8	TXD	O	Transmit Data/Power

I = input, O = output, P = power, PU = pulled up. For example, the HS1* signal is a direct input signal to the Visor Platinum CPU.

3.3 Signal Descriptions

Ground **GND**

GND is the ground connection to the handheld. This signal must be connected to the ground reference in the cradle or peripheral.

HotSync Interrupt **HSI***

This active-low interrupt pin is asserted low in order to initiate a HotSync® with the handheld. In a cradle application, a push button would momentarily short this signal to GND to begin a HotSync. This signal is pulled up internally within the CPU.

Keyboard Detect **KBD***

This active-low pin is held low in order to indicate the presence of a keyboard or a serial cradle. While KBD* is held low, the handheld expects data to be sent on the RXD pin. Refer to the next section on keyboard support for information on the keyboard data packet structure supported by the handheld.

When this signal is asserted in Visor Platinum, a keyboard daemon is automatically launched to support keyboard input. The signal is detected when the handheld is placed in the cradle. If the handheld is reset while in the cradle, the daemon will not be re-launched. When Hotsync is asserted, this signal is also polled to determine which communications library should be used.

Receive Data **RXD**

RXD connects directly to the Visor Platinum's CPU UART. **Note that RXD is TTL level, not RS-232 level**. This signal is used for asynchronous serial communications between the handheld and a cradle or peripheral. RXD is an input to the handheld and an output from a cradle or peripheral.

Transmit Data/Power **TXD**

TXD connects directly to the CPU's UART. **Note that TXD is TTL level, not RS-232 level**. This signal is used for asynchronous serial communications between the handheld and a cradle or peripheral. TXD is an output from the handheld and an input to a cradle or peripheral. Internally, this pin is the output of the gate with the series resistor. This pin provides up to 3 mA maximum at 2.7 V when KBD* is asserted for low-power peripherals, such as keyboards. Short circuit current is 6mA minimum.

USB Data **USB+, USB-**

USB_D+ is the positive signal in the USB differential pair. USB_D- is the negative signal. These signals implement the USB signaling protocol for communicating with a USB host, such as a PC or a Mac.

Cradle Power **VDOCK**

This pin can provide a charging supply to the module when the handheld is placed in a special charging dock. The handheld passes this charging supply from a pin on its cradle connector through to pins (VDOCK) on the Springboard Expansion Module connector. VDOCK provides 4.75 - 6.2V on two pins, with a maximum current of 500mA.

3.4 Keyboard Support via Remote UI

The handheld platform supports character input from an external keyboard or pen-based device through the Cradle Connector located at the bottom of the handheld device. Pin 2 of the eight-pin Cradle Connector is the "keyboard detect" pin (KBD*). Grounding this pin indicates to the handheld processor that:

1. A keyboard or other remote UI device is present on the Cradle Connector, and,
2. Incoming serial data packets on RXD (pin 1 on the Cradle Connector) should be interpreted as described by this document.

Note that the Cradle Connector does not include hardware signaling for buffer overflow conditions within the handheld device. Therefore, the maximum recommended serial transfer speed to the handheld device is limited to 9600 bps.

Remote UI is supported in all existing versions of Palm OS; for more information please see the *Palm OS Programmer's Companion* at http://www.palmos.com/dev/tech/docs/.

3.4.1 Remote UI Packet Description

As long as pin 2, KBD*, on the Cradle Connector is held low, the handheld will receive incoming data packets on RXD and interpret them as Remote UI Packets. Remote UI Packets have three sections: a header, a body, and a CRC, as shown in Table 2.

Table 2: Remote UI Packet

Header	Body	CRC	
10	16	2	# of bytes

The Remote UI Packet structure is flexible enough to support remote input from a variety of devices, but for software simplification, most of these fields can be hardcoded for keyboard-specific input.

Table 3 describes in detail the fields in the header, body, and CRC sections.

Table 3: Remote UI Packet Fields

Field	Data Length (in Bytes)	Parameter Name	Value	Comment
Header Fields				
1	2	signature1	0xBEEF	Indicates serial link packet.
2	1	signature2	0xED	Indicates serial link packet.
3	1	dest	0x02	Indicates remote UI serial link packet.
4	1	src	0x02	Indicates remote UI serial link packet. Typically used as "return address" for response messages. Not required for keyboard input.
5	1	type	0x00	Indicates system packet type to the handheld.
6	2	bodySize	calculated	Size of body in bytes.
7	1	transactionID	calculated	Increment by one for each new message. Not required for keyboard input (will not be checked).

Field	Data Length (in Bytes)	Parameter Name	Value	Comment
8	1	checksum	calculated	8-bit sum of header fields NOT including this field.
Data Fields				
9	1	command	0x0D	Indicates that input is a remote event.
10	1	filler	don't care	For word alignment.
11	1	penDown	0 or 1	Indicates pen event. Reset to 0 for keyboard input.
12	1	filler	don't care	For word alignment.
13	2	penX	0	Pen X coordinate. Reset to 0 for keyboard input.
14	2	penY	0	Pen Y coordinate. Reset to 0 for keyboard input.
15	1	filler	don't care	For word alignment.
16	1	keyPress	0x01	Indicates a key has been pressed.
17	2	keyModifier	lookup	Modifier bits (shift, control, etc.) Bitmapped to this 16-bit field.
18	2	keyAscii	lookup	Palm OS keycodes see **chars.h.**
19	2	keyCode	0x00	Reserved - Always set to 0.
CRC Field				
20	2	CRC	calculated	Computed using table method; see Section 3.4, "CRC Computation," for more information. Uses big-endian byte ordering.

The following subsections provide more details on the contents of the header, body, and CRC.

3.4.2 Remote UI Packet Header

The Remote UI Packet header consists of eight fields as shown in Table 4. Each field consists of the number of bytes indicated.

Table 4: Remote UI Packet Header

signature1	signature2	dest	src	type	bodySize	transactionID	checksum
2	1	1	1	1	2	1	1

The first two fields (`signature1` and `signature2`) contain a predefined code that indicates to the handheld that the incoming packet is a serial link packet.

The `dest` and `src` fields refer to the logical socket used for communication for remote UI. For keyboard applications, these fields are both set to 0x02.

The `type` field indicates that this packet is a system packet. It is hardcoded to 0x00.

The 16-bit `bodySize` field must contain the size of the body portion of the packet (in bytes). Do not include the size of the header or CRC bytes in this calculation.

The `transactionID` field is a running message counter, and is typically used for two-way communications over the serial port. Reply messages are tagged with the `transactionID` of the original message. Because

the handheld does not send responses to keyboard packets, this field is not used and can be set to any value. However, other types of remote UI devices should increment `transactionID` by one for each packet sent.

The `checksum` value is a simple eight-bit addition of the bytes in the header. If a checksum mismatch occurs, the handheld searches all incoming data bytes for `signature1` in order to resynchronize to the sender. Sample code to generate a checksum is shown below.

```
/*************************************************************************
 * SlkChecksum      PrvCalcHdrChecksum(Checksum SlkChecksum, BytePtr bufP,
 *                  Long count);
 *
 * Computes the 16-bit checksum of bytes in a buffer.
 *
 * Arguments:
 *            SlkChecksum start                -- starting checksum value
 *            BytePtr bufP                     -- ptr to the data buffer
 *            Long count                       -- number of bytes in buffer
 *
 * Returns:
 *            8-bit checksum of the data
 *
 *************************************************************************/
static SlkPktHeaderChecksum
PrvCalcHdrChecksum(SlkPktHeaderChecksum start, BytePtr bufP, Long count)
{
       // The compiler produces the fastest code with a while(--count) loop...
       do {
              start += *bufP++;
              } while(--count);


       return( start );
}
```

3.4.3 Remote UI Packet Body

Table 5 shows the Remote UI Packet body. Each field consists of the number of bytes indicated.

Table 5: Remote UI Packet Body

command	filler	pen Down	filler	penX	penY	filler	key Press	key Modifier	keyAscii	key Code
1	1	1	1	2	2	1	1	2	2	2

The `command` field is always set to 0x0D, indicating that the packet contains remote event data. The `penDown`, `penX`, and `penY` fields define pen events; they are all reset to 0 for keyboard input. The `keyPress` field contains a flag that indicates a keypress has occurred on the remote UI device. The `keyModifier` and `keyAscii` fields define the keypress. The `keyCode` field is reserved for future purposes and should always be set to 0x0.

The `keyAscii` values supported by Palm OS are defined in the `chars.h` include file within the Palm OS source code. The `keyModifier` values are defined in the Palm OS source file `event.h`; they are also listed in Table 6 below.

Table 6: Key Modifiers for Palm OS

Key	KeyModifier Field Values
shiftKey	0x0001
capsLock	0x0002
numLock	0x0004
commandKey	0x0008
optionKey	0x0010
controlKey	0x0020
autoRepeatKey	0x0040
doubleTapKey	0x0080
poweredOnKey	0x0100
appEvtHookKey	0x0200
libEvtHookKey	0x0400

3.4.4 CRC Computation

The 16-bit cyclic redundancy check (CRC) value is calculated using the contents of both the header and the body. Note that the handheld device uses big-endian byte ordering for computing a CRC, so keyboards must generate CRCs using big-endian byte ordering. CRCs are computed using the table look-up method. Source code to generate CRCs is shown in the example below.

If the handheld detects a bad CRC, the packet is ignored, but no error response is sent back to the keyboard.

```
/*************************************************************************************
* FUNCTION: Crc16CalcBlock
*
* DESCRIPTION: Calculate the 16-bit CRC of a data block using the table lookup method.
*
* PARAMETERS:
*             bufP                -- pointer to the data buffer;
*             count               -- the number of bytes in the buffer;
*             crc                 -- the seed crc value; pass 0 the first time
*                                    function is called, pass in new crc result
*                                    as more data is added to packet and crc is updated.
* RETURNS:
*             A 16-bit CRC for the data buffer.
*
*************************************************************************************/
Word Crc16Calc (VoidPtr bufP, Word count,  Word crc)
{
       register     BytePtr      byteP  =  (BytePtr)bufP;
       WordPtr              crctt       =  (WordPtr)crctt_16;            // CRC
translation table

//
// Calculate the 16 bit CRC using the table lookup method.
//
       if ( count )  {
              do  {
```

```
                    crc = (crc << 8) ^ crctt[ (Byte)((crc >> 8) ^ *byteP++) ];
                } while ( --count );
        }

        return( crc & 0xffff );
}

// This is the lookup table used when performing the 16-bit CRC calculation.  //

static Word crctt_16[ 256 ] =
{
0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};
```

3.4.5 Remote UI Packet Example

The following example shows a hex dump of a packet representing the keystroke ‘a’ (lowercase ‘a’):

```
BE    EF    ED    02    02    00    00    10

02    B0    0D    CC    00    CC    00    00

00    00    01    CC    00    00    00    61

00    00    2C    D8
```

Table 7 categorizes the above data into the respective packet fields.

Table 7: Packet Example Breakdown

Header	
BE EF	signature1
ED	signature2
02	dest
02	src
00	type
0010	bodySize
02	transactionID
B0	checksum
Body	
0D	command
CC	filler
00	penDown
CC	filler
00 00	penX
00 00	penY
01	keyPress
CC	filler
00 00	keyModifier
00 61	keyAscii
00 00	keyCode
Checksum	
2C D8	CRC

4 Mechanical Information

Complete mechanical information for Handspring handheld devices and associated peripherals are updated on the website. The formats used are DWG, DXF, PRO-E or IGES. For the latest files go to:

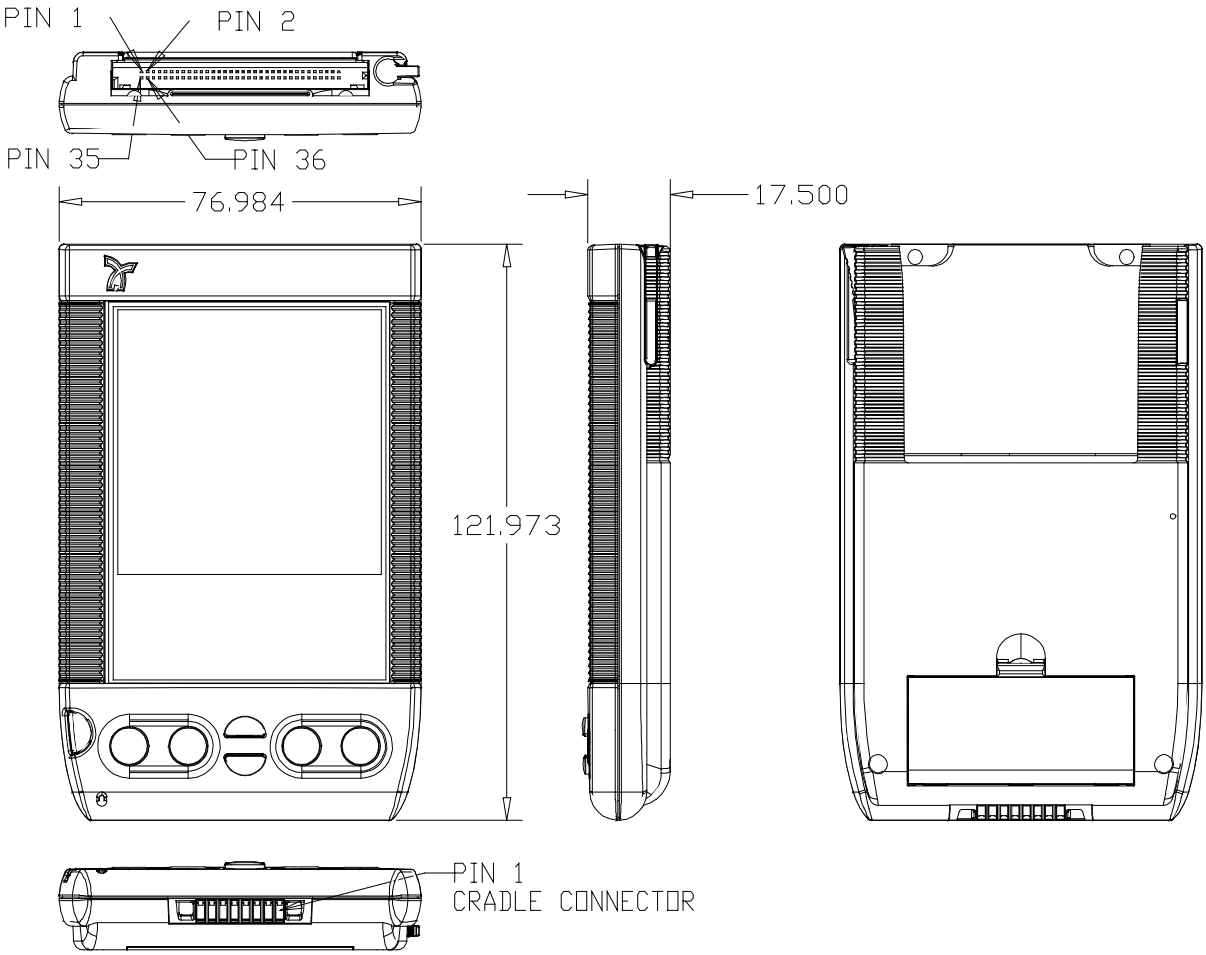
http://www.handspring.com/developers/dev_mechanical.jhtml

Mechanical files on the website cover three general topics:

- Handspring handhelds
- Springboard modules
- Handheld cradle

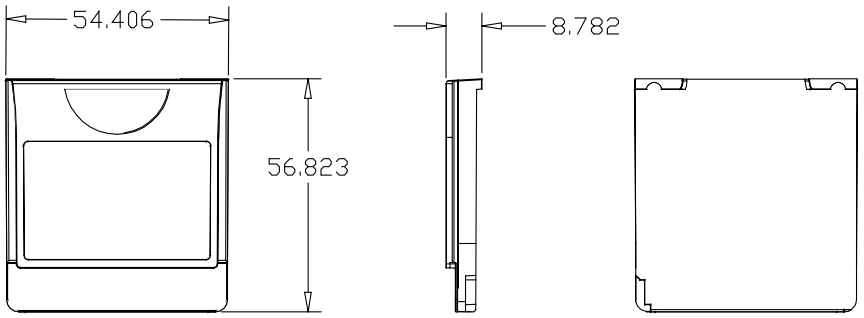
4.1 Handspring Handhelds

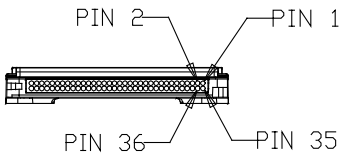
Mechanical files for all Handspring handhelds are fully documented on the Handspring web site.

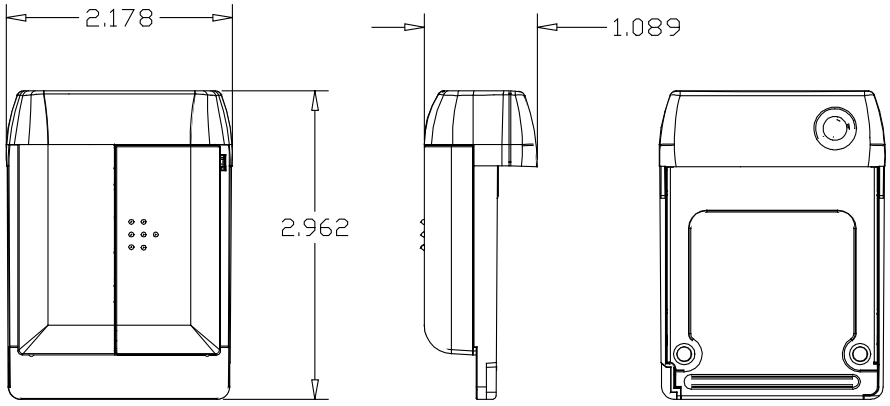


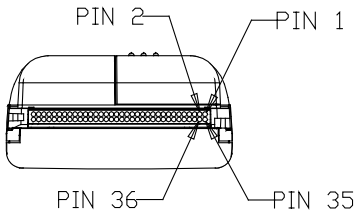
4.2 Springboard Modules

There are several mechanical variations of the Springboard module. Some modules are designed to house internal batteries. The Handspring modem module is an example of this design. The extra space provided in this design can also be used to house larger components that may not fit within a Standard Module. The latest mechanical files are available on Handspring's website.



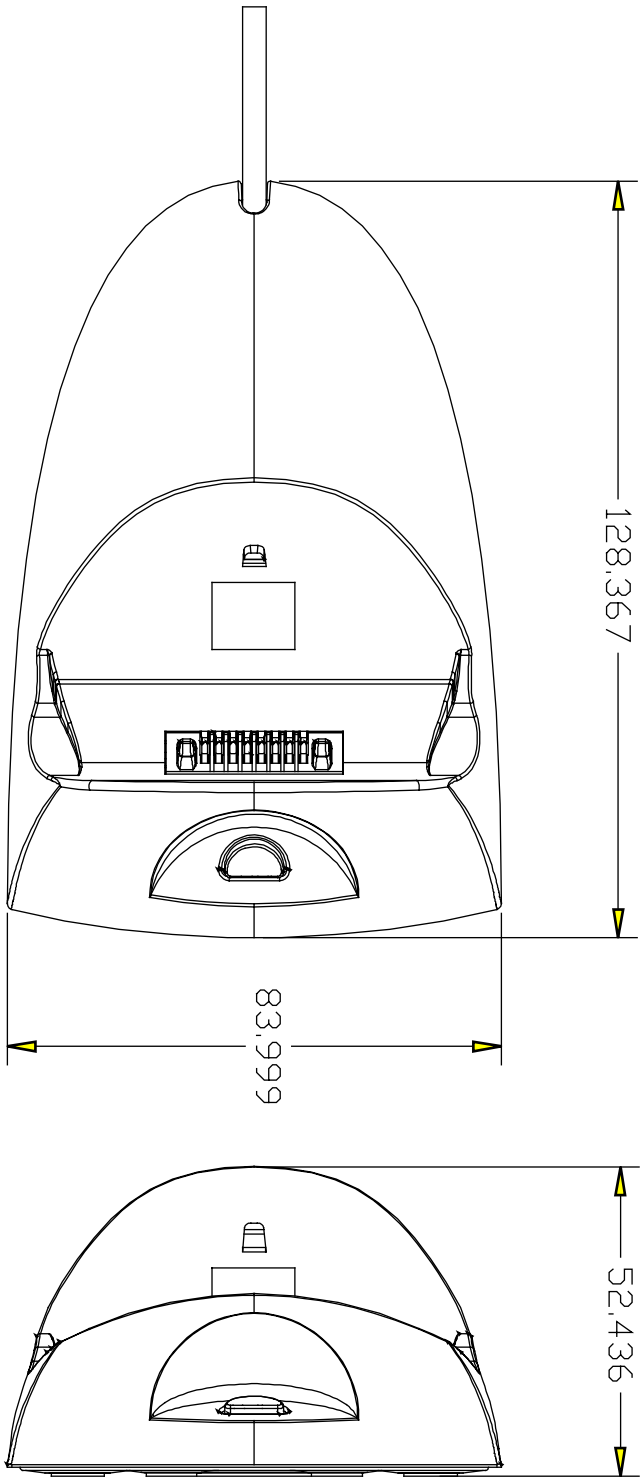






4.3 Cradle Base

Mechanical files for Handspring cradles are available on the website.



4.4 Non-Encroachment Zones and Backward Compatibility

One of Handspring's goals is to provide future products that allow backward-compatibility with earlier Springboard module designs; however, take care when designing your Springboard modules. To maintain compatibility with existing Springboard modules, the Springboard Expansion Slot depth and width will remain constant, but the surrounding molding may change in thickness or size. Please refer to the mechanical drawings on the website for the recommended non-encroachment areas. For further details on mechanical information for Springboard, please refer to the Mechanical Information section of the *Springboard Development Guide for Handspring Handheld Computers*.

5 Environmental Test Specifications

Altitude	
Operating	15,000 ft @ 25° C
Non-Operating	30,000 ft @ 25° C
Ambient Temperature	
Operating Range	-10°C to 40°C
Power-off Storage	-20°C to 60°C
Temperature Gradient	20°C/hour maximum
Ambient Humidity	
Operating	5% to 90% RH
Power-off Storage	5% to 90% RH
ESD	
Non-metallic areas	Performed to IEC 1000-4-2 +/- 8kV air discharge
Metallic areas	Performance to IEC 1000-4-2 +/- 4kV touch discharge
Durability	
Shock – Mechanical	ASTM D3332, Method A (Operational) Step velocity – six sides, minimum of 130 in/sec, 2.0mS duration half-sine pulse 10 impacts per side
Shock – Drop Test Non Operational	48” drop onto carpeted surface and 36” drop onto rigid surface, six sides and four corners (excluding LCD)
Springboard Insertion Test	4000 insertions
Agency Approvals	
North American EMC, EMI, and Safety	Test in accordance with FCC-CFR, Title 47, Part 15 and applicable third party hardware specification. Test compliance with UL, CSA, and other applicable agencies.
Other Regional Agency Approvals	As required Examples: C-Tick (Australia), VCCI (Japan), CE (Europe), FCC Class B (US), CSA (Canada)

6 Handspring Developer Agreement

HANDSPRING, INC.
Developer Agreement

PLEASE READ THE TERMS OF THE FOLLOWING AGREEMENT CAREFULLY. BY USING THE MATERIALS DISTRIBUTED WITH THIS AGREEMENT (THE "DEVELOPMENT KIT"), YOU ARE AGREEING TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, PLEASE DO NOT USE THE DEVELOPMENT KIT. INSTEAD, PLEASE DESTROY ALL COPIES OF THE DEVELOPMENT KIT WHICH YOU MAY HAVE.

DEFINITION. "Springboard™ Enabled Products" are Handspring handheld computers that contain an external "slot" (the "Springboard™ slot") into which compatible third-party hardware or software products can be inserted.

LICENSE GRANT. Subject to the terms and conditions of this Agreement, Handspring hereby grants to Developer a non-exclusive, non-transferable license under Handspring's intellectual property rights in the Development Kit (a) to use, reproduce and create derivative works of the materials provided by Handspring under this Agreement, solely internally in connection with Developer's development and manufacture of i) products which plug into the Springboard™ slot and meet Handspring's Springboard™ compatibility requirements ("Licensed Plug-Ins") or ii) accessory products (such as keyboards or reference manuals) for use with Springboard™ Enabled Products (such plug-in products and accessory products, collectively, "Licensed Products"); (b) to make, have made, use, distribute and sell Licensed Products directly or indirectly to end users for use with Springboard™ Enabled Products; and (c) to distribute the unmodified Development Kit in its entirety (including this Agreement) to third parties who agree to be bound by the terms and conditions of this Agreement.

LICENSE RESTRICTIONS. Except as otherwise expressly provided under this Agreement, Handspring grants and Developer obtains no rights, express, implied, or by estoppel, in any Handspring intellectual property, and Developer shall have no right, and specifically agrees not to (a) transfer or sublicense its license rights to any other person; (b) decompile, decrypt, reverse engineer, disassemble or otherwise reduce the software contained in the Development Kit to human-readable form to gain access to trade secrets or confidential information in such software, except and only to the extent such activity is expressly permitted by applicable law notwithstanding such limitation; (c) use or allow others to use the Development Kit, in whole or part, to develop, manufacture or distribute any products other than Licensed Products; (d) use or allow others to use the Development Kit, in whole or part, to develop, manufacture or distribute products (including Licensed Products) for use as a plug-in or accessory to any product other than Springboard™ Enabled Products; (e) use or allow others to use the Development Kit, in whole or part, to develop, manufacture or distribute any products incorporating an external or internal slot design; or (f) modify or create derivative works of any portion of the Development Kit.

OWNERSHIP. Handspring is the sole and exclusive owner of all rights, title and interest in and to the Development Kit, including, without limitation, all intellectual property rights therein. Developer's rights in the Development Kit are limited to those expressly granted hereunder. Handspring reserves all other rights and licenses in and to the Development Kit not expressly granted to Developer under this Agreement. Subject to Handspring's rights in the Development Kit and the Springboard™ Enabled Products, Developer shall retain all rights in the Licensed Products developed by Developer in accordance with this Agreement.

COMPATIBILITY TESTING AND BRANDING. Prior to Developer's use of Handspring's Springboard™ compatibility trademark (the "Mark") in connection with a Licensed Plug-In, Developer shall conduct reasonable testing in accordance with Handspring's compatibility testing guidelines to ensure that the Licensed Plug-In conforms in all respects to Handspring's Springboard™ compatibility requirements (the "Compatibility Criteria"). Developer agrees that it will not use the Mark or make any statements claiming or implying compatibility with the Springboard™ slot in connection with any Licensed Plug-Ins which have not passed such compatibility testing and that, if Handspring determines that any Licensed Plug-In is not compliant with the Compatibility Criteria, Developer shall immediately cease use of the Mark in connection with that Licensed Plug-In. All goodwill generated by Developer's use of the Mark shall inure to Handspring's benefit.

Subject to the terms and conditions of this Agreement, Handspring hereby grants to Developer a non-exclusive, non-transferable license to use, subject to the guidelines set forth in Handspring's trademark policy and other applicable

guidelines, (i) Handspring's Springboard™ compatibility trademark solely in connection with the marketing and sale of Licensed Plug-ins which comply with the Compatibility Criteria; and (ii) artwork, icons, logos, color schemes, and other industrial designs and designations of source provided by Handspring to Developer hereunder solely in connection with the marketing and sale of Licensed Products

DEVELOPER INDEMNIFICATION. Developer will defend at its expense any action brought against Handspring to the extent that it arises from or relates to Developer's development, manufacturing, marketing or distribution of Licensed Products, and Developer will pay any settlements and any costs, damages and attorneys' fees finally awarded against Handspring in such action which are attributable to such claim; provided, the foregoing obligation shall be subject to notifying Developer promptly in writing of the claim, giving it the exclusive control of the defense and settlement thereof, and providing all reasonable assistance in connection therewith. Notwithstanding the foregoing, Developer shall have no liability for any claim of infringement to the extent required by compliance with the Compatibility Criteria.

WARRANTY DISCLAIMER. HANDSPRING MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, AS TO ANY MATTER WHATSOEVER, AND SPECIFICALLY DISCLAIMS ALL WARRANTIES OR CONDITIONS OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE.

LIMITATION OF LIABILITY. EXCEPT FOR BREACHES OF THE SECTIONS ENTITLED "LICENSE GRANT", OR "LICENSE RESTRICTIONS", IN NO EVENT WILL EITHER PARTY BE LIABLE TO THE OTHER FOR LOST PROFITS, LOST BUSINESS, OR ANY CONSEQUENTIAL, EXEMPLARY OR INCIDENTAL DAMAGES ARISING OUT OF OR RELATING TO THIS AGREEMENT, REGARDLESS OF WHETHER BASED IN CONTRACT OR TORT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TERM AND TERMINATION. This Agreement shall remain in effect for the partial calendar year ending on the first March 31 following the effective date, and shall automatically renew for additional one (1) year terms ending on each subsequent March 31, except that the Agreement shall automatically terminate if either party materially breaches or is in default of any obligation hereunder or if either party provides notice of non-renewal by January 1. The parties agree that Handspring may provide notice by making the notice available in a manner similar to the manner in which the Development Kit was made available.

GENERAL. This Agreement will be governed by and construed and interpreted in accordance with the internal laws of the State of California, excluding that body of law applicable to conflict of laws. No waiver, amendment or modification of any provision hereof or of any right or remedy hereunder will be effective unless made in writing and signed by the party against whom such waiver, amendment or modification is sought to be enforced. No failure by any party to exercise, and no delay by any party in exercising, any right, power or remedy with respect to the obligations secured hereby will operate as a waiver of any such right, power or remedy. Neither this Agreement nor any right or obligation hereunder may be assigned or delegated by Developer (including by operation of law) without Handspring's express prior written consent, which consent will not be unreasonably withheld, and any assignment or delegation without such consent will be void. This Agreement will be binding upon and inure to the benefit of the successors and the permitted assigns of the respective parties hereto. If any provision of this Agreement is declared by a court of competent jurisdiction to be invalid, void, or unenforceable, the parties will modify such provision to the extent possible to most nearly effect its intent. In the event the parties cannot agree, then either party may terminate this Agreement on sixty (60) days notice. This Agreement constitutes the entire understanding and agreement of the parties hereto with respect to the subject matter hereof and supersedes all prior agreements or understandings, written or oral, between the parties hereto with respect to the subject matter hereof.