

# Technical Reference Guide: Visor Handheld Family

Release 1.10



Information herein is subject to change without notice.

Copyright © 2001 by Handspring, Inc. All rights reserved.

**TRADEMARK ACKNOWLEDGMENT**

Handspring™, Visor™, Visor Edge™, Visor Neo™, Visor Pro™ and Springboard™ are trademarks of Handspring, Inc.

All other trademarks are the properties of their respective owners.

Document number: 80-0172-00

Handspring, Inc.

189 Bernardo Ave.

Mountain View, CA 94043-5203

TEL: (650) 230-5000

FAX: (650) 230-2100

[www.handspring.com](http://www.handspring.com)

Change History		
Revision #	Date	Description
1.10	17 September 2001	Added Visor Pro and Visor Neo products.
1.0	16 July 2001	Initial Release

## Table of Contents

---

<b>1. Product Overview .....</b>	<b>7</b>
1.1 Product Introduction History .....	7
1.1.1 Visor Solo, Visor and Visor Deluxe .....	7
1.1.2 Visor Platinum and Visor Prism .....	7
1.1.3 Visor Edge .....	8
1.1.4 Visor Neo and Visor Pro .....	8
<b>2. CPU .....</b>	<b>9</b>
2.1 CPU In Visor Handhelds .....	9
2.2 CPU Reference Information .....	9
2.3 Addressing Differences Between the EZ and VZ Processors .....	9
2.4 Springboard Timing .....	9
2.4.1 CPU Wait States and Access Time .....	10
<b>3. Memory: RAM and ROM .....</b>	<b>11</b>
3.1 RAM .....	11
3.2 ROM .....	11
3.2.1 Tips For Working Without A Serial Number .....	11
<b>4. Operating System .....</b>	<b>12</b>
4.1 Software in ROM .....	12
<b>5. Display and Display Controller .....</b>	<b>13</b>
5.1 Grayscale Support .....	13
5.2 16-Bit Color Support .....	14
5.2.1 Determining If 16-bit Color APIs Are Present .....	14
5.2.2 Foreground, Background, and Text Colors .....	14
5.2.3 Pixel Reading and Writing .....	15
5.2.4 Direct Color Bitmaps .....	16
5.2.5 Special Drawing Modes .....	17
<b>6. Battery .....</b>	<b>18</b>
6.1 Battery Types .....	18
6.2 Battery Voltage Thresholds .....	18
<b>7. Springboard Expansion .....</b>	<b>21</b>
7.1 Springboard Version .....	21

7.2	Expansion On Visor Edge.....	21
7.2.1	Springboard Expansion Slot On Visor Edge .....	21
7.2.2	The Edge Connector On Visor Edge.....	22
7.3	Characterizing Springboard.....	27
7.3.1	Module Memory Space In Visor/Visor Deluxe .....	27
7.3.2	Measuring Springboard Performance .....	27
7.3.3	Interrupt Latency.....	27
7.3.4	Address Bus .....	28
7.3.5	Microphone .....	28
7.3.6	Power Supply (VCC).....	29
7.3.7	Battery Input Voltage .....	29
<b>8.</b>	<b>Cradle Interface .....</b>	<b>30</b>
8.1	Physical Cradle Design.....	30
8.1.1	Handspring Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro .....	30
8.1.2	Handspring Visor Prism.....	31
8.1.3	Handspring Visor Edge .....	31
8.2	Cradle Connector .....	32
8.2.1	Handspring Visor, Visor Solo, Visor Deluxe, Visor Platinum, Visor Neo, Visor Pro and Visor Prism .....	32
8.2.2	Visor Edge .....	33
8.3	General Cradle Connector Functional Description.....	33
8.3.1	Serial Communications on the Cradle Interface .....	34
8.4	Cradle Interface Pinout.....	36
8.4.1	Visor, Visor Deluxe, Visor Platinum, Visor Neo, Visor Pro and Visor Prism .....	36
8.4.2	Visor Edge .....	37
8.4.3	Cradle Interface Signal Descriptions.....	38
8.4.4	Charging Springboard Modules .....	39
<b>9.</b>	<b>Mechanical Information .....</b>	<b>40</b>
9.1	Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro.....	41
9.2	Visor Prism Mechanicals .....	42
9.3	Visor Edge Mechanicals .....	43
9.4	Springboard Modules .....	44
9.5	Cradle Base.....	45
9.5.1	Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro Cradle.....	45
9.5.2	Visor Prism Cradle .....	46
9.5.3	Visor Edge Cradle.....	47
9.6	Detachable Springboard Slot (Visor Edge Only) .....	48

9.7	Non-Encroachment Zones and Backward Compatibility.....	49
<b>10.</b>	<b>Environmental Test Specifications .....</b>	<b>50</b>
<b>11.</b>	<b>Handspring Developer Agreement.....</b>	<b>52</b>

## 1. Product Overview

---

### 1.1 Product Introduction History

#### 1.1.1 Visor Solo, Visor and Visor Deluxe

In Fall 1999 Handspring introduced Visor, Visor Solo and Visor Deluxe. Visor contains 2MB of DRAM and is available in Graphite only. Visor Deluxe contains 8MB of DRAM and is available in Graphite, Blue, Ice, Orange and Green. Both Visor and Visor Deluxe include a USB cradle. Visor Solo is Visor (2MB) without a cradle.

Visor Deluxe



#### 1.1.2 Visor Platinum and Visor Prism

In Fall 2000, Handspring introduced Visor Prism and Visor Platinum. These handhelds were the first to use the DragonBall VZ processor. The Visor Prism was the first Palm OS handheld to support 16-bit color.

Visor Platinum



Visor Prism



### 1.1.3 Visor Edge

In Spring 2001, Handspring introduced Visor Edge. This is Handspring's first ultra-slim handheld.

Visor Edge



### 1.1.4 Visor Neo and Visor Pro

In Fall 2001, Handspring introduced Visor Neo and Visor Pro. Visor Neo is an affordable, fast Visor handheld. Visor Pro comes with 16MB of RAM and a rechargeable battery.

Visor Neo



Visor Pro





## 2. CPU

### 2.1 CPU In Visor Handhelds

All Visor handhelds utilize the Motorola DragonBall line of processors.

Handheld	CPU Specification
Visor (and Visor Solo)	Motorola MC68EZ328 DragonBall-EZ processor, 16.58MHz
Visor Deluxe	Motorola MC68EZ328 DragonBall-EZ processor, 16.58MHz
Visor Platinum	Motorola MC68VZ328 DragonBall-VZ processor, 33MHz
Visor Prism	Motorola MC68VZ328 DragonBall-VZ processor, 33MHz
Visor Edge	Motorola MC68VZ328 DragonBall-VZ processor, 33MHz
Visor Neo	Motorola MC68VZ328 DragonBall-VZ processor, 33MHz
Visor Pro	Motorola MC68VZ328 DragonBall-VZ processor, 33MHz

### 2.2 CPU Reference Information

Complete documentation for the Motorola DragonBall CPUs is located on the Motorola website ([www.motorola.com](http://www.motorola.com)). The processor uses a 68000 core. Information on the DragonBall features is listed under the processor names listed above. Information related to the 68000 core (e.g. instruction details) may be listed separately.

### 2.3 Addressing Differences Between the EZ and VZ Processors

Internal address decoding on Handspring handhelds varies depending on the processor used. DragonBall EZ processors ignore the most significant three address bits in an internal 32-bit address. Therefore, addresses that are invalid may not generate a bus error even if the 32-bit address maps to a region of memory that doesn't correspond to a programmed chip select. The DragonBall VZ processor (as implemented in Handspring handhelds) does not ignore the three most significant address bits.

The Palm OS will sometimes execute application code without access to global variables. For example, if an application is configured to execute following a reset, global variables that are normally referenced with the A5 address register, are not available. To prevent illegal accesses, the operating system may set the high address bit of the A5 register in an attempt to generate a bus error should the application access global data that isn't available. In DragonBall EZ based systems, this may not cause the desired error and may generate invalid data instead. A DragonBall VZ based system will generate a bus error. To the end customer, this will appear as a Fatal Error dialog that only occurs on DragonBall VZ based handhelds.

### 2.4 Springboard Timing

To accommodate the faster DragonBall VZ processor, the Springboard bus timing specification was updated in revision 1.1 of the Springboard Development Guide. The latest version of this guide is located on the Handspring website:

<http://www.handspring.com/developers/documentation.jhtml>

### 2.4.1 CPU Wait States and Access Time

The DragonBall EZ processor can be configured to insert 0 to 6 wait states (60ns each on a 16.58MHz system) when accessing the Springboard Expansion Slot. The DragonBall VZ processor can be configured to insert 0 to 13 wait states (30ns each on a 33MHz system).

The system initially sets the slowest access speed possible to read the header information on the Springboard module. The Access Time supplied in the header is then used to calculate the appropriate number of wait states needed. The Access Time can be supplied in nanoseconds as a ROM token. The Palm-MakeROM utility is used to generate this token. More information on the Palm-MakeROM utility can be found in the Handspring Development Tools Guide located here:

<http://www.handspring.com/developers/documentation.jhtml>

An access time can also be dynamically read and set through the Handspring API HsCardAttrSet and HsCardAttrGet. These APIs are fully documented in the Springboard Development Guide located here:

<http://www.handspring.com/developers/documentation.jhtml>

## 3. Memory: RAM and ROM

### 3.1 RAM

The RAM for each handheld is given in the table below. The allocated dynamic RAM (dynamic heap) is also provided. For a detailed description of dynamic memory in the Palm OS architecture please refer to the Palm OS documentation.

### 3.2 ROM

Visor handhelds use Masked-ROM, not Flash. They are upgradeable through patches, but the ROM as a whole is not upgradeable. There is no support for a unique ID (i.e. serial number) assigned to each handheld.

Handheld	RAM	Dynamic RAM Allocation	ROM
Visor (and Visor Solo)	2MB RAM	~256KB	2MB <sup>1</sup>
Visor Deluxe	8MB RAM	~256KB	2MB <sup>1</sup>
Visor Platinum	8MB RAM	~256KB	2MB <sup>1</sup>
Visor Prism	8MB RAM	~256KB	2MB <sup>1</sup>
Visor Edge	8MB RAM	~256KB	2MB <sup>1</sup>
Visor Neo	8MB RAM	~256KB	2MB <sup>1</sup>
Visor Pro	16MB RAM	~256KB	2MB <sup>1</sup>

1. Japanese and multi-lingual systems contain 4MB ROMs.

#### 3.2.1 Tips For Working Without A Serial Number

If the intention is to provide copy-protection security for software on a Springboard module, each module can be encoded with a module and manufacturer ID. These are created by the Palm-MakeROM utility that takes your software and creates a ROM image file (.bin file). The module and manufacturer IDs are stored as tokens in the ROM header. These can be later retrieved by your software using the MemCardInfo() function. The token cannot be modified by a user application like FileMover. Modules (the 8MB Flash Module, or your production module) are either programmed with the Palm-Debugger (for quick testing) or our CardUpdaterMaker SDK (for distributing user updates).

Another alternative that some developers have used is to key off the HotSync name. While this does not provide ideal security, it does provide protection against “casual” software theft.

## 4. Operating System

**Operating System Revision By Handheld**

<b>Handheld</b>	<b>Operating System Revision</b>
Visor (and Visor Solo)	Palm OS 3.1H
Visor Deluxe	Palm OS 3.1H
Visor Platinum	Palm OS 3.5.2H
Visor Prism	Palm OS 3.5.2H
Visor Edge	Palm OS 3.5.2H2
Visor Neo	Palm OS 3.5.2H3
Visor Pro	Palm OS 3.5.2H3

### 4.1 Software in ROM

**Major Software In ROM By Handheld**

<b>Handheld</b>	<b>Software In ROM</b>
Visor (and Visor Solo)	Standard Palm OS applications, plus DateBook+, Advanced Calculator, CityTime, MathLib, and Remote UI
Visor Deluxe	Standard Palm OS applications, plus DateBook+, Advanced Calculator, CityTime, MathLib, and Remote UI
Visor Platinum	Standard Palm OS applications, plus DateBook+, Advanced Calculator, CityTime, MathLib, and Remote UI
Visor Prism	Standard Palm OS applications, plus DateBook+, Advanced Calculator, CityTime, MathLib, and Remote UI
Visor Edge	Standard Palm OS applications plus DateBook+, Advanced Calculator, CityTime, MathLib, Remote UI, Fast Lookup, Silent Alarms, "Dialing" Address Book
Visor Neo	Standard Palm OS applications plus DateBook+, Advanced Calculator, CityTime, MathLib, Remote UI, Fast Lookup, Silent Alarms, "Dialing" Address Book
Visor Pro	Standard Palm OS applications plus DateBook+, Advanced Calculator, CityTime, MathLib, Remote UI, Fast Lookup, Silent Alarms, "Dialing" Address Book

## 5. Display and Display Controller

**Display Information By Handheld**

Handheld	Display	Display Controller
Visor (and Visor Solo)	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>
Visor Deluxe	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>
Visor Platinum	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>
Visor Prism	160x160 resolution, TFT, 2.2" x 2.2" (3.1" diagonal)	Epson SED1376. Supports up to 16-bit per pixel color.
Visor Edge	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>
Visor Neo	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>
Visor Pro	160x160 resolution, 2.2" x 2.2" (3.1" diagonal)	Embedded LCD Controller on the DragonBall CPU. Supports 16 levels of gray. <sup>1</sup>

1. Grayscale support is a combination of hardware and software support. While the hardware may support grayscale, certain revisions of the Palm OS may not. The Palm Knowledge Base contains detailed information on grayscale support within the operating system.

### 5.1 Grayscale Support

16 level (4-bit) grayscale is supported in Handspring devices with Palm OS 3.5.2H and higher. Grayscale support in Visor and Visor Deluxe systems with Palm OS 3.1H is more complex.

There are several articles in the Palm Knowledge Base entitled "Seeing Gray" and "3.0 Grayscale Overview" covering this issue:

<http://www.palmos.com/dev/tech/support/>

On Handspring Visor and Visor Deluxe handhelds, the LCD controller's refresh rate is set differently depending on whether you're in grayscale mode or in black and white mode. The refresh rate is faster in grayscale mode. In general, here are the effects of the refresh rate on what is being displayed:

- The first characteristic is that at the faster refresh rate, you need a higher contrast level. For example, if you turn up the contrast manually, the screen will look normal.
- If you're running at a high refresh rate, black and white screens looks "streaky" (you see ghosting for vertical lines). The CPU also runs a little slower, since the LCD controller takes cycles from the memory bus more often.
- If you're running at a low refresh rate, you see flickering in grayscale mode.

Visor and Visor Deluxe runs in high-refresh rate mode in grayscale but low-refresh rate mode in black and white. When you switch from black and white to grayscale, the OS moves the contrast for you.

Some potential problems:

- Setting the CPU registers directly will cause problems. Use `ScrDisplayMode()`, or equivalent API, if possible.
- Trying to save and restore the contrast to work around the fact that the Palm V always resets the contrast when you switch to grayscale mode also causes problems. This is because Visor and Visor Deluxe handhelds not only preserves the contrast, it actually adjusts it for you. This is one reason you might get the washed out look.

## 5.2 16-Bit Color Support

Applications using the Palm OS 3.5 color APIs will work on Visor Prism without modification. A color depth of up to 8 bits per pixel is supported through an index mechanism. A full description of the associated functions and data structures is documented in the Palm OS 3.5 development guides.

An index is used to reference an 8-bit pixel value to an RGB color that that will be displayed on the LCD. By contrast, 16-bit color support is implemented using a DirectColor mechanism. A 16-bit RGB value is used directly, rather than referenced through an index table, to provide pixel color information. Developers that wish to take advantage of the DirectColor will need to use the new API calls described in this document.

### 5.2.1 Determining If 16-bit Color APIs Are Present

The API calls documented here are only necessary for applications that wish to take full advantage of the expanded range of colors available with a direct color display. Applications written to use the base OS 3.5 color API calls will continue to work without modification on systems with direct color displays.

To determine if the new API calls for direct color are available, an application should check to see if the `WinSetForeColorRGB()` system call is implemented. If this call is available, then *all* of the direct color API calls documented here are available:

```
directColorAvailable = false;

FtrGet (sysFtrCreator, sysFtrNumROMVersion, &version);
if (version > 0x03500000)
{
    void* procP;

    // See if it supports 16-bit color
    procP = SysGetTrapAddress (sysTrapWinSetForeColorRGB);
    if (procP && procP != SysGetTrapAddress (sysTrapSysUnimplemented))
        directColorAvailable = true;
}
```

### 5.2.2 Foreground, Background, and Text Colors

The base Palm OS 3.5 API calls for setting the foreground, background, and text colors are designed to take an index value as the color parameter. The index value is basically an index into a color lookup table where each entry in the table specifies the red, green, and blue components of the color. In order to set a certain drawing color for example, an application first has to lookup the index of that color in the color lookup table (using `WinRGBToIndex`) before passing that index value to `WinSetForeColor()`.

Displays that support 1, 2, 4, or 8 bits per pixel rely on a color lookup table in the display hardware in order to map pixel values into actual colors and the only colors that can be displayed on the screen at any given time are

those that are found in the display's color lookup table. Thus, the indexed form of the set color API calls covers the entire range of available colors.

Direct color displays on the other hand, do not rely on a color lookup table because the value stored into each pixel location specifies the amount of red, green, and blue components directly. For example, a 16-bit direct color display could have 5 bits of each pixel assigned as the red component, 6 bits as the green component, and 5 bits as the blue component. With this type of display, the application is no longer limited to drawing with a color that is in a color lookup table.

The base indexed mode calls for setting the foreground, background, and text colors continue to work even with direct color displays because *the system uses a translation table for mapping color index values into actual colors*. This translation table is the color lookup table of the destination bitmap. If the destination bitmap does not include a color lookup table (the more common case), then the color lookup table of the screen itself is used. The screen's color lookup table contains the system's default palette of colors, but it can be changed through the `WinPalette()` call.

When the screen is configured as a direct color display, the system gives it an 8-bit color table, providing 256 possible indexed colors (the maximum possible for indexed modes). Note that this color lookup table for the screen is present only for compatibility with the indexed mode color calls and has no effect on the actual display hardware itself since the display hardware derives the color from the actual red, green, and blue bits stored in each pixel location of the frame buffer.

Even though the indexed mode calls continue to work with direct color displays, new forms of the calls must be introduced to make the entire range of direct colors available to an application. The new forms of these calls take an `RGBColorType` parameter that specifies the exact amount of red, green, and blue to use and thus are not limited to colors in the destination bitmap's color lookup table.

The prototypes of these calls are shown below. For each call, `newRgbP` is the new color to use and the previous color is returned *in* `prevRgbP`. These new calls are more generic than the older indexed forms and can be used with both indexed (1, 2, 4, or 8 bit) or direct color displays (the system will automatically look up the color index value of the closest color for you, if necessary).

```
WinSetForeColorRGB (const RGBColorType* newRgbP,
                    RGBColorType* prevRgbP);
WinSetBackColorRGB (const RGBColorType* newRgbP,
                    RGBColorType* prevRgbP);
WinSetTextColorRGB (const RGBColorType* newRgbP,
                    RGBColorType* prevRgbP);
```

Because these new RGB forms of the calls are only available on systems with the direct color enhancements present, applications should generally stick to using the older indexed form of these calls (`WinSetForeColor()`, `WinSetBackColor()`, `WinSetTextColor()`) unless they need finer control over the choice and dynamic range of colors.

### 5.2.3 Pixel Reading and Writing

The base OS 3.5 API call for reading a pixel value, `WinGetPixel()`, is designed to return a color index value (`IndexedColorType`). When this call is performed on a direct color display, it must first get the actual pixel value (a 16 or 24 bit direct color value) and then look up the closest color from the bitmap's color lookup table and return the index of the closest color from that table. If the bitmap does not include a color table (the common case), then the 256-entry color table for the screen itself is referenced. This mode of operation ensures compatibility for applications that will in turn take the return value from `WinGetPixel()` and use it as an indexed color to `WinSetForeColor()`, `WinSetBackColor()`, etc.

If the intent of the application is to copy pixels exactly from one bitmap to another, the application will experience a loss of color accuracy on a direct color display because of the closest-match color table lookup operation that `WinGetPixel()` performs.

To avoid this potential loss of color accuracy with direct color displays, applications can instead use the new `WinGetPixelRGB()` call. This call returns the pixel as an `RGBColorType` with a full 8 bits each of red, green, and blue ensuring no loss of color resolution. This new call is more generic than the base `WinGetPixel()` call, and can be used with both indexed (1, 2, 4, or 8 bit) or direct color displays (the system will automatically look up the RGB components of indexed color pixels as necessary). The prototype of this function is:

```
WinGetPixelRGB (Coord x, Coord y, RGBColorType* rgbP);
```

The pixel setting API calls (`WinPaintPixel()`, `WinDrawPixel()`, etc.) all rely on using the current foreground and background colors and don't require new forms for direct color displays. An application can simply pass in the return `RGBColorType` from `WinGetPixelRGB()` to `WinSetForeColorRGB()` and then call `WinDrawPixel()` in order to copy a direct color pixel.

### 5.2.4 Direct Color Bitmaps

The new Window and Blitter managers now support 16-bits per pixel direct color bitmaps, as well as the previously supported 1-, 2-, 4-, and 8-bit indexed color bitmaps. The format and version of the `BitmapType` structure is still 2, but a new `directColor` flag is defined for the flags field. This bit, when set, indicates that the pixels in the bitmap are direct color pixels. In addition to this flag, a direct color bitmap must also include the following four fields. If the bitmap also includes a color lookup table, then these fields follow the color lookup table; otherwise they immediately follow the bitmap structure itself.

```
typedef struct BitmapDirectInfoType
{
    UInt8      redBits;
    UInt8      greenBits;
    UInt8      blueBits;
    UInt8      reserved;    // <- must be zero
    RGBColorType transparentColor;
}
BitmapDirectInfoType;
```

The `redBits`, `greenBits`, and `blueBits` fields indicate the number of bits in each pixel for each color component. The current implementation only supports 16-bits per pixel bitmaps with 5 bits of red, 6 bits of green, and 5 bits of blue. This type of 16-bit direct color pixel is laid out like this:

```

R R R R  R G G G  G G G B  B B B B
MSB                                LSB
```

The `transparentColor` field contains the red, green, and blue components of the transparent color of the bitmap. For direct color bitmaps, this field is used instead of the `transparentIndex` field to designate the transparent color value of the bitmap, since the `transparentIndex` field is only 8 bits wide and can only represent an indexed color. The `transparentColor` field, like the `transparentIndex` field, is ignored unless the `hasTransparency` bit is set in the bitmap's flags field.

It is important to note that as long as the new version 3 Window manager is present, a 16-bit direct color bitmap can always be rendered, *regardless* of the actual screen depth. The color APIs will automatically perform the necessary bit depth conversion to render the bitmap into whatever the depth of the destination.



Bitmap resources can be built to contain multiple depth images in the same bitmap resource - up to one per each possible depth. A potential incompatibility could arise if an application includes only a direct color version of a bitmap, though. Trying to draw a direct color bitmap with an older version of the Blitter manager will cause the system to crash. Consequently, applications need to either check that version 3 of the Window manager is present before drawing a direct color bitmap, or they must always include a 1, 2, 4, or 8 bit per pixel image of the bitmap in the bitmap resource along with the direct color version.

### 5.2.5 Special Drawing Modes

The special drawing modes of `winErase`, `winMask`, `winInvert`, and `winOverlay` introduce a complication when it comes to direct color models. These drawing modes were originally conceived of for use with monochrome bitmaps in which black is designated by 1 bits and white is designated by 0 bits. With these "color" assignments, these various modes can be described as:

- `winErase` becomes an AND operation (black pixels in the source leave the destination alone, whereas white pixels in the source make the destination white).
- `winMask` becomes an ANDNOT operation (black pixels in the source make the destination white, whereas white pixels leave the destination alone).
- `winInvert` becomes an XOR operation (black pixels in the source invert the destination, whereas white pixels leave the destination alone).
- `winOverlay` becomes an OR operation (black pixels in the source make the destination black, whereas white pixels in the source leave the destination alone).

In a direct color bitmap, black is designated by all 0's and white is designated by all 1's. Because of this, if all the drawing modes were implemented as logical operations in the same way they were for indexed color modes, the desired *effect* would not be achieved.

The assumption made by the direct color APIs is that the desired effect is more important to the caller than the actual logical operation that is performed. Thus, the various drawing modes, when drawing to a direct color bitmap, become:

- `winErase` becomes an OR operation (black pixels in the source leave the destination alone, whereas white pixels in the source make the destination white).
- `winMask` becomes an ORNOT operation (black pixels in the source make the destination white, whereas white pixels leave the destination alone).
- `winInvert` becomes an XORNOT operation (black pixels in the source invert the destination, whereas white pixels leave the destination alone).
- `winOverlay` becomes an AND operation (black pixels in the source make the destination black, whereas white pixels in the source leave the destination alone).

As long as the source and destination bitmaps contain only black and white colors, the new interpretations of the drawing modes in direct color modes will produce the same effects as they would with an indexed color mode. With non-black-and-white pixels, however, an application may get unexpected results from these drawing modes if they assume that the color APIs will perform the same logical operation in direct color mode as they do in indexed color mode.

## 6. Battery

---

### 6.1 Battery Types

For handhelds that use replaceable alkaline batteries, rechargeable batteries can be used but are subject to some disadvantages. The internal resistance characteristics of rechargeable batteries are such that as the battery becomes depleted, the battery voltage will drop at a much faster rate than alkaline technology. The voltage levels that the system uses to detect low battery conditions are designed to accommodate a voltage curve that reflects alkaline batteries and not rechargeable batteries. The consequence is that user warnings may happen very rapidly if at all. Some users report good results with rechargeable batteries, but customers should be advised that the system is not optimized for this type of battery technology.

The Palm OS attempts to accommodate rechargeable batteries with a “shortcut” that indicates the battery type to the OS (shortcut.7). However, Palm documentation also warns that this has limited effectiveness. More information on “shortcuts” and the battery type shortcut can be found in the Palm Knowledge Base ([www.palmos.com](http://www.palmos.com)).

Handheld devices that are designed for rechargeable batteries (e.g. Visor Prism, Visor Edge and Visor Pro) have wider voltage detection tolerances near low battery conditions to accommodate the steeper voltage decline.

**Battery Type By Handheld**

Handheld	Battery
Visor (and Visor Solo)	Two AAA Alkaline batteries.
Visor Deluxe	Two AAA Alkaline batteries.
Visor Platinum	Two AAA Alkaline batteries.
Visor Prism	Li-Ion rechargeable 1550mAh (non-removable).
Visor Edge	Li-Ion rechargeable 490mAh (non-removable).
Visor Neo	Two AAA Alkaline batteries.
Visor Pro	Li-Ion rechargeable 900mAh (non-removable).

### 6.2 Battery Voltage Thresholds

Developers have access to battery level thresholds through the `SysBatteryInfo()` function. These thresholds provide battery voltage levels where the system will begin to perform various actions to warn the user of a low battery condition and eventually shut down the system to protect data.

In general, applications should use the percentages returned from `SysBatteryInfo()` rather than actual voltage levels. Percentages will tend to be more accurate since the current battery voltage will change with the current load due to internal battery resistance. Also, voltage references are likely to change between handheld models (which may use different batteries) and battery type (e.g. alkaline versus rechargeable).

The Palm OS specifies two thresholds that can be retrieved through the `SysBatteryInfo()` function; “warning threshold” and “critical threshold”.

**Warning Thresholds:** When the battery drops below this level, the system puts a `lowBatteryChr` key event into the event queue. Applications call `SysHandleEvent()` in their event loop that in turn calls

SysBatteryDialog( ) in response to this event. There are two levels. The system will begin with “mild” warnings that will become “severe” (different message and more persistent) as the voltage drops.

**Critical Thresholds:** Critical Thresholds are provided to protect the user against data loss. When the battery drops below the *Software* Critical Threshold, the operating system turns the handheld off without warning to protect user data. The operating system doesn’t allow the user interface to turn back on until the battery rises above the Wakeup Threshold to prevent the system from falsely detecting a usable battery level due to the reduced load.

When the *Hardware* Critical Threshold is reached, the LOWBAT\* signal on the Springboard Expansion Slot is asserted. However, 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.

In general, developers should check that enough power is available before putting a Springboard module into a high power state. For example, checking battery levels prior to programming a flash ROM may avoid failures due to insufficient battery power. The table below summarizes various battery thresholds and the resulting system action.

**Power Thresholds And System Behavior**

Threshold	System Action
“Mild” Warning Threshold	The operating system begins to warn the user that batteries are low. These warning begin at 20% of remaining capacity.
“Severe” Warning Threshold	More persistent warning messages from the operating system. These warnings begin at 10% of remaining capacity.
Critical Threshold – Software	The operating system will put the system into sleep mode to protect user data.
Wakeup Threshold	Once the operating system has been placed in sleep mode due to a critical low battery, it will not power on the user interface until the battery is above this level.
Critical Threshold - Hardware	A hardware comparator will briefly assert LOWBAT* then remove power to the Springboard Slot.

For reference, the following battery voltages apply to hardware threshold levels.

**Voltage Level Thresholds By Handheld**

<b>Handheld</b>	<b>Battery At Capacity</b>	<b>Critical Threshold - Hardware</b>
Visor (and Visor Solo)	3.0V	1.6V
Visor Deluxe	3.0V	1.6V
Visor Platinum	3.0V	1.6V
Visor Prism	4.2V	3.5V
Visor Edge	4.2V	3.5V
Visor Neo	3.0V	1.6V
Visor Pro	4.2V	3.5V

## 7. Springboard Expansion

### Expansion Summary By Handheld

Handheld	Expansion
Visor (and Visor Solo)	Built-in Springboard Expansion Slot.
Visor Deluxe	Built-in Springboard Expansion Slot.
Visor Platinum	Built-in Springboard Expansion Slot.
Visor Prism	Built-in Springboard Expansion Slot.
Visor Edge	A detachable Springboard Slot allows the use of Springboard modules. The native connector (Edge Connector) is optimized for ultra thin form-factor of Visor Edge.
Visor Neo	Built-in Springboard Expansion Slot.
Visor Pro	Built-in Springboard Expansion Slot.

### 7.1 Springboard Version

The history of modifications and enhancements to the Springboard specification is fully documented in the Springboard Development Guide located here:

<http://www.handspring.com/developers/documentation.jhtml>

### Springboard Revision By Handheld

Handheld	Springboard Revision
Visor (and Visor Solo)	Springboard 1.1
Visor Deluxe	Springboard 1.1
Visor Platinum	Springboard 1.1
Visor Prism	Springboard 1.1
Visor Edge	Springboard 1.1
Visor Neo	Springboard 1.1
Visor Pro	Springboard 1.1

### 7.2 Expansion On Visor Edge

#### 7.2.1 Springboard Expansion Slot On Visor Edge

The native Edge Connector on Visor Edge is optimized for the product's thin form-factor. This connector is *not* part of the Springboard specification and is detailed separately in this guide.

The Detachable Springboard Slot allows the use of Springboard modules with Visor Edge. The specifications provided in this “*Implementation Details*” section covers the behavior of Springboard modules used with the Detachable Springboard Slot.

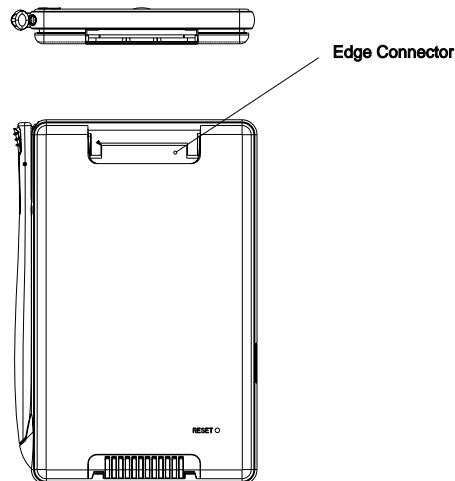
## Detachable Springboard Slot



### 7.2.2 The Edge Connector On Visor Edge

This section provides detailed specifications for the Edge Connector.

The Springboard Expansion Slot is a Handspring specification that is designed to be a consistent platform from product to product. **The Edge Connector design is unique to Visor Edge and is not part of the Springboard specification. Springboard compatibility from an electrical, software (plug and play architecture) and mechanical standpoint is maintained through the Detachable Springboard Slot.**



This pinout is *not* pin compatible with the Springboard Expansion Slot. Refer to the Springboard Development Guide for the Springboard pinout. The table below summarizes the signal names with their respective pin numbers on the 68-pin Edge Connector. The signal direction (I/O/P/PU<sup>1</sup>) is viewed with reference to the module. For example, CS1\* is driven by the CPU and is an input (I) to the module.

### Edge Connector Pin Summary (Visor Edge Only)

Pin	Name	I/O/P/PU <sup>1</sup>	Function	Pin	Name	I/O/P/PU	Function
1	GND	P	Module Ground	35	GND	P	Module Ground
2	CD2*	O/PU	Card Detect	36	VCC	P	Module VCC
3	D7	I/O	Data Bus	37	VCC	P	Module VCC
4	D6	I/O	Data Bus	38	D15	I/O	Data Bus
5	D5	I/O	Data Bus	39	D14	I/O	Data Bus
6	D4	I/O	Data Bus	40	D13	I/O	Data Bus
7	D3	I/O	Data Bus	41	D12	I/O	Data Bus
8	D2	I/O	Data Bus	42	D11	I/O	Data Bus
9	D1	I/O	Data Bus	43	D10	I/O	Data Bus
10	D0	I/O	Data Bus	44	D9	I/O	Data Bus
11	A0	I	Address Bus	45	D8	I/O	Data Bus
12	A1	I	Address Bus	46	RXD2	O	Receive Data 2 <sup>3</sup>
13	A2	I	Address Bus	47	TXD2	I	Transmit Data 2 <sup>3</sup>
14	A3	I	Address Bus	48	CTS2*	O	Clear To Send 2 <sup>3</sup>
15	A4	I	Address Bus	49	RTS2*	I	Request To Send 2 <sup>3</sup>
16	A5	I	Address Bus	50	A8	I	Address Bus
17	A6	I	Address Bus	51	A9	I	Address Bus
18	A7	I	Address Bus	52	A10	I	Address Bus
19	CS0*	I	Chip Select	53	A11	I	Address Bus
20	CS1*	I	Chip Select	54	A12	I	Address Bus
21	OE*	I	Output Enable	55	A13	I	Address Bus
22	RESET*	I	Module Reset	56	A14	I	Address Bus
23	WE*	I	Write Enable	57	A15	I	Address Bus
24	LOWBAT*	I	Low Battery	58	A16	I	Address Bus
25	IRQ*	O/PU	Interrupt Request	59	A17	I	Address Bus
26	PASSTHROUGH4	I/O	Pass Through 4	60	A18	I	Address Bus
27	PASSTHROUGH3	I/O	Pass Through 3	61	A19	I	Address Bus
28	PASSTHROUGH2	I/O	Pass Through 2	62	A20	I	Address Bus
29	PASSTHROUGH1	I/O	Pass Through 1	63	A21	I	Address Bus
30	MIC+	I	Microphone	64	A22	I	Address Bus
31	MIC-	I	Microphone	65	A23	I	Address Bus
32	VDOCK	P	Docking Voltage	66	VDOCK	P	Docking Voltage
33	CD1*	O/PU	Card Detect	67	Reserved		Reserved
34	GND	P	Module Ground	68	GND	P	Module Ground

1. I = input, O = output, and P = power, with respect to the module. For example, the IRQ signal is driven by the module and is an output to the handheld. PU indicates the signal is internally pulled up within the handheld.
2. Indicates an active low signal.
3. These signals are directly connected to the second UART on the DragonBall VZ processor and are *not* buffered.

### 7.2.2.1 Signal Descriptions (Visor Edge Only)

The signals for the Edge Connector are described below in alphabetical order. Active-low signals are denoted with an asterisk (“\*”).

#### Address Bus A[23:0]

Each of the two chip selects (CS0 and CS1) has direct access of up to 16MB. A total range of 32MB is addressable. Address line A23 is the most significant address bit, and A0 is the least significant address bit. The default size of each region is 16MB and is software programmable. This bus is an output from the handheld; it is always driven during normal and sleep mode. The address bus is valid throughout the entire bus cycle. When using the Detachable Springboard Slot, note that the Springboard specification defines the data bus as 16-bit only and memory accesses are conducted on even-byte boundaries. Note that A0 is inactive.

#### Card Detect CD1\*, CD2\*

CD1\* and CD2\* are active-low module detect signals that indicate to the handheld when an expansion module is in place. The two Card Detect pins on the Edge Connector are physically shorter than all the other pins on the expansion connector. This is intentional to assure a firmly seated connection prior to the assertion of Card Detect by the module.

On the handheld side, the signals perform two functions: 1) they interrupt the handheld to alert the CPU that a module has been inserted or removed, and, 2) they begin turning on the Vcc power supply. Both signals should be tied directly to GND on the expansion module.

Note that both Card Detect signals are tied through an AND gate. Both signals must be asserted for the handheld to detect the module.

#### Chip Select CS0\*, CS1\*

These two active-low chip select signals control access to the two addressable regions on a module. The address space for CS0\* is referred to as csSlot0; the address space for CS1\* is referred to as csSlot1. In order for the Palm OS to recognize the module and its contents, use CS0\* to access ROM or Flash. CS1\* is optional and can be used to interface with additional ROM, Flash, UARTs, or other peripheral devices. Both chip select signals are asserted for the duration of the memory cycle. Only one of the two chip selects is valid for each module access. Refer to the *Springboard Development Guide* for more information on the chip select signals and their corresponding address spaces.

#### Clear To Send 2 CTS2\*

CTS2 is connected directly to the second UART on the DragonBall VZ processor. It is not buffered and appropriate care should be taken when using this signal. This signal is not present in the Detachable Springboard Expansion Slot.

#### Data Bus D[15:0]

The data bus consists of 16 data lines, D[15:0]. D15 is the most significant data bit, and D0 is the least significant data bit. Only 16-bit operations are performed on the data bus.

#### Module Ground GND

GND is the ground connection to a module. All GND signals must be connected to the module's ground reference or plane.

#### Interrupt Request IRQ\*

The active low interrupt request is level-sensitive. This signal is output from a module whenever interrupt service is required from the handheld computer. There is no default interrupt service routine for the module, so the application resident on the expansion module must install the interrupt service routine (ISR) during initialization.



Interrupt acknowledgment is user-defined and must be accommodated by the expansion module application as well. The handheld's internal interface has a pull-up resistor, so the module does not require one.

### Low Battery Warning

**LOWBAT\***

The low battery warning signal indicates that the handheld's batteries are below a critical threshold or are being swapped out. When this signal is asserted, the buffers driving the expansion slot are powered off to prevent data loss in the handheld. When the batteries are replaced or recharged, the system behaves as if the module has been "removed" and "re-inserted." Note that this signal is only asserted for several milliseconds before the buffers are powered off.

### Microphone

**MIC+,MIC-**

These two signals interface to the microphone on the handheld unit. These signals are a differential pair and are directly connected to an electret condenser microphone. Appropriate bias must be supplied by the module on the MIC+ signal. Details on the microphones within Handspring handheld computers are located in an application note on the Handspring website titled "AN-08: Visor Microphone."

### Output Enable

**OE\***

OE\* is the active-low output enable, or read signal, for the module. When qualified with a low on either CS0\* or CS1\*, a low on OE\* indicates a read cycle from the module. The address bus is valid before the assertion of OE\*. The module can drive the data bus as soon as a chip select and OE\* are asserted. Data must be driven for the entire cycle, as determined by the levels on the chip select and OE\*. WE\* is deasserted during read cycles. The chip selects are not guaranteed to be asserted prior to the assertion of OE\*. Also the cycle is ended when either a chip select or OE\* is deasserted.

### Module Reset

**RESET\***

RESET\* is an active-low signal for an expansion module. To maintain compatibility with Springboard modules in the Detachable Springboard Slot, RESET\* is asserted while VCC is rising during module insertion. The Springboard specification guarantees 25ms of power-on reset time for modules. This is derived from a VCC ramp up time of approximately 5ms and the assertion of RESET\* for over 30ms.

Application software can also assert this signal at any time to reset the module through the HsCardAttrSet ( ) API. RESET\* does *not* assert LOWBAT\* or remove power to VCC.

### Pass Through

**PASSTHROUGH [1:4]**

These pins are passed through the Cradle Interface to the Edge Connector. They are intended for future peripheral use and are presently *reserved*. These signals are not brought out in the Detachable Springboard Slot.

### Request To Send 2

**RTS2\***

RTS2 is driven directly from the second UART on the DragonBall VZ processor. It is not buffered and appropriate care should be taken when using this signal. This signal is not present in Detachable Springboard Expansion Slot.

### Receive Data 2

**RXD2**

RXD2 is connected directly to the second UART on the DragonBall VZ processor. It is not buffered and appropriate care should be taken when using this signal. This signal is not present in the Detachable Springboard Expansion Slot.

### Transmit Data 2

**TXD2**

TXD2 is driven directly from the second UART on the DragonBall VZ processor. It is not buffered and appropriate care should be taken when using this signal. This signal is not present in Detachable Springboard Expansion Slot.

## Module VCC

## VCC

VCC is the 3.0-3.6V power supply that can be used to power a module (some modules can supply their own power). To maintain compatibility with Springboard Modules in the Detachable Springboard Slot, the Springboard specifications for VCC are adhered to.

When using the Detachable Springboard Slot, power is not provided on these pins until the module is firmly seated and both Module Detects (CD1\* and CD2\*) are asserted. After insertion, the power supply ramps up to VCC in approximately 5 ms. The maximum current that can be supplied by the Module VCC is 100 mA. Expansion modules that use this power source (rather than a separate battery or line power source) should take into account User Interface issues when LOWBAT\* is asserted and the Springboard Expansion Slot power is removed.

## Docking Voltage

## VDOCK

To maintain compatibility with Springboard modules in the Detachable Springboard Slot, the Springboard VDOCK specification is maintained on Visor Edge. The handheld passes charging power from a pin on the Cradle Interface through two pins (VDOCK) on the Edge Connector. VDOCK provides 4.75 – 6.2V with a maximum current of 500mA. These pins provides charging power to a module when the handheld is placed into a Visor Edge cradle.

Note that Handspring handhelds are not designed to receive power from the module. Developers should ensure that VDOCK is not connected to VCC in an attempt to back power the handheld.

## Write Enable

## WE\*

WE\* is the active-low write enable signal for a module. When qualified with a low on either CS0\* or CS1\*, a low on WE\* indicates a write cycle to the module. The address bus is valid before the assertion of WE\*. The data bus, driven by the host interface, is valid before the assertion of WE\*. In addition, WE\* is deasserted prior to the deassertion of the chip select. OE\* is deasserted during write cycles. Since there is no separate write enable for each byte, all 16 bits of the data bus are written at the same time; thus, there is no support for byte writes to the expansion module.

## Reserved

Saved for future use. These pins must remain unconnected.

### 7.2.2.2 Serial Interface (Second UART on Visor Edge only)

There are two UARTs that are integrated on the DragonBall VZ processor. The first is used internally for serial communications through various ports (e.g. cradle interface, IR transceiver) as in other Handspring handheld computers. The UART is buffered by gates that drive the transmit and receive signals.

The second UART is provided only through the Edge Connector. It is not passed through the Detachable Springboard Slot. Unlike the internal UART, the second UART provides hardware handshaking (RTS/CTS) signaling. This second UART is not buffered and is therefore not 5V tolerant. The second UART signals are connected directly to the system CPU and care must be taken when exercising this hardware feature.

To access the second UART in software, use the creator code 'HSSS' when opening the port. This definition is included in the Handspring Extensions header file, HsCreators.h.

When working with the serial library for the second UART, there are a couple of points of interest:

- When the serial port is closed, the four pins are pulled up internally.
- When the handheld is in sleep mode, the RTS pin is de-asserted.
- The electrical details for this port are fully described in the Motorola DragonBall VZ documentation.

## 7.3 Characterizing Springboard

### 7.3.1 Module Memory Space In Visor/Visor Deluxe

Using `HsCardAttrSet()` and the `hsCardAttrCsSize` selector to set the chip select memory size to over 1.8MB does not work in Handspring Visor and Visor Deluxe. Generally, this does not cause a problem, since the size is typically not *changed* dynamically. By default, the memory space is configured as a contiguous space of two 16MB blocks (one per chip select) and the processor has full access to this entire memory space.

### 7.3.2 Measuring Springboard Performance

A full description of Springboard performance benchmarking is available as an application note (AN19: Measuring Springboard Performance) and is located on the Handspring website here:

[http://www.handspring.com/developers/tech\\_notes.jhtml](http://www.handspring.com/developers/tech_notes.jhtml)

**Springboard Throughput By Handheld**

	<b>120ns Springboard Access Time (Requested)</b>	<b>0ns Springboard Access Time (Requested)</b>
Visor/Visor Deluxe	~3.2Mbytes/sec	~3.5Mbytes/sec
Visor Prism	~4.0Mbytes/sec	~5.2Mbytes/sec
Visor Platinum	~4.0Mbytes/sec	~5.2Mbytes/sec
Visor Edge	~4.6Mbytes/sec	~6.4Mbytes/sec
Visor Neo	~4.1Mbytes/sec	~5.5Mbytes/sec
Visor Pro	~4.1Mbytes/sec	~5.5Mbytes/sec

### 7.3.3 Interrupt Latency

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 (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 (the `*sysAwakeP` parameter is true).

In the third case, the designer should consider adding an *additional* 20 milliseconds if the OS needs to open the USB library. Note that this case only happens if the USB library was open when the device went to sleep. This situation is rare as 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 like interrupt handlers. The table below summarizes interrupt latency from various system states.

Notice that later handhelds which utilize the DragonBall VZ processor requires a longer period to transition from asleep to awake. This tradeoff results in lower sleep current.

### Interrupt Latency By Handheld

Handheld	Device Awake	Device Asleep (*sysAwakeP == false)	Device Asleep (*sysAwakeP == true)
Visor (and Visor Solo)	0.047ms	1.2ms	3.1ms
Visor Deluxe	0.047ms	1.2ms	3.1ms
Visor Platinum	0.033ms	3.1ms	4.0ms
Visor Prism	0.041ms	3.1ms	4.8ms
Visor Edge	0.028ms	3.1ms	4.4ms
Visor Neo	0.038ms	3.1ms	4.4ms
Visor Pro	0.036ms	3.1ms	4.7ms

The recommended design times for interrupt latency for Springboard is in the table below.

### Recommended Design Times By Condition

Condition	Recommend Design Times - Springboard Maximum Latency
Device already awake	0.15 ms max
Device asleep (*sysAwakeP == false)	4 ms max
Device asleep (*sysAwakeP == true)	10 ms max

#### 7.3.4 Address Bus

In all Handspring handhelds, A0 is not used (pulled low). All memory accesses are 16 bits wide and conducted on even byte boundaries.

#### 7.3.5 Microphone

All Handspring handhelds use a microphone with a 2.2K  $\Omega$  impedance, a standard operating voltage of 2.0V, and a maximum current consumption of 0.5mA.

Full details on utilizing the Visor handheld microphone can be found in an application note (AN-08: Visor Microphone) located here:

[http://www.handspring.com/developers/tech\\_notes.jhtml](http://www.handspring.com/developers/tech_notes.jhtml)

### 7.3.6 Power Supply (VCC)

The Springboard specification for Vcc is 3.0V to 3.6V.

**System VCC By Handheld**

<b>Handheld</b>	<b>VCC For A Specific Handheld</b>
Visor (and Visor Solo)	3.3V
Visor Deluxe	3.3V
Visor Platinum	3.15V
Visor Prism	3.15V
Visor Edge	3.15V
Visor Neo	3.15V
Visor Pro	3.15V

### 7.3.7 Battery Input Voltage

This only applies to Handspring handheld devices that use replaceable batteries. The maximum voltage allowed through the battery contacts is 3.2V.



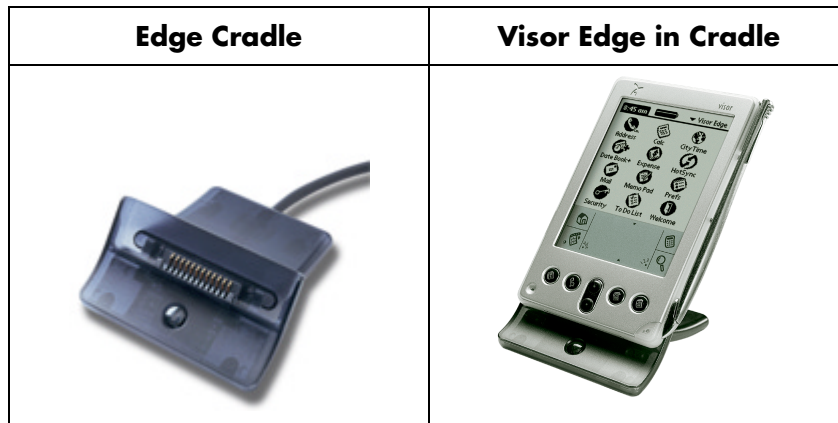
### 8.1.2 Handspring Visor Prism

The Handspring Visor Prism requires a slightly different cradle due to a thicker overall form-factor and curved back.



### 8.1.3 Handspring Visor Edge

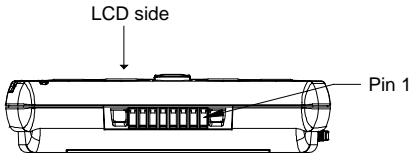
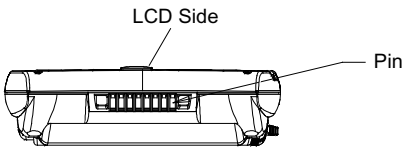
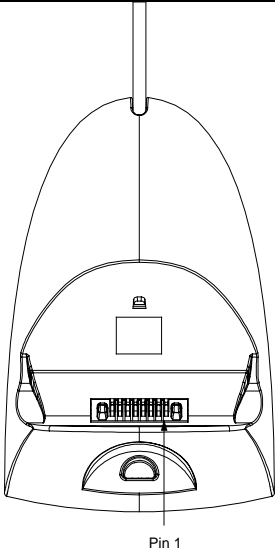
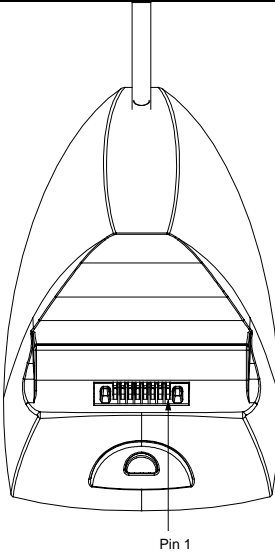
The Visor Edge requires a new cradle design that matches the overall industrial styling. A new cradle connector is also used on the Visor Edge.



## 8.2 Cradle Connector

### 8.2.1 Handspring Visor, Visor Solo, Visor Deluxe, Visor Platinum, Visor Neo, Visor Pro and Visor Prism

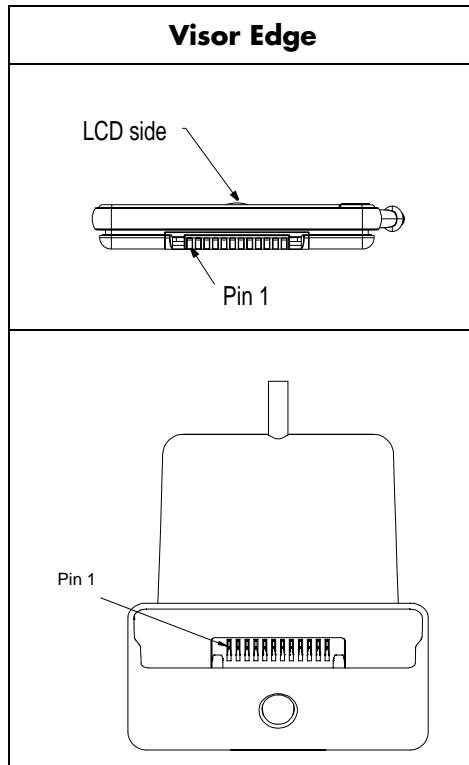
The Handspring Visor, Visor Solo, Visor Deluxe, Visor Platinum, Visor Neo, Visor Pro and Visor Prism share a common 8 pin cradle interface connector. The orientation of pin 1 is the same for all of these devices.

<b>Visor, Visor Solo, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro</b>	<b>Visor Prism</b>
	
	



### 8.2.2 Visor Edge

The Handspring Visor Edge uses a new cradle interface connector that supports additional pass-through signals.



## 8.3 General Cradle Connector Functional Description

The cradle connector provides serial and USB communication 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 or modem can also be interfaced to the handheld unit through this port.

The Cradle Connector contains a serial port (connected to a UART) and a USB bus. The high-speed USB interface is a slave-only peripheral interface between a host (PC or Mac) and the handheld device, and therefore can't communicate directly with other USB slave devices.

### 8.3.1 Serial Communications on the Cradle Interface

The serial interface on the Visor Edge handheld cradle connector is implemented with only TXD and RXD signals. To provide room for USB signals, there is no hardware handshaking provided. Additionally, the TXD and RXD signals implement TTL logic levels based on system VCC. 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. *The level-shifting circuitry is powered from the DTR and RTS lines on the desktop/laptop side of the interface.*

Here are a couple of tips that may help developers work with the serial port on the Cradle interface of Handspring handhelds.

1. Handspring devices have only RXD, TXD and GND serial signals on the Cradle Interface. Some devices require hardware handshaking to operate. On some Palm OS® devices, this is done through RTS/CTS signaling. When communicating between two PCs DTR/DSR signaling is sometimes used. It's useful to note that hardware handshaking should be turned off, and there is a possibility that some serial lines may need to be tied.
2. The serial signals on the Cradle Connector are at TTL voltage levels, not RS232. Some devices are OK with these levels. Others, like laptops and desktop PCs, are not. TTL logic levels are based on system voltage (VCC). For example, VCC on Visor and Visor Deluxe is 3.3V. The level shifting circuitry to communicate with RS232 based systems is done in the serial cradle. The circuit is powered from the DTR and RTS lines on the host side of the interface. Even though power is drawn from these signals, only the TXD/RXD signals are active. Some 3rd party vendors\*, have created the same functionality in a cable form-factor.
3. The handheld is configured as a DCE and the desktop/laptop is a DTE. This is the typical configuration based on a desktop/laptop connected to a handheld for HotSync™ purposes. The distinction is important when you want to communicate with another DCE (e.g. a modem) and a null modem adapter is required.
4. USB and Serial lines are not shared on the Cradle Interface. In fact, for debugging purposes, some developers have started with the serial cradle and wired in USB lines. The debugger can be forced to use USB, while applications continue to communicate through the serial port. The active-low KBD pin in the Cradle Interface helps the system determine which communications channel to use. When this pin is asserted (active-low, so it's tied to ground in the serial cradle), the handheld can determine that it should use serial rather than USB for applications such as HotSync.
5. An overview of data communications in Palm OS is provided in an application note on the Handspring web site. It's called "AN-09: Data Communications on the Springboard Platform." The overview is applicable to communications through the Cradle Interface, as well as, through the Springboard slot. Applications notes are located on the Development Kit: Application Notes, Presentations and FAQs page.  
([http://www.handspring.com/developers/tech\\_notes.jhtml](http://www.handspring.com/developers/tech_notes.jhtml))
6. Another possible area of conflict is what happens when a Handspring handheld is placed in the serial cradle. It automatically detects this state and starts up a keyboard daemon in the background that opens the serial port. This was done to facilitate keyboard support, but it gets in the way of other serial applications. Since the serial port is open, running a program that opens the serial port will fail with error 775 (port already open, 0x0307). The fix, which may be a bit inconvenient for developers, is to modify the serial application to check if the application is running on a Handspring handheld and, if so, call HsExtKeyboardEnable(false) to disable the keyboard daemon before opening the serial port. The serial port is then closed and the application is free to carry on as usual.

Example:

```
#include <HsExt.h>
```

```
...    // If on Handspring device, disable the keyboard
        // thread before opening the serial library.
        if (!FtrGet ('hsEx', 0, &ftrHSVersion))
            HsExtKeyboardEnable (false);
...
    (continue with SysLibFind(), SerOpen(), etc)
```

Note that there is no need to re-enable the keyboard support. It is automatically re-enabled each time it is placed into the cradle.

A simple workaround is to have the application open the serial port before it is placed in the cradle (e.g. start the application before putting it in the cradle). The reason a soft-reset in the cradle also works is because the keyboard daemon is started at the time the device is placed into the cradle, not just from being in the cradle. Since the state hasn't changed since boot, the daemon won't start.

\* These vendors can be reached at:

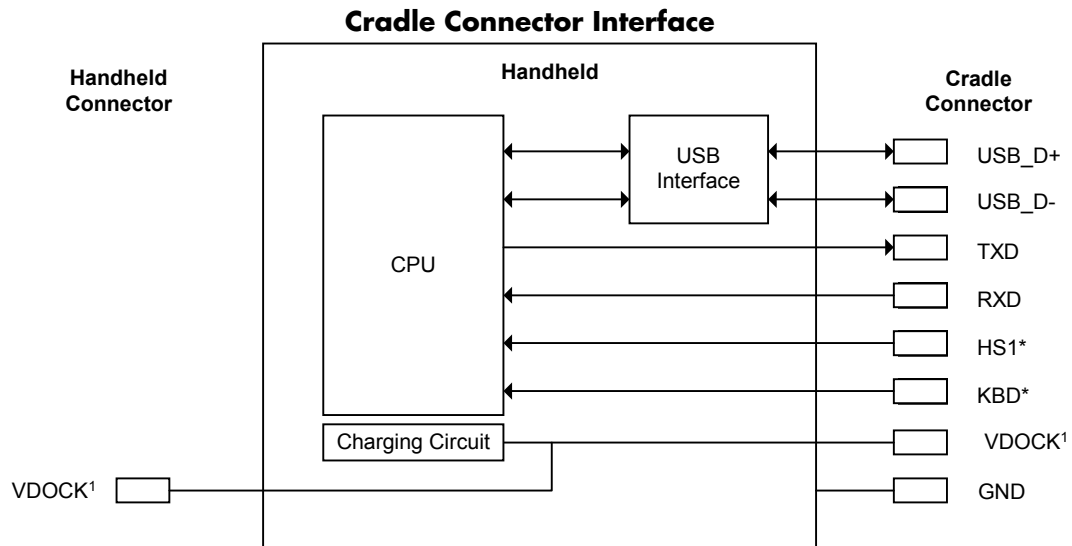
<http://www.markspace.com/datacord.html>

<http://www.atlconnect.com/handspringsource.html>

## 8.4 Cradle Interface Pinout

### 8.4.1 Visor, Visor Deluxe, Visor Platinum, Visor Neo, Visor Pro and Visor Prism

This section defines the functions of the signals on the eight-pin Cradle Connector. The signals are described in alphabetical order following the table below. Active-low signals have a "\*" at the end of their names.



1. Only rechargeable products (in this section, Visor Prism and Visor Pro) include a cradle that supplies power to this pin.

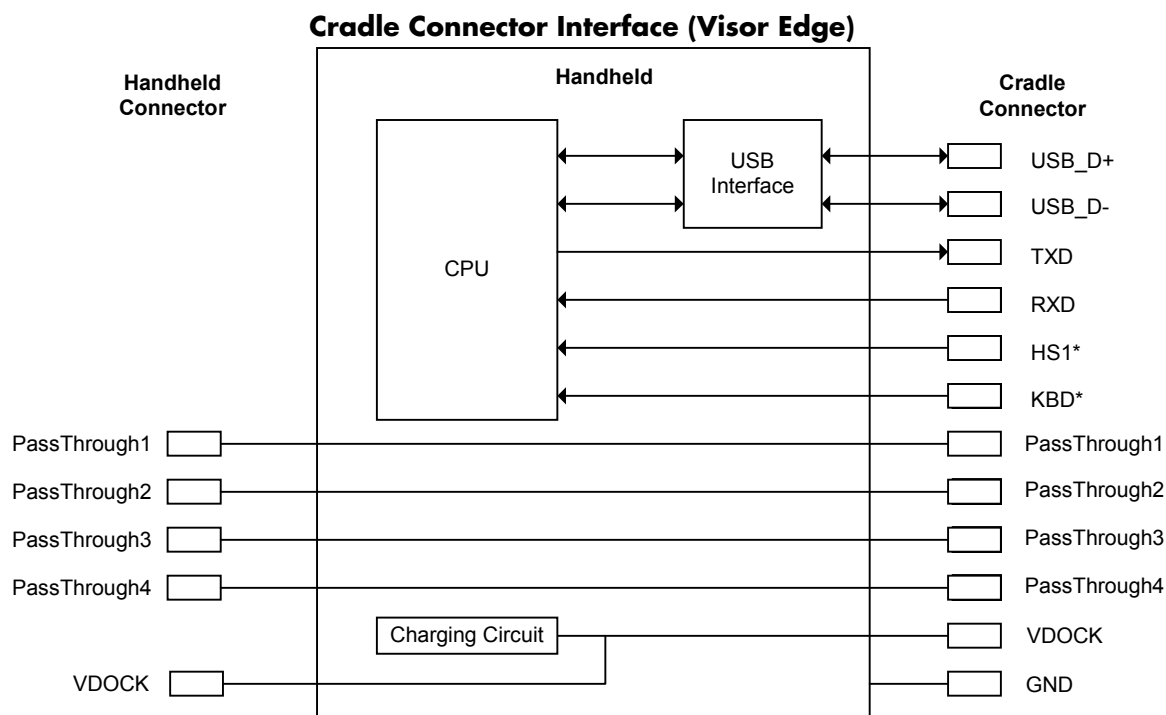
**Cradle Connector Pin Summary**

Pin	Name	I/O/P <sup>1</sup>	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, and PU = pulled up. For example, the HS1\* signal is a direct input signal to the Visor CPU.

## 8.4.2 Visor Edge

This section defines the functions of the signals on the twelve-pin Visor Edge Cradle Connector. The signals are described in alphabetical order following the table below. Active-low signals have a “\*” at the end of their names.



**Cradle Connector Pin Summary (Visor Edge)**

Pin	Name	I/O/P <sup>1</sup>	Function
1	RXD	I	Receive Data
2	TXD	O/P	Transmit Data/Power
3	KBD*	I	Keyboard Detect
4	HS1*	I/PU	HotSync Interrupt
5	USB_D-	I/O	USB Data
6	USB_D+	I/O	USB Data
7	VDOCK	P	Cradle Power
8	GND	P	Ground
9	PASSTHROUGH1	I/O	Pass Through
10	PASSTHROUGH2	I/O	Pass Through
11	PASSTHROUGH3	I/O	Pass Through
12	PASSTHROUGH4	I/O	Pass Through

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

### 8.4.3 Cradle Interface Signal Descriptions

Active-low signals are denoted with an asterisk (“\*”).

#### **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** **HS1\***

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 through 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 on the RXD pin. Refer to the [Remote UI](#) section for more details.

In Visor, when this signal is asserted, 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.

#### **Pass Through** **PASSTHROUGH[1:4]**

These pins are passed through the Cradle Interface to the Edge Connector. They are intended for future peripheral use and are presently *reserved*. These signals are not brought out in the Detachable Springboard Slot. These signals are only present on the Visor Edge native Edge Connector.

#### **Receive Data** **RXD**

RXD connects to the Visor Edge 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 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 2.7V when KBD\* is asserted for low-power peripherals, such as keyboards. The short circuit current is 6mA minimum

#### **USB Data** **USB\_D+, USB\_D-**

USB\_D+ is the positive signal in the USB differential pair. USB\_D- is the negative signal. This signal and USB\_D- implement the USB signaling protocol for communicating with a USB master, 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 into a special charging dock. The handheld passes this charging supply from a pin on its cradle connector through two pins (VDOCK) on the Springboard expansion module connector. VDOCK provides 4.75 – 6.2V on two pins with a maximum current of 500mA. Note that VDOCK is not current limited. Developers should design modules such that the VDOCK specification is met. Details on the Visor Edge AC Adapter are provided in the next section.

## 8.4.4 Charging Springboard Modules

There are a few considerations to be aware of when Visor Edge and a Springboard module are charging simultaneously.

### 8.4.4.1 Battery Charging Overview

The Cradle Power pin (VDock) is used to charge the handheld (where applicable), as well as provide charging power to Springboard modules with a built-in battery. The Springboard specification for VDock at the Springboard connector is 4.75 to 6.2V, 500mA max. While measured voltages may result in a smaller range, developers should design for the full 4.75 - 6.2V range due to the possibility of various 3<sup>rd</sup> party peripherals providing power to the handheld at the limits of this specification.

The AC Adapter used to charge Visor Prism, Visor Edge and Visor Pro is rated at 5.9VDC (+/- 5%), 2A. The VDock specification accommodates voltage drops across the cradle and cables. Developers should use the AC Adapter that comes with the device if at all possible. Developers providing their own power adapter will need to ensure that the VDock specification is maintained across optional peripheral equipment when the handheld device is charging, as well as ensure that sufficient current is available to charge the handheld and module simultaneously.

### 8.4.4.2 Stay On In Cradle

This feature is only applicable to rechargeable Handspring handheld devices.

Some developers may want the unit to stay on indefinitely while in the cradle. To accomplish this within software, developers should design the application to periodically call the `EvtResetAutoOffTimer()` function rather than programmatically adjust the “StayOnInCradle” preference. The operating system does not support this preference setting.

### 8.4.4.3 Remote UI

The Visor Edge handheld supports character input from an external keyboard or pen-based device through the Cradle Connector located at the bottom of the handheld device. Grounding the KBD\* pin on the Cradle Connector indicates to the handheld processor that:

- A keyboard or other remote UI device is present on the Cradle Connector, and,
- Incoming serial data packets on the RXD pin 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.

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/>.

The packet format for the Remote UI protocol is described in an application note titled “AN-16: Remote UI” on the Handspring website at:

[http://www.handspring.com/developers/tech\\_notes.jhtml](http://www.handspring.com/developers/tech_notes.jhtml)

## 9. Mechanical Information

---

Complete mechanical information for Handspring handheld devices and associated peripherals is available on the Handspring website. The formats used include DWG, DXF, PRO-E, IGES and STEP. Not all formats are available in each documentation package. For the latest files go to:

[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)

Mechanical files on the website will cover three general topics:

- Handspring handhelds
- Handheld cradle
- Springboard modules
- Detachable Springboard Slot

All dimensions in this section are in millimeters and the accuracy has been intentionally reduced to 0.1mm. These drawings are intended to provide a general overview of the type of information available. The files described above provide measurements in more detail to a greater precision.

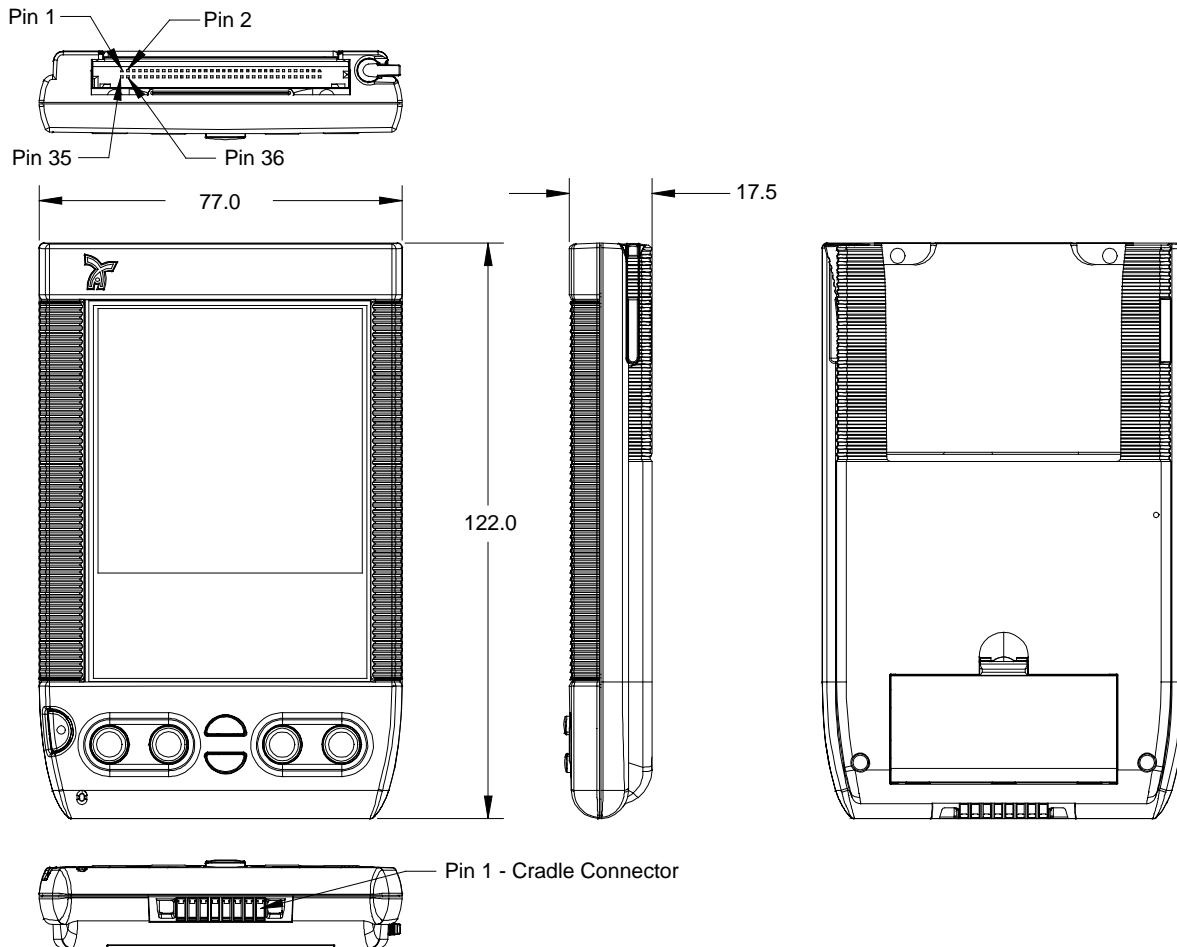


## 9.1 Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro

Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro use the same basic mechanical design. Note that Visor Pro does not have the external battery door (shown below) because it is a rechargeable product.

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

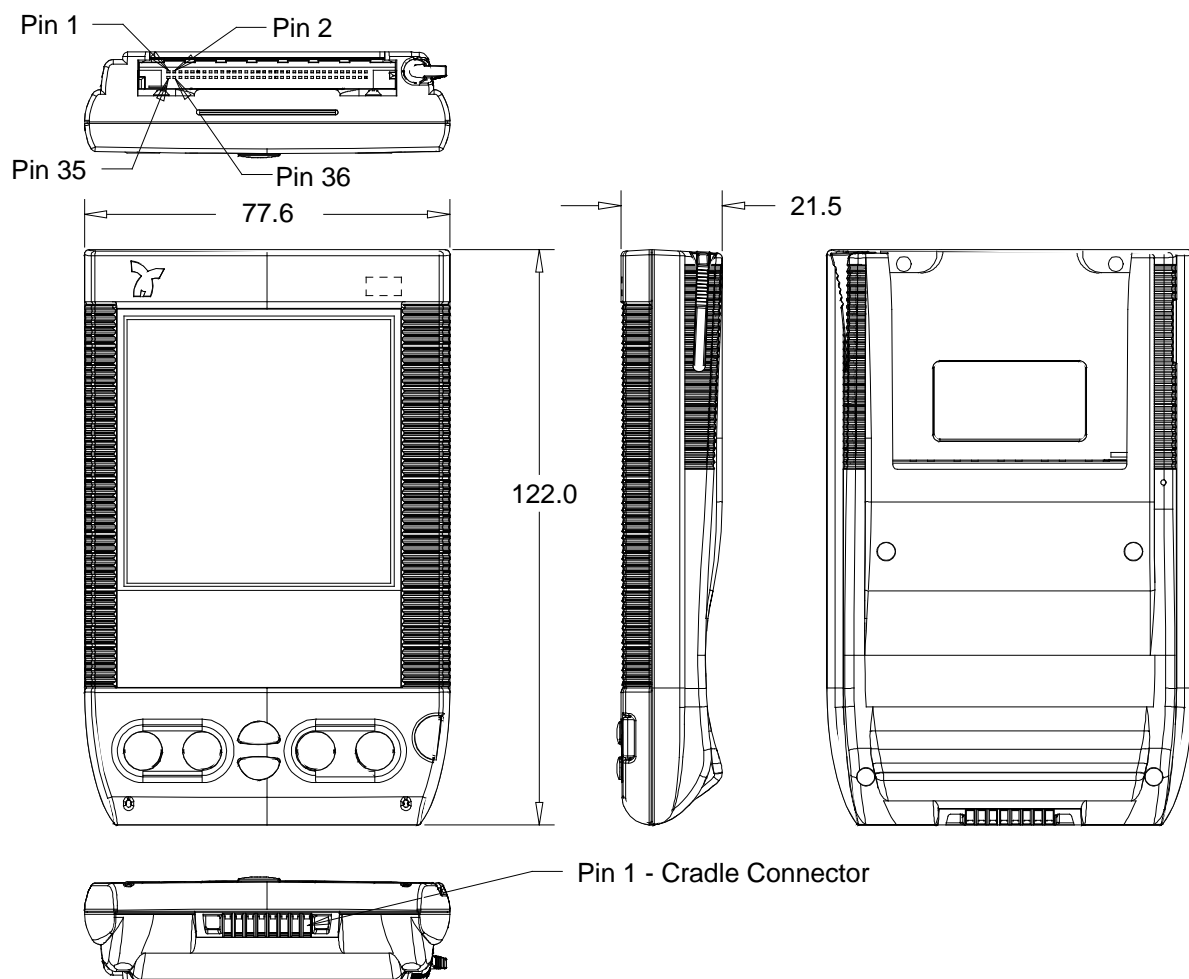
[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)



## 9.2 Visor Prism Mechanicals

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

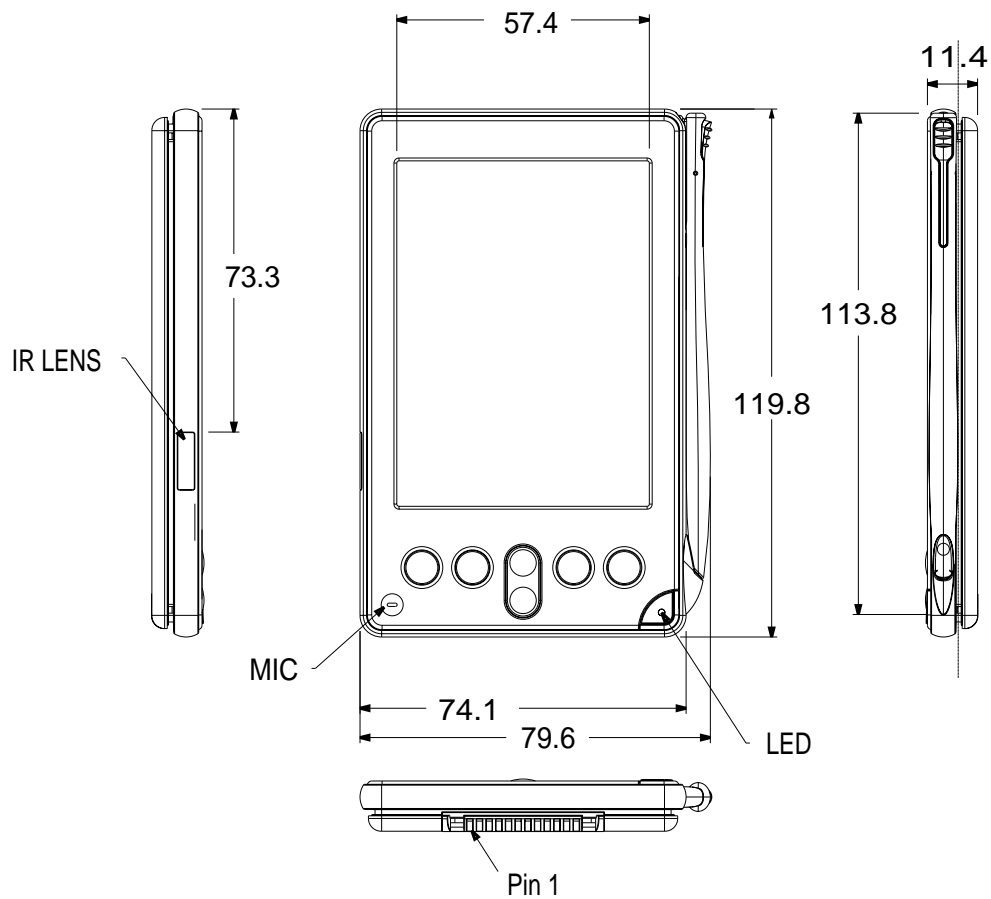
[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)



## 9.3 Visor Edge Mechanicals

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

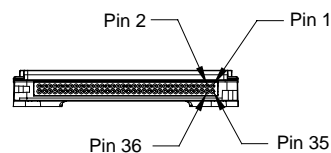
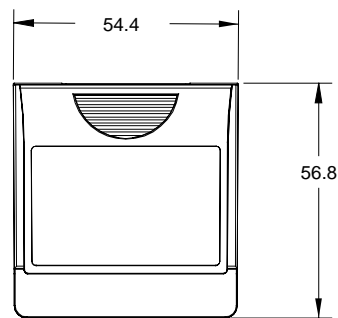
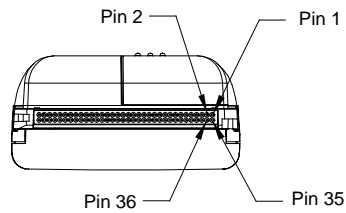
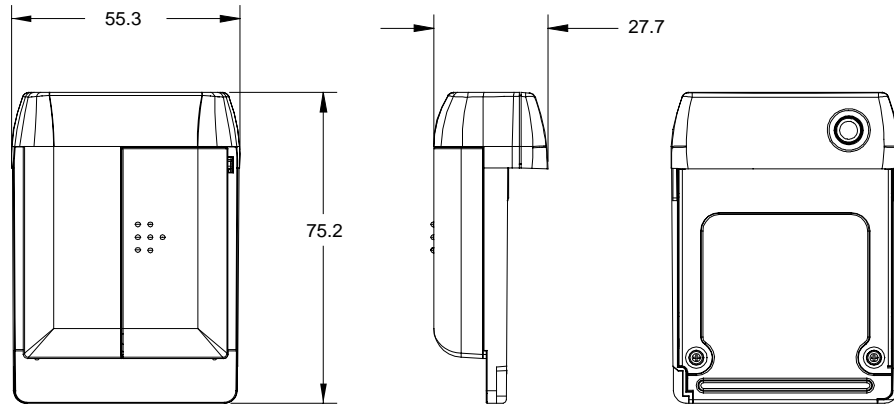
[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)



## 9.4 Springboard Modules

Springboard modules are mechanically compatible with the Detachable Springboard Slot.

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 at: [http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)



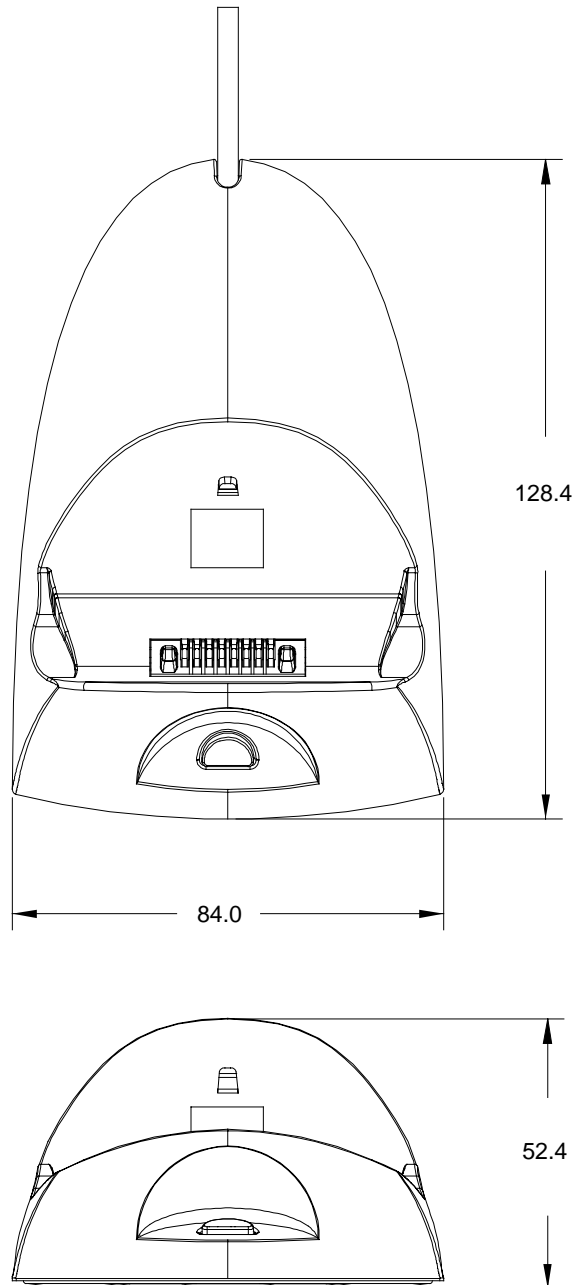
## 9.5 Cradle Base

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)

### 9.5.1 Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro Cradle

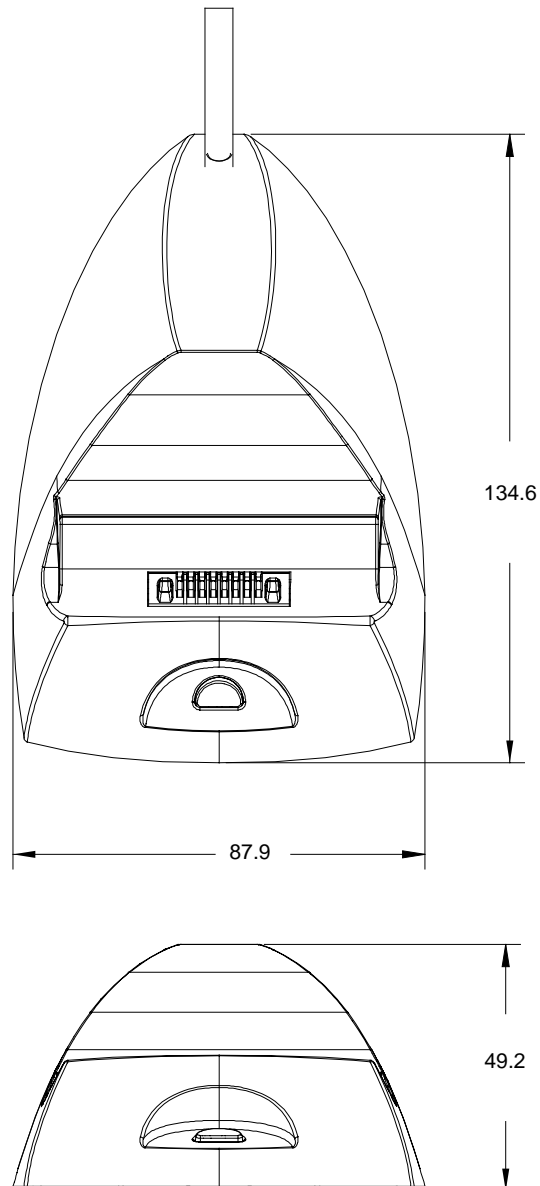
Visor, Visor Deluxe, Visor Platinum, Visor Neo and Visor Pro use the same basic cradle mechanical design. Note that the Visor Pro cradle is a charging cradle.



### 9.5.2 Visor Prism Cradle

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

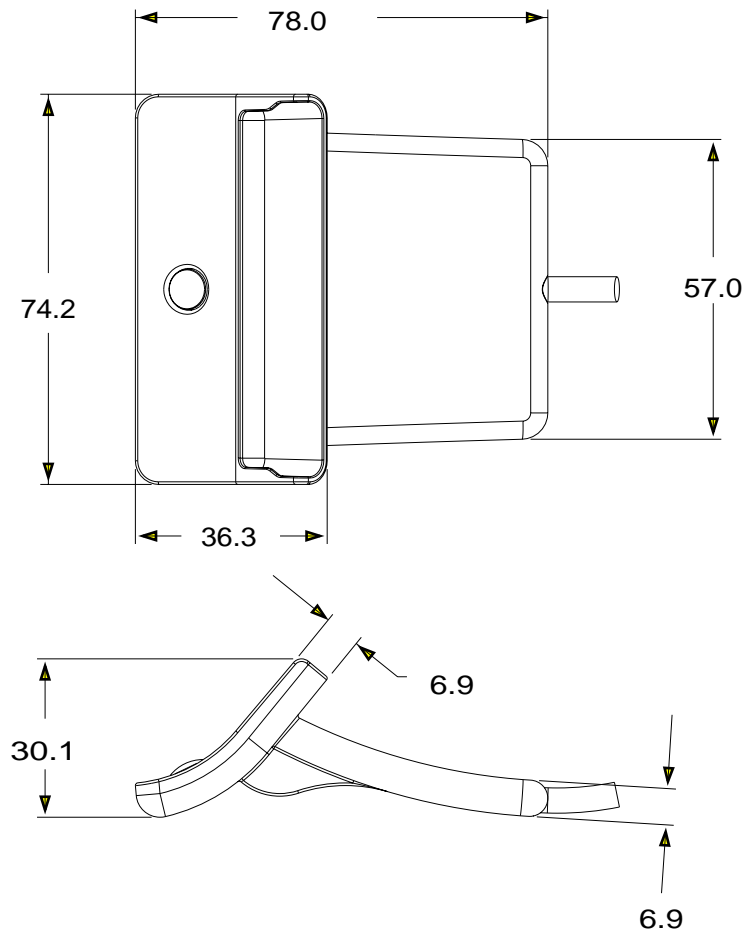
[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)



### 9.5.3 Visor Edge Cradle

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)

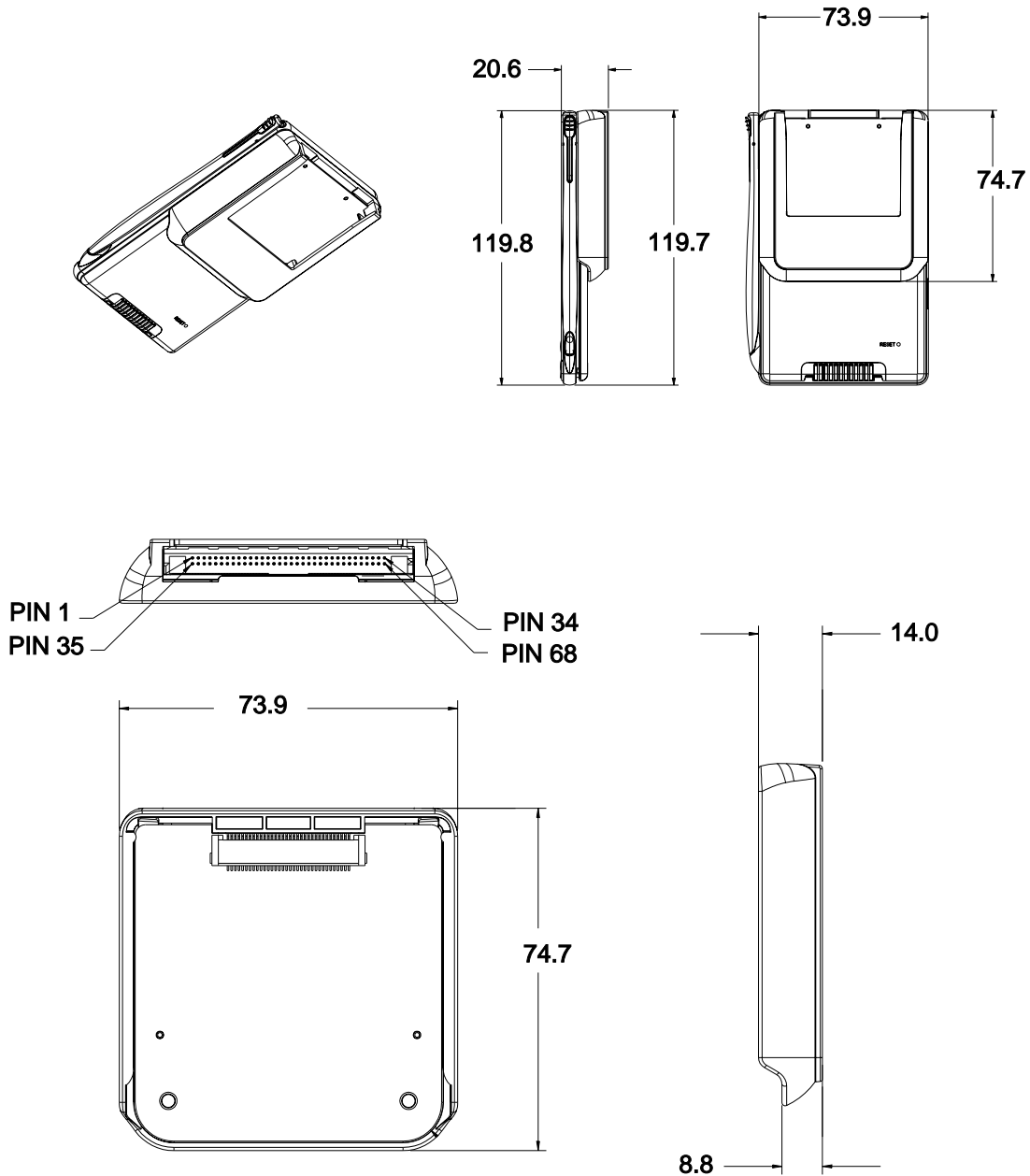


## 9.6 Detachable Springboard Slot (Visor Edge Only)

The Detachable Springboard Slot allows the use of Springboard modules with the Visor Edge handheld.

For complete mechanical files, please visit the Mechanical section of the Handspring WWW site at:

[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)





## 9.7 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; however, 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* and the Mechanical section of the Handspring WWW site:

[http://www.handspring.com/developers/dev\\_mechanical.jhtml](http://www.handspring.com/developers/dev_mechanical.jhtml)

## 10. Environmental Test Specifications

The following specifications characterize generic “normal” usage guidelines and do not necessarily reflect more stringent acceptance tests for a particular handheld. Developers may want to consider testing to a more rigorous range to ensure sufficient margin and allow for deviations in manufacturing. Developers should note that an inserted module may significantly alter the results of a specific test (e.g. may change overall mechanical characteristics).

Finally, tests for reliability of a specific mechanical feature (e.g. hinge) of a peripheral is based on anticipated “worst-case” usage that varies by product. Our guidelines in these cases is to test a “larger” (relatively) sample to “worst-case”+20%, and a smaller sample tested to failure or to such a level that far exceeds worst-case.

<b>Altitude</b>	
Operating	15,000 ft @ 25° C
Non-Operating	30,000 ft @ 25° C
<b>Ambient Temperature</b>	
Operating Range	0°C to 40°C
Power-off Storage	-20°C to 60°C
Temperature Gradient	20°C/hour maximum
<b>Ambient Humidity</b>	
Operating	5% to 90% RH
Power-off Storage	5% to 90% RH
<b>ESD</b>	
Non-metallic areas	Performed to IEC 1000-4-2 +/- 8kV air discharge
Metallic areas	Performance to IEC 1000-4-2 +/- 4kV touch discharge
<b>Durability</b>	
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)
Random Vibration – Operational	6.0g RMS Input, 10 Hz – 1 KHz, 1 hour dwell in each axis
Springboard Insertion Test	4000 insertions

<b>Agency Approvals</b>	
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)

## 11. Handspring Developer Agreement

---

Handspring Licensing

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(TM) Enabled Products" are Handspring handheld computers that contain an external "slot" (the "Springboard(TM) 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(TM) slot and meet Handspring's Springboard(TM) compatibility requirements ("Licensed Plug-Ins") or ii) accessory products (such as keyboards or reference manuals) for use with Springboard(TM) 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(TM) 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(TM) 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(TM) 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(TM) 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(TM) 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(TM) 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(TM) 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.