

ScanConfig

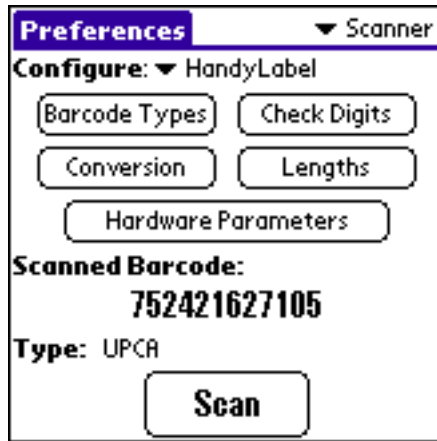
Configuration Software for Symbol PalmOS Devices including SPT1500, SPT17xx, SPT18xx, and CSM150

Users Guide

Version 1.2
October 2003

- [Introduction](#)
- [Installation](#)
- [Running the Software](#)
 - From the Main (Home) Screen
 - From Another Application
- [Configuring the Scanner](#)
 - [Barcode Types](#)
 - [Check Digits](#)
 - [Conversion](#)
 - [Lengths](#)
 - [Hardware Parameters](#)
- [Testing the Settings](#)
- [Creating Multiple Configurations](#)
- [Miscellaneous Menu Options](#)
 - [Copying Barcode](#)
 - [AutoConfiguration](#)
 - [Restoring Factory Defaults](#)
 - [Password Protection](#)
 - [Version Information](#)
- [Configuring Multiple Handhelds](#)
- [Technical Support](#)
- [Programming Notes](#)
 - [Bad Scan Beep](#)
 - [Interactive External Configuration](#)
 - [Automatic Configuration](#)

Introduction



ScanConfig is a new Palm "Preference" which lets you configure all aspects of the operation of PalmOS barcode-scanning handheld devices from Symbol Technologies, including the SPT1500, SPT17xx, and the plug-in CSM150 for the Handspring Visor. All of these aspects are typically under programmatic control, so they have been accessible to programmers writing custom software running on those units, but without **ScanConfig** they were not accessible to the end user.

ScanConfig not only lets you configure all parameters which control the operation of the scanner, it even lets you set up multiple sets of configurations for different applications, and even lets programmers use **ScanConfig** to automatically preconfigure the scanner to the parameters you have chosen for that application.

Installation

Windows:

Double-click on the file **ScanConfig.prc**. A window labelled **Install Tool** should appear, with a UserName selection box with the name of one or more Palm handheld units. Select the one you wish to install the software in, and click on **OK**. A second **Install Tool** window will appear, showing **ScanConfig** in a list of programs to be installed. Click on **Done**, and another window will appear informing you that **ScanConfig** will be installed in your Palm the next time you do a HotSync. Perform a HotSync and the software will be installed.

Macintosh:

Select **Install Handheld Files** from the **HotSync** menu. Make sure the User is set to the handheld unit in which you want to install the software, and drag the **ScanConfig.prc** file into the large box in the window (or use the **Add To List** button to select the file). Close the window, perform a HotSync, and the software will be installed.

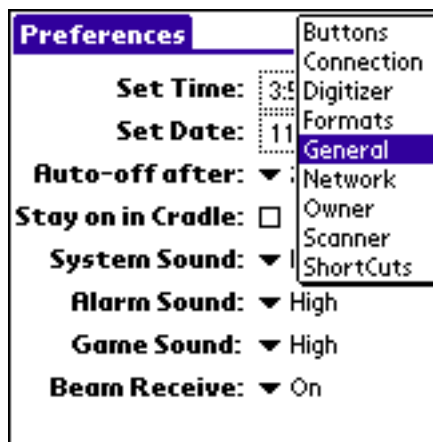
Running the Software

From the Main (Home) Screen

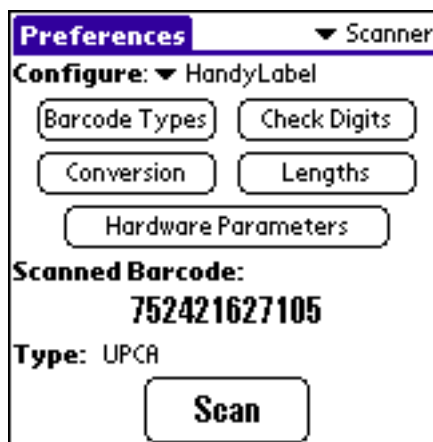
On the main Palm ("Home") screen you'll see a series of icons representing different programs. ScanConfig does NOT appear on this screen directly. Instead, it is one of the Palm "Preferences" which are accessed using the "Prefs" icon:



When you select the **Prefs** application, the most recently used preference screen will appear. It might, for example, be the "General" preferences which looks like this:



When you see this screen, in the upper right you'll see a list of all the available preferences. Select **Scanner** from this list, and the **ScanConfig** preference application will start.

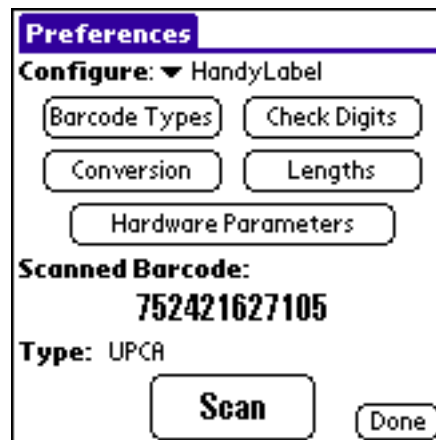


You can return to any of the other preferences using the same pop-up menu in the upper right corner of the screen, or tap the Home button to return to the main screen.

From Another Application

Some other applications, such as **On Hand** from **Stevens Creek Software**, are

"ScanConfig-aware", and have a menu or other means to transfer directly to **ScanConfig** to configure the scanner. In that case, the **ScanConfig** screen will look slightly different:



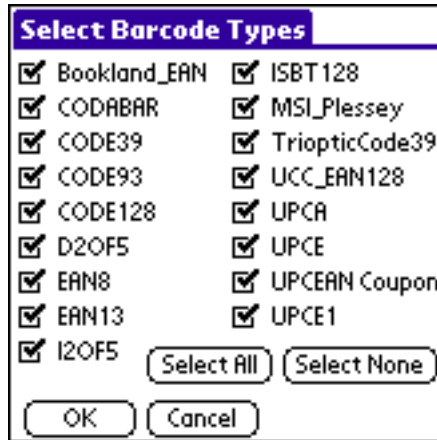
Note that the pop-up menu of different preferences in the upper-right hand corner of the screen is absent, and instead, a **Done** button appears in the lower right of the screen. After you are finished using **ScanConfig**, tap on **Done** and you will be returned to the application which you were using before you launched **ScanConfig**.

Configuring the Scanner

There are five configuration buttons on the main screen, leading to five separation configuration windows:

Barcode Types

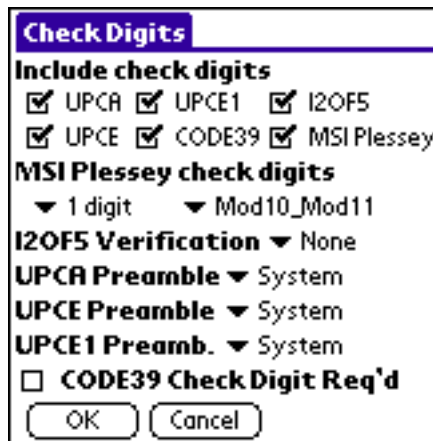
There are a series of barcode types which can be scanned or not, depending on which boxes are checked:



It might seem that one should always check all types so that all barcodes will be scanned. After all, if you never encounter an MSI_Plessey barcode, whether that box is checked or not is irrelevant. However, there are some very specific cases where the boxes checked will actually change the barcode which you scan. For example, if you scan a food coupon in your Sunday paper and "UPCEAN Coupon" is checked, you will typically see a 22-digit "Coupon" code, consisting of a 12-digit UPC code followed by a 10-digit coupon code. If you uncheck the "UPCEAN Coupon" box, and scan again, you'll scan just a 12-digit UPC code. Even "stranger" to those unfamiliar with barcodes are the barcodes which appear on books, which are actually dual barcodes (leaving aside the supplemental barcode for the price, discussed below). If you scan a book barcode with all codes enabled, you'll read a 10-digit "Bookland EAN" barcode, which corresponds to the ISBN number of the book. But if you disable Bookland_EAN by unchecking that box, now if you scan the same barcode you'll see a 13-digit EAN barcode. If you are trying to match the scanned barcode to information in a database, clearly you need to select the correct barcode types so that the scanned barcode will match as you expect.

Check Digits

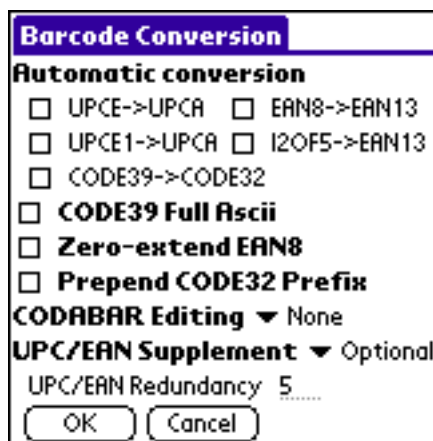
The "Check Digit" screen lets you select whether or not the scanned barcode should include a check digit for several barcodes for which that option is available; again, whether or not you want this to happen depends on whether or not the barcodes in your existing databases have or don't have that information. In addition to the check digits, there are a number of related items on this screen:



- **MSI Plessey check digits**, unlike other check digits, actually have several options. They can either have **1** or **2** check digits, and, if they have two, they can use two different check digit schemes, known as **Mod10_Mod10** and **Mod10_Mod11**.
- **I2OF5 Verification** can be three different choices - **None** (no verification of the check digit), **USS** (one check digit verification scheme), and **OPCC** (another scheme).
- **UPCA, UPCE, and UPCE1 Preambles** (the initial digit of a barcode) have three choices: **None** strips off the preamble digit, **System** returns the "normal" starting digit of the barcode, and **System/Country** adds an additional starting digit at the front which corresponds to the country.
- **CODE39 Check Digit Required**, if checked, will only return CODE39 barcodes which include a check digit (not all do).

Conversion

Sometimes, the barcode that is scanned is not exactly what you want; various "conversions" can be automatically performed by the scanner:



- **Automatic conversion** converts one barcode type to another. One of the most commonly needed is UPCE->UPCA, which automatically converts "compressed" 8-digit UPC codes (typically found on smaller items) into "normal" 12-digit UPC codes.
- **CODE39 Full ASCII** specifies whether various special characters are allowed in CODE39 barcodes.
- **Zero-extend EAN8** does what it says.
- **Prepend CODE32 Prefix** prepends a prefix on CODE32 barcodes.
- **CODABAR Editing** presents three options for conversion of CODABAR barcodes: **None** scans the barcode as is, **Notis** editing strips the leading and trailing characters from

the barcode, and **Clsi** editing strips the leading and trailing characters, and inserts spaces after the first, fifth, and tenth characters.

- **UPC/EAN Supplement** has three options for scanning barcodes which may or may not have "supplements." Typical examples of these are books, which have a 5-digit "supplemental" barcode representing the price, and magazines, which have a 2-digit supplemental barcode representing the month. The three options for these barcodes are **Required**, which will ONLY scan barcodes which include such supplements and will not scan "plain" UPC/EAN barcodes, **Optional**, which will include such supplements in the scanned barcode if they exist, and not if they don't, and **Ignore**, which will strip out the supplementals if they do exist, and will produce only the "plain" barcode.
- **UPC/EAN Redundancy** is a number which specifies how many times UPC/EAN barcodes are scanned in cases in which the supplement is **Optional**, to be absolutely sure that they are or are not present.

Lengths

A number of barcodes can have specific length restrictions placed on them.

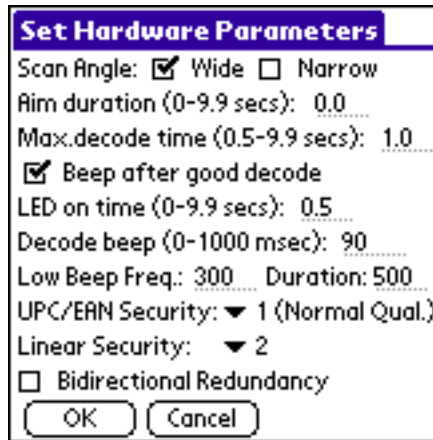
Barcode Lengths			
		Length 1	2
CODABAR	▼ Range	7	14
CODE39	▼ Any	0	0
CODE93	▼ Any	0	0
D2OF5	▼ Any	0	0
I2OF5	▼ Any	0	0
MSI Plessey	▼ Any	0	0

OK Cancel

For each of these barcodes, you can select four choices for length: **Any**, **1 Fixed**, **2 Fixed**, and **Range**. **Any** allows any length. **1 Fixed** allows only barcodes of the length specified in column 1. **2 Fixed** allows two different specific lengths, specified in columns 1 and 2. And **Range** allows any barcode whose length is greater than or equal to the number in column 1, and less than or equal to the number in column 2. The subtle issue here is that these restrictions apply BEFORE other modifications. Thus if a CODABAR barcode is being subject to Clsi or Notis editing (see previous section), the lengths here apply before that editing.

Hardware Parameters

A number of hardware parameters control the actual operation of the scanner:

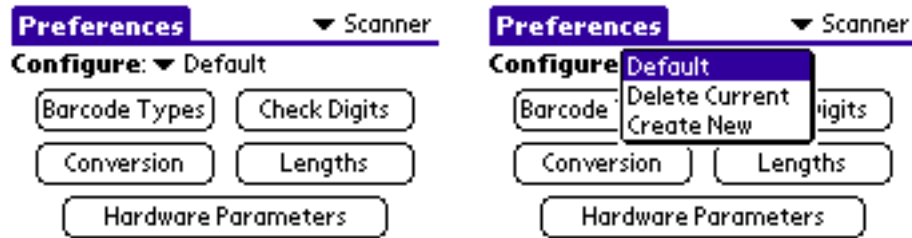


- **Scan Angle** can be wide or narrow.
- **Aim duration** can be 0.0 to 9.9 seconds. If the duration is non-zero, than for the specified amount of time, before the actual barcode scan occurs, the units operates in "laser pointer" mode, allowing the user to aim the beam directly at the barcode which is then scanned.
- **Maximum decode time** can be 0.5 to 9.9 seconds. After this amount of time, the scanner will "give up" and assume that whatever you are scanning can't be scanned.
- **Beep after good decode** does what it says.
- **LED on time** can be 0.0 to 9.9 seconds, and turns on the green LED on the unit for the specified time after a good scan occurs.
- **Decode beep** is specified in milliseconds from 0 to 1000 (1 second), and controls how long the scanner beeps after a good scan (if the "Beep after good decode" box is checked).
- **Low Beep Freq** (in Hz) and **Duration** (in msec) control the "low beep" of the unit. Unlike the "Beep after good decode" discussed above, the Symbol unit does not by itself beep after a "bad decode." However, if a programmer specifies in their software that the unit should issue a "low beep" after a bad scan, then the frequency and duration specified here will control the sound of that beep.
- **UPC/EAN Security** and **Linear Security** can range from 1 to 4. The higher the number, the more times a barcode must be scanned (and the same result achieved) before the scan is reported. Only use high numbers if the quality of the barcodes is poor (i.e., faded or otherwise damaged). **Linear Security** applies to "linear" barcodes including CODE39 and I2OF5.
- **Bidirectional Redundancy** requires that barcodes be scanned successfully in both directions before decoding.

Testing the Settings

Whenever you change settings on any of the five screens described above, the scanner is immediately reconfigured to those settings (you'll hear a multiple beep sound which confirms that the scanner has been given the new settings). Now you can press the on-screen **Scan** button to scan a barcode, or, if you have a Symbol SPT1500 or SPT17xx, the physical scan buttons, and confirm that the settings you have established produced the results you want.

Creating Multiple Configurations

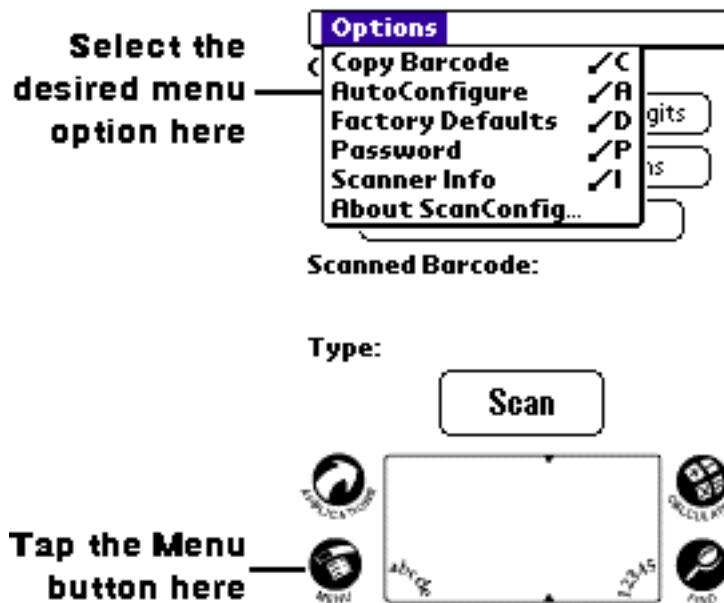


Near the top of the **ScanConfig** screen is a pop-up menu labelled **Configure**, as shown above. Tapping on the arrow will show a list of three or more items. The first time you use the program, the list will appear exactly as shown here. **Default** represents the current or default configuration of the scanner. **ScanConfig** lets you create up to 14 different configurations of the scanner, to match your needs for various applications which might require different configurations. To create a new configuration, select **Create New** from the menu. You'll be asked to name the new configuration, and then it will appear on the list (in this example, it will appear immediately below "Default" in the updated list). **Delete Current** deletes the currently selected configuration (the one which appears when the pop-up menu is now showing, as on the left above), with the proviso that you cannot delete the last remaining configuration (that is, you must always have one configuration).

One special feature won't be obvious from **ScanConfig** itself, but will become apparent if you have another program installed on the unit which uses **ScanConfig** to configure the scanner, such as **On Hand** (version 3.3 or higher), **CatScan** (version 1.3 or higher), or **Take An Order!** (version 2.3 or higher) from **Stevens Creek Software**. If you are running these or a similar program, and use a menu option in that program to access **ScanConfig** (rather than accessing **ScanConfig** via the **Prefs** application), then a configuration named after the program ("On Hand" for example) is automatically created.

Miscellaneous Menu Options

There are a number of special options which are available using the **Options** menu. To select one of these options, first tap the **Menu** button on the lower left-hand corner of the screen to cause the menu to appear; then tap the desired menu option.



Copying Barcode

Under rare circumstances, you may need to scan a barcode to use in another program which is not barcode-enabled. To do this, after scanning a barcode using **ScanConfig**, select **Copy** barcode from the **Options** menu, then transfer to the other application and use its **Paste** command.

AutoConfigure on Reset

When the Symbol unit is reset using a "soft" or "hard" reset, the configuration of the unit is reset to its "factory default" settings. You may prefer that it be set instead to the current settings you have established using **ScanConfig**. To make sure this happens, select **AutoConfigure** from the **Options** menu, and check the box marked **AutoConfigure on Reset**.



With this box checked, whenever the unit is reset, **ScanConfig** is automatically run once the unit "boots up," and the default configuration (the last configuration chosen the last time **ScanConfig** was run directly by the user) is established.

Restoring Factory Defaults

Symbol establishes certain factory defaults for the settings of the scanner. Although there is no guarantee that these settings are particularly appropriate to your needs, you can if you choose to initialize all settings to those defaults. To do this, select **Factory Defaults** from the **Options**

menu. The currently selected configuration ("Default" in the example in the previous section) will be the one which is modified.

Password Protection

If some cases, you may want a supervisor or IT person to establish the configuration of the scanner, but prevent the end-user from making any changes to that configuration. **ScanConfig** lets you do this by establishing password protection. To do this, select **Password** from the **Options** menu and you'll see this screen:



A dialog box titled "ScanConfig Password Entry" with a purple header. It contains a checkbox labeled "Password required for changes". Below the checkbox is a text field labeled "Password:" followed by a series of dots. At the bottom are two buttons: "OK" and "Cancel".

In order to restrict access to the software, check the box marked "Password required for changes" and enter a password (15 characters or fewer). Once you do this, future attempts to use the software will be met by this screen:



A dialog box titled "ScanConfig Password Entry" with a purple header. The text inside reads: "This unit is password protected. In order to change the configuration, you need to enter the password:". Below the text is a series of dots representing a password input field. At the bottom is a single button labeled "OK".

Only after the password is correctly entered is the user able to modify configurations, change configurations, or select different configurations, in other words, to do anything which will change the operation of the scanner.

BE SURE TO WRITE DOWN THE PASSWORD SOMEWHERE if you use this feature. If you password protect the software, and forget the password, your only recourse is to delete the software from your Palm (using the **Delete** menu from the main Palm "Home" screen) and then reinstall it.

Version Information

When obtaining technical support from Symbol or Stevens Creek Software, or when reporting a problem, you may need to have access to the software revision levels on your system, for both the scanner software itself as well as for **ScanConfig**.. To access this information, select **Scanner Info** from the **Options** menu and you'll see this screen:



A dialog box titled "Version Information" with a purple header. It displays the following information: "Scan Manager: SML125", "Decoder: SPT1D101N", "Port Driver: SPD125", and "ScanConfig: 1.0". At the bottom is a single button labeled "OK".

Configuring Multiple Handhelds

If you are using **ScanConfig** on multiple handheld units, you might want to transfer the settings from one unit to another one. Because these settings are maintained in a standard Palm "database", this is a simple matter. After creating the configuration(s) on one handheld, make sure that the HotSync Manager has its "System" conduit (called "Backup" if you are using a Macintosh) set to "Backup" (this is the default setting for this conduit, so most likely you'll find it already set to this value). With this setting, all Palm databases which do not have their own "conduit" are backed up automatically to a special folder in your user folder. Using Windows, and assuming your handheld unit is named "Sales1" and that you have a "typical" setup, the folder in question will be

C:\Palm\Sales1\Backup (on Macintosh, it will be named

MacintoshHD:Palm:Users:Sales1:Backups). In both cases, the file itself containing the database of scanner configurations is called `ScanCfgDB.pdb` (so if you have trouble locating the right folder, just use your computer's "Find" feature to search for that file). `ScanCfgDB.pdb` is a file which can be installed, just like a Palm program, into any PalmOS handheld unit, using the standard Palm install program. So all you need to do to configure multiple units identically is to install `ScanConfig.prg` in one handheld unit, set up the configuration or configurations that you want, even enter a password if you wish, then perform a HotSync, and now install *both* `ScanConfig.prc` AND `ScanCfgDB.prc` into any other handheld you wish to configure in the same way.

Technical Support

If you need technical support for **ScanConfig**, you should first check our support web page, <http://www.stevenscreek.com/palm/support.html>, where we have tried to assemble answers to all the most commonly encountered problems with downloading, installing, and using our software. If that doesn't solve your problem, we encourage you to do so by email at support@stevenscreek.com. We provide phone support for our "regular" software (like **On Hand**, **CatScan**, and **Take An Order!**), but we provide email support ONLY for **ScanConfig**. If you have programming questions, please email them to development@stevenscreek.com.

Programming Notes

This section is designed to be read by programmers, who are developing programs which will interact with [ScanConfig](#). If you are an "end-user" of [ScanConfig](#), and not a programmer, you should skip this section.

[ScanConfig](#) is designed to be used by other programs in a number of ways:

Bad Scan Beep

Symbol units can be configured by the user (see above) to beep after a good scan, but strangely enough they can NOT be routinely configured to make a different sound after a bad scan; that is something that must be done by the programmer. [ScanConfig](#) does allow the user to configure the frequency and duration of the "low beep" which can be accessed by the programmer. To make use of this feature, we recommend writing your software in the following way:

```
else if (event->eType == scanDecodeEvent) {
    scanStatus = G_ScanGetDecodedData(&decodeDataMsg);
    if (scanStatus == STATUS_OK) {
        if (decodeDataMsg.type!=BCTYPE_NOT_APPLICABLE) {
            // Process the scan here
        }
        else G_ScanCmdBeep(One_Long_Low);
    }
    handled=true;
}
```

The key line here is the `G_ScanCmdBeep(One_Long_Low);` command, which uses the frequency and duration established by the user on the Hardware Parameters screen of [ScanConfig](#) to sound a long, low-frequency beep informing the user that a bad scan has occurred. Of course you can also use this in other ways, for example, you might use it even if the scan itself was good, but it didn't match anything in a database in your application. Of course that kind of use is entirely up to you.

Interactive External Configuration

It isn't at all necessary, since the user can always access [ScanConfig](#) via the **Prefs** application, but you may wish to add a "Configure Scanner" menu to your application, and have that menu launch [ScanConfig](#) so that the user can configure the scanner. As described above, in this mode, [ScanConfig](#) will display a "Done" button which, when pressed, will return the user to your application. Note also that in this mode, the "normal" exits for [ScanConfig](#) (such as pressing the "Home" button to return to the main screen of the Palm) are disabled, so that the ONLY way out of [ScanConfig](#) is back through your application. Thus if your application wants to hold the user "captive" and not allow the user to run other applications residing on the Palm, this behavior will not be "betrayed" by [ScanConfig](#).

Programming access to [ScanConfig](#) for configuration MUST be done using the `SysUIAppSwitch` API command, and NOT the `SysAppLaunch` command; this occurs because, unfortunately, the Symbol scanner library uses global variables in ways that it probably shouldn't. As part of this, [ScanConfig](#) requires that you "tell it who you are", that is, you tell [ScanConfig](#) the "creator ID" of the application to which it should return when the **Done** button is pressed. Here is typical code that you can use to do this:

```

#define appFileCreator 'SCSi'
UInt32* creatorP;
LocalID ScanConfigID;
DmSearchStateType theSearch;
UInt cardNo;

else if (event->eType == menuEvent) {
    MenuEraseStatus(NULL);
    handled=true;
    switch (event->data.menu.itemID) {
        case scanSetupMenu:
            if (DmGetNextDatabaseByTypeCreator(true,
                &theSearch, 'panl', 'SCsc', false,
                &cardNo, &ScanConfigID) == errNone) {
                creatorP = MemPtrNew(4);
                MemPtrSetOwner(creatorP, 0);
                *creatorP = appFileCreator;
                SysUIAppSwitch(0, ScanConfigID,
                    sysAppLaunchCmdPanelCalledFromApp, creatorP);
            }
            else FrmCustomAlert(infoAlert, "ScanConfig application not found", "", "");
            break;
    }
}

```

Some of the key features of this code: First, note that **ScanConfig** is a 'panl' (not an 'appl') and its creator ID is 'SCsc'. Second, note that in order to pass information to **ScanConfig** (in this case, the creator ID of YOUR application), you need to create a new pointer and set that pointer to be owned by the system (the `MemPtrSetOwner` line). `appFileCreator` is, of course, the creator ID of your application, which is typically declared as a `#define` at the top of your code. And finally, note that we are using a "launch code" of `sysAppLaunchCmdPanelCalledFromApp`, which lets **ScanConfig** know it has been called by another application, and not via the Prefs application.

The other key feature of accessing **ScanConfig** in this way is the RETURN from **ScanConfig** to your application. This will occur NOT with a "normal launch" code but with a `sysAppLaunchCmdReturnFromPanel` launch code, which your application must be able to handle. Thus the `PilotMain` of your application might need to look like this:

```

if ((cmd == sysAppLaunchCmdNormalLaunch)
    || (cmd == sysAppLaunchCmdReturnFromPanel)) {
    StartApplication(cmd);
    FrmGotoForm(mainForm);
    EventLoop();
    StopApplication();
    return(0);
}
else return(0);

```

You could use the fact that different launch codes are used to access your application (one from the main Home screen, the other when returning from **ScanConfig**) to do different things. For example, you might want to display a "splash screen" on the initial entry to your application, but not when returning from configuring the scanner:

```

if ((cmd == sysAppLaunchCmdNormalLaunch)
    || (cmd == sysAppLaunchCmdReturnFromPanel)) {
    StartApplication(cmd);
    if (cmd == sysAppLaunchCmdNormalLaunch) FrmGotoForm(introForm);
}

```

```

    else FrmGotoForm(mainForm);
    EventLoop();
    StopApplication();
    return(0);
}
else return(0);

```

Automatic Configuration

If the user has established a configuration in **ScanConfig** named after your application (either automatically using the "Interactive External Configuration" method described in the previous section, or manually simply by entering **ScanConfig** and using the **Create New** option to create a suitably named configuration), then your application can use that configuration to automatically configure the scanner when the application starts. In other words, the user launches your application, your application immediately (before displaying any UI to the user) launches **ScanConfig** using a special launch code which tells **ScanConfig** to find the right configuration, configure the scanner, and exit. Your application is then launched a second time, this time with a different launch code, and this time it can "really" start up. All this is not only transparent to the user, but also occurs extremely quickly, so it is a perfectly viable way to write your application, and to be sure that when it starts, it will always start in a "known" state.

Typical PilotMain code which would accomplish this task might look like this:

```

#define sysAppLaunchConfigureScanner 32768

DWord PilotMain(Word cmd, Ptr cmdPBP, Word launchFlags)
{
    LocalID ScanConfigID;
    DmSearchStateType theSearch;
    UInt cardNo;
    UInt32* creatorP;
    Err err;

    if ((cmd == sysAppLaunchCmdNormalLaunch)
        || (cmd == sysAppLaunchCmdReturnFromPanel)) {
        if (cmd == sysAppLaunchCmdNormalLaunch) {
            if (DmGetNextDatabaseByTypeCreator(true, &theSearch, 'panl', 'SCsc',
                false, &cardNo, &ScanConfigID) == errNone) {
                creatorP = MemPtrNew(4);
                MemPtrSetOwner(creatorP, 0);
                *creatorP = appFileCreator;
                err = SysUIAppSwitch(0, ScanConfigID, sysAppLaunchConfigureScanner, creatorP);
            }
        }
        StartApplication(cmd);
        FrmGotoForm(mainForm);
        EventLoop();
        StopApplication();
        return(0);
    }
    else return(0);
}

```

As you can see, on the "NormalLaunch" (your application being started from the Home screen or "Launcher"), your application looks to see in **ScanConfig** is present; if it is, it is launched with a

special launch code `sysAppLaunchConfigureScanner`; if not, the application continues launching normally. If **ScanConfig** IS present, this application exits, **ScanConfig** runs, displays a message "Initializing Scanner..." on the screen (so the user knows what is happening and why your application isn't immediately appearing on screen), configures the scanner (non-interactively), removes the message from the screen, and then exits and re-launches your application, this time with the launch code `sysAppLaunchCmdReturnFromPanel`. With it receives this launch code, of course, your application must NOT launch **ScanConfig** on entry, lest an endless loop ensue, but instead just does its "normal" thing.

Note that **ScanConfig** only *configures* the scanner, it does not *activate* the scanner. Even when **ScanConfig** has been run automatically, your application still needs to activate the scanner using standard Symbol API commands:

```
ScanOpenDecoder();  
ScanCmdScanEnable();
```

and, when finished,

```
ScanCmdScanDisable();  
ScanCloseDecoder();
```

Copyright 2001-3 by [Stevens Creek Software](#)
All Rights Reserved