iPAD

Programmer's Reference Guide

Revision 1.0

Jan 15, 2003

P/N 90000607

FUJITSU

1 Introduction

The goal of this document is to provide the technical information necessary for a programmer proficient with Microsoft Visual C++ and Microsoft Windows CE.NET, to develop or port a C++ application to the iPAD. The following documents provide additional information on the use of the iPAD.

1. The iPAD System User's Guide (P/N 90000608).
2. The iPAD Operations Guide (P/N 90000609).
3. The iPAD Quick Reference Guide (P/N 90000610).

Microsoft's Compact Frame Work support is not included, therefore only C++ and browser based applications can be ported to the iPAD.

2 General Architecture

The iPAD has been designed primarily for use in 802.11b environments. The iPAD is delivered with:

- An Intel 802.11b compact RF card (optional).
- 32 Megs of flash ROM containing Windows CE.NET.
- 64 Megs of RAM.
- 32 Megs of Flash File System (FFS) to be used for non-volatile program storage.
- A Symbol 923 scanner.
- A Two Track Magnetic Stripe Reader (Tracks 1 and 2).
- Bluetooth

While Microsoft's Windows CE.NET development tools provide the interface to 802.11b and Bluetooth, the interface to the scanner and magnetic stripe reader are custom extensions to Windows CE.NET. Both wedge and API interfaces are provided for the magnetic stripe reader and scanner. The wedge interface places data received from the scanner or magnetic stripe reader into the Windows CE.NET keyboard buffer. The wedge program can be used with an application not specifically designed to work with these peripherals. An API interface is also provided which allows programmatic control over the arrival of data from the scanner and magnetic stripe reader..

While an application can be manually loaded into the iPAD via the network or ActiveSync, the recommendation is that an application bootstrap program be manually loaded into the iPAD via the iPAD software install process (see Section 3) and the application downloaded and updated automatically by the server. Each iPAD comes with a pre-paid license for Wavelink's Avalanche client that provides such a function.

While the iPAD contains both main and backup lithium ION batteries to guard against volatile memory loss, the application should be design to optimize the use of the 32 Meg flash disk folder. While the likelihood is greater that the content of RAM will be lost due to a cold boot to recover from a software lockup than due to low battery voltage, it is recommended that critical data be loaded or backed up in the flash file system (FFS) of the iPAD. The flash file system is 32 Megs in size and is accessed as the "\Flashdisk" folder.

3 Software Installation.

The iPAD contains a special software installation process that will automatically backup all the files being installed into the RAM Object Store (RAM disk) to the flash disk (permanent storage) folder. In the event of a cold boot, the files in the backup ("\flashdisk\system") folder will automatically be restored to the appropriate RAM Object Store folder. The contents of the files restored will revert to the contents at the time of the software installation. In addition to backing up the RAM Object Store files, all registry entries updated during the installation process are written to flash as well. Registry entries set by importing a REG file and/or as a result of running a control panel applet will be saved when completing the installation process.

While files can be placed in the RAM object store by a number of other methods, only files processed via this special install process will be automatically recovered.

Software installation is governed by a set of commands in the CONFIGHHT file. The installation process is started from the "Setup Menu" displayed when the unit first boots (after calibrating the touch screen and entering the date and time). Select entry #2, "Install the Application", from the "Setup Menu". Next select the install method. In a development environment, the preferred method will likely be via ActiveSync (USB). In a production environment, the method of choice will likely be via the RF LAN. **A storage card SHOULD NOT be used to load the iPAD if the iPAD already has an 802.11b card installed. The 802.11b card is difficult to remove and install**.

If the iPAD has been purchased with an 802.11b card installed, the software installation process will have already been performed to load the 802.11b driver. The "Setup Menu" will not be displayed when the iPAD is powered on. To conduct a new software installation, press the reset switch on the back of the iPAD with the Stylus and quickly hold the "SFT "+ "CAN" keys. After the "Setup Menu" is displayed first enter "3. iPAD configuration" and initialize the iPAD to remove the effects of the first software install prior to running the new install.

3.1 Software Installation Via ActiveSync

Once "Install the application" has been selected on the "Setup Menu", the first option is to install via ActiveSync (USB). Installation via ActiveSync is described in section 6.3.1.1. ActiveSync is used to transfer files from the PC to the iPAD and the commands in CONFIGHHT are used to deploy the files. The format of CONFIGHHT can be found in the System User's Guide.

3.2 Software Installation Via RF LAN

A second option for installing an application can be used in a Windows network environment. The following are the major differences between the ActiveSync and LAN install processes:

1. The LAN process does not use ActiveSync as the method of transferring files..
2. The LAN install processes the CONFIGHHT directly from the network. Files do not have to be copied to temporary storage in the iPAD. This is of particular importance if file space is an issue.

3. The LAN install can be conducted without interacting with a PC.
   This is often important in a production environment and usually less important in a development environment.

3.2.1 Establish the ESSID

When installing from LAN is selected, the next screen provides the opportunity to select the network adapter and establish the ESSID. The network adapter selection will default to "NETWLAN1/802.11b Wireless LAN". Do not change this value. If the SSID is not correct, set the SSID of the 802.11b access point. The input box prompting for ESSID should not be used to set the SSID for the RF card. Use the "Network Properties" button to start Microsoft's Zero Configuration utility. Select the "Wireless" folder to set the SSID. Select the "Restart before connection" check box if the SSID has been changed and then select "Next".

3.2.2 Identify the Host

After the SSID has been established, the iPAD will prompt for the server name. In the "Device Name" field enter a unique name for this iPAD. The device name must be changed from the default before the iPAD will be allowed on the network. In the "Server Name" field enter the server name. The name should not be preceded by back slashes or followed by back slashes. In the "Folder" field, enter the path to the Config.hht file on the server. The path should not be preceded by back slashes or followed by back slashes. When "Retry" is selected the iPAD will attempt to connect to the network. If the server is located, a User Name, Password, and Domain will be requested. Once this data is entered the CONFIGHHT file will be processed. When the iPAD returns to the "Setup Menu" select "End" to complete the install.

3.3 Recovery

In the event of a cold boot, the operator will be prompted to acknowledge the loss of RAM memory by responding to the message "The contents of memory has been lost. Touch "Restore" to recover the application program." When the "Restore" button is touched, the operator will be prompted to calibrate the touch screen and enter the date and time. On completion of entering the date and time, the "Recovery" process will automatically restore the files backed up in the "\flashdisk\system" folder to the appropriate folders in the RAM Object Store.

A recovery process can be initiated by conducting a cold boot after software has been installed. Remove the main battery and press the cold boot (FULL RESET) switch located under the main battery. When the unit is powered on, the iPAD will detect the contents of RAM has been lost and lead the operator through the following sequence.

1. The message "The contents of memory has been lost. Touch "Restore" to recover the application program." will be displayed. Respond by touching "Restore".
2. The operator will then be prompted to calibrate the touch screen.
3. The operator will then be prompted to enter the date and time.
4. The unit will automatically restore all files written to the backup directory during the install process.

It is recommended that prior to deploying an application, the effects of the recovery process be tested by the systems integrator.

4 Peripheral Devices

This section provides an overview for each of the peripherals embedded in the iPAD, along with some general guidelines/hints on how to access them. The detail's on the API's that access the peripherals unique to the iPAD are found in section 7 of this document.

4.1 RAM Object Store (Standard CE Device)

The iPAD "RAM Object Store" follows the conventions defined by Microsoft for RAM usage. Up to ~50 Megs of the 64 Megs of RAM can be defined as a "RAM Object Store" (RAM disk). The memory allocation can be adjusted by selecting the "System" icon in the "Control Panel", followed by selecting the "Memory" folder. It is important to set the size of the "RAM Object Store" to provide sufficient room for the application. Not allowing enough memory for the execution of programs will result in the application operating very, very slowly. The MAX "RAM Object Store" size can also be defined in CONFIGHHT.

While every effort has been made to make the RAM Object Store secure, the "RAM Object Store" will always be more vulnerable to data loss than the Flashdisk (FFS) folder. There is a reset switch (warm boot) on the back of the iPAD. In the event of a system lockup, this reset switch will reboot the unit and restart the application without corrupting the "RAM Object Store". Files open at the time of reboot may be missing data. The "RAM Object Store" should only be lost if the voltage level of both the Main Battery and Backup Battery fall below acceptable voltage levels or by pressing the cold boot switch located under the main battery.

4.1.1 Memory Leak

At the initial release of the iPAD two forms of a memory leak exist. The first memory leak occurs each time the iPAD is suspended and the second known memory leak is in the Media Player.

The workaround for the suspend/resume memory leak is to reboot the iPAD periodically to free up the memory lost by the system. The application should call the TeamPadRestartSystem() API at the completion of some nightly maintenance process. If the iPAD is not rebooted at least once every 200 suspend / resume cycles, then a warning message will be displayed. Approximately 400 bytes are lost on each suspend/resume cycle.

If the Media Player is used to play files in multicast, then a Codec error may be displayed. Each time the error is displayed some memory is lost. The only solution is to distribute the files to play in unicast and not multicast.

4.2 Display (Standard CE Device)

The iPAD contains a color screen 240 pixels (wide) by 320 pixels (height).

The back light brightness can be controlled manually via the "SFT"+"ENT" key sequence on the keyboard.

The back light brightness should be set to automatically turn off to conserve battery life.

The back light timer can be set via the "Display" icon in the "Control Panel" or via CONFIGHHT.

4.3 Keyboard

The iPAD has both a mechanical numeric keyboard and an alphanumeric touch screen keyboard. The CE.NET touch screen keyboard can be activated by the "SFT" + "0" key sequence, or by calling the CE API function SipShowIM. An optional full screen keyboard is provided on the iPAD "Documentation & Utilities" CD that provides a larger version of the touch screen keyboard useable without a stylus. While configurable, the "F1" key is used as the default to activate this keyboard.

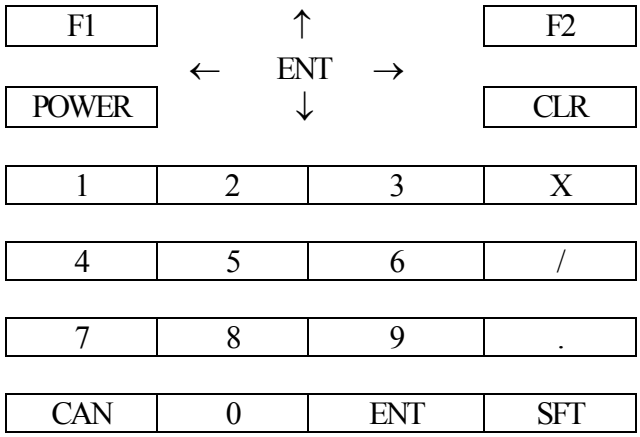
4.3.1 Mechanical Keys

The power key is used to both power the iPAD on (resume) and off (suspend). The power key must be held down for ~1 second before the power key is considered depressed. Once it is determined that the power key has been pressed, the LED at the top of the iPAD will temporarily turn green. It should be noted that the iPAD does not really power off when the power key is depressed as the unit is placed in a non-operating suspended mode. When the power key is pressed a second time, the unit will resume from where it was suspended.

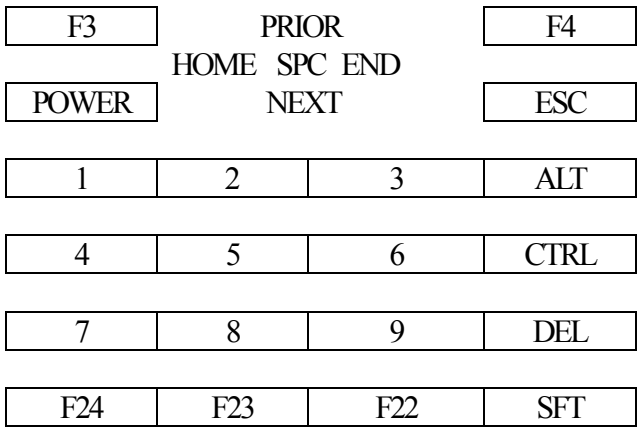
The following table shows the key sequences with special values.

Key Sequence	Function	Function Is Enabled/Disabled By
"SFT"+"0"	Toggles presence of touch screen keyboard on display	CONFIGHHT or Setup Menu
"SFT"+"ENT"	Displays battery status, allows speaker volume control, brightness control, and touch screen calibration	CONFIGHHT or Setup Menu
"SFT"+"CAN"	Display "Setup Menu" on boot, even if an application has already been loaded.	
"SFT"+"X"+tap	While holding the "SFT"+"X" keys, taps to the touch screen will be right mouse clicks.	
"SFT" + tap	The same as SFT + left mouse on a desktop. Used to select a block of consecutive entries	
"SFT"+"/"+tap	The same as CTRL + left mouse on a desktop. Used to select multiple individual entries.	

4.3.1.1 Non Shifted Keyboard



4.3.1.2 Shifted Keyboard



F23 = Touch Screen Keyboard
F22 = System Info

4.3.1.3 Mapping Of Physical Keys To Virtual Key Codes

Normal codes		
Key	Symbol	Value
0	VK_0	0x30
1	VK_1	0x31
2	VK_2	0x32
3	VK_3	0x33
4	VK_4	0x34
5	VK_5	0x35
6	VK_6	0x36
7	VK_7	0x37
8	VK_8	0x38
9	VK_9	0x39
CAN	VK_CANCEL	0x03
ENT	VK_RETURN	0x0D
F1	VK_F1	0x70
F2	VK_F2	0x71
↑	VK_UP	0x26
↓	VK_DOWN	0x28
→	VK_RIGHT	0x27
←	VK_LEFT	0x25
CLR	VK_CLEAR	0x0C
.	VK_DECIMAL	0x6E
/	VK_DIVIDE	0x6F
X	VK_MULTIPLY	0x6A

With Shift code		
Key	Symbol	Value
F23	VK_F23	0x86
Numpad1	VK_NUMPAD1	0x61
Numpad2	VK_NUMPAD2	0x62
Numpad3	VK_NUMPAD3	0x63
Numpad4	VK_NUMPAD4	0x64
Numpad5	VK_NUMPAD5	0x65
Numpad6	VK_NUMPAD6	0x66
Numpad7	VK_NUMPAD7	0x67
Numpad8	VK_NUMPAD8	0x68
Numpad9	VK_NUMPAD9	0x69
F24	VK_F24	0x87
F22	VK_F22	0x85
F3	VK_F4	0x72
F4	VK_F3	0x73
PRIOR	VK_PRIOR	0x21
NEXT	VK_NEXT	0x22
END	VK_END	0x23
HOME	VK_HOME	0x24
ESC	VK_ESCAPE	0x1B
DEL	VK_BACK	0x08
CTRL	VK_CONTROL	0x11
ALT	VK_MENU	0x12

4.3.2 CE.NET Touch Screen Keyboard

The touch screen can be used to extend the iPAD's mechanical numeric keyboard. An application can either develop its own custom keyboard or use the default keyboard that comes with the iPAD. The "SFT"+"0" keys can be used to toggle the presence of CE.NET's touch screen keyboard on the display. The presence of this embedded touch screen keyboard can be programmatically controlled by the Windows CE SipShowIM function. The use of the "SFT"+"0" keys are particularly important to applications that do not control the presence of the touch screen keyboard from within the application and assume the presence of an alphanumeric keyboard.

The size of the embedded touch screen keyboard can be controlled by a parameter in the "Options" section of the "Input Panel" in the "Control Panel". Two different touch screen keyboard layouts exist, however both embedded keyboards require the use of a stylus. In environments where the use of gloved hands is required without a stylus, the application should define its own keyboard to take advantage of the entire screen or use the optional full screen keyboard.

4.3.3 Full Screen Touch Keyboard

An alternate full screen keyboard is provided on the "iPAD Documentation & Utilities" CD. This touch screen keyboard (VKBD.EXE) uses the entire display for the keyboard function allowing the user to input alphanumeric data without using a stylus. Once the data has been assembled in the edit box at the top of the keyboard the data is sent to CE.NET's keyboard buffer. This keyboard can be used with both application and CE.NET functions.

4.3.3.1 VKBD.EXE Configuration

The On-Screen keyboard requires the installation of three files. These files can be found on the "iPAD Documentation & Utilities" CD under the \CODE_SAMPLES\UTILS\VKBD folder.

1. VKBD.EXE is the On-Screen Keyboard utility.
2. VKBDHOOK.DLL is a supporting DLL.
3. VKBD.REG contains the registry entries for On-Screen Keyboard utility.

4.3.3.2 VKBD.EXE Registry

The On-Screen Keyboard utility requires mandatory entries in the registry for the supporting DLL utility (VKBDHOOK.DLL) and optional entries for the On-Screen keyboard utility (VKBD.EXE).

VKBDHOOK.DLL

```
[SOFTWARE\Fujitsu\ExtHook\Hook1]
REG_DWORD;Order;1
REG_SZ;Entry;Hook1
REG_SZ;DLL;\FlashDisk\VKBDHOOK.dll
END
```


The On-Screen supporting DLL requires three mandatory registry entries. The only entry that should be modified is the DLL value. This entry defines the path of the DLL. Changing the other entries may cause the On-Screen Keyboard utility to not work properly.

VKBD.EXE

```
[TEAMPADKEY\VKBD]
REG_SZ;LaunchApp;\FlashDisk\VKBD.EXE
REG_BINARY;LaunchKey;
70
REG_BINARY;Drag;
01
REG_BINARY;EditBox;
01
REG_BINARY;EditColor;
FF 00 00
REG_BINARY;ButtonColor;
00 00 FF
END
```

The On-Screen Keyboard has optional registry entries. One or all can be included in the registry update. If a registry entry can't be found, then the default settings will be used.

LaunchApp: Defines the location of the On-Screen Keyboard utility (VKBD.EXE). By default, the location is the root of the \FlashDisk with an executable name of VKBD.EXE.

LaunchKey: Defines the key to start the On-Screen Keyboard utility. This key is also used as a toggle to minimize and maximize the On-Screen keyboard window. The key definition is a binary value equal to the Virtual key code generated through the Windows CE operating system. By default, this value is the F1 key (VK_F1 or hex 70).

Drag: Defines if the On-Screen keyboard can be dragged by its application title bar. A value of zero disables the ability to drag the window. A non-zero value allows the window to be dragged. By default, this value is no-drag (ie. 00).

EditBox: Defines if an edit box appears on the On-Screen keyboard utility. A value of zero disables the edit box. A non-zero value enables the edit box. By default, this value is edit box enabled.

The major difference between the two modes is how the data is sent to the focused window. When the Edit box is disabled, the key strokes/presses are sent immediately to the windows in focus. When the Edit box is enabled, valid edit key strokes/presses are stored in the edit box where they can be modified through the On-Screen keyboard utility before sending with the "Enter" or "Done" keys.

When the edit box control is disabled, context sensitive editing can be performed by the window in focus. For example, only numeric input. When the edit box is disabled, the On-Screen keyboard utility has no visibility to the actual data requirements of the focused window. However, the drawback to Edit Box disabled is that you can't see the data entered since the On-Screen keyboard covers the entire LCD. The maximum data accepted when the Edit Box is enabled is 300 characters.

EditColor: Defines the Text Color for the entered key strokes/presses shown in the Edit Box. By default the color is Red (FF 00 00). The first hex value (00-FF), defines the intensity of Red. The second hex value (00-FF), defines the intensity of Green. The third hex value (00-FF) defines the intensity of Blue. If the user would like black, then the color scheme should be changed to (00 00 00). There is no ability to change the background for the Edit Box, which is set to white. The best way to define a custom color is to use the Microsoft Paint utility. This utility will give you a RGB value for various color combinations.

ButtonColor: Defines the Text Color for the key press buttons. By default the color is Blue (00 00 FF). The first hex value (00-FF), defines the intensity of Red. The second hex value (00-FF), defines the intensity of Green. The third hex value (00-FF) defines the intensity of Blue. If the user would like black, then the color scheme should be changed to (00 00 00). There is no ability to change the background for the Buttons, which is set to white. The best way to define a custom color is to use the Microsoft Paint utility. This utility will give you a RGB value for various color combinations

4.3.3.3 VKBD.EXE Setup

1. Modify, if necessary, the Registry entries contained in VKBD.REG file as described above. You can use a text editor program such as Notepad or Edit.
2. Transfer the files (VKBD.REG, VKBD.EXE, VKBDHOOK.DLL) over to the iPAD using ActiveSync or the Install Application process.
3. Import the Registry entries either through TPREGEDIT.EXE or through CONFIGHHT.
4. Reboot the iPAD. This will load the On-Screen DLL procedure. When the Launcher key is pressed, the DLL will start the On-Screen keyboard program VKBD.EXE. This is assuming that the Registry file has been properly configured.

4.3.3.4 VKBD.EXE Using the On-Screen Keyboard

The utility has eight permanent buttons that are available on each of the four screen templates. The close window icon at the top of the application title is also available on each screen ("X").

1. "Bks" is the backspace key.
2. "Space" is the SPACE key
3. "Clear" will clear the Edit Box Window.
4. "<-" will scroll to the prior On-Screen input panel.
5. "Hide" will place a "*" in place of the actual key in the edit box. This can be used to protect sensitive data.
6. "Enter" will send the Edit Box text to the Window with focus with an appended ENTER and terminate the utility
7. "Done" will send the Edit Box text to the Window with focus as is (no ENTER appended) and terminate the utility.
8. "->" will scroll to the next On-Screen input panel.

The utility also makes use of the physical hardware keys. The definitions below assume that the key described hasn't been re-mapped to another virtual key code value.

1. Defined Launcher key. Pressing this key will launch the On-Screen keyboard utility (if not already running). If the application is running, then pressing the Launcher key will toggle the application from a minimized to maximized state.
2. "CLR" will clear the Edit Box Window. Similar to the "Clear" button.

3. "CAN" will close the window (similar to close window "X").
4. "ENT" will send the Edit Box text to the Window with focus with an appended ENTER and terminate the utility. Similar to "Enter" button.
5. Left Arrow (on the Wheel) will move to caret in the Edit Box one character to the left.
6. Right Arrow (on the Wheel) will move the caret in the Edit Box one character to the Right.
7. Up Arrow (on the Wheel) will scroll to the prior On-Screen input panel. Similar to the "<-" button.
8. Right Arrow (on the Wheel) will scroll to the next On-Screen input panel. Similar to the "->" button.

The following are Screen Captures from the On-Screen Keyboard utility.

Upper Case Alpha Keyboard:

Lower Case Alpha Entry:

Numeric and Special Characters:

Function Keys and Special Characters:

The Function keys (F1 through F10) are considered delimiter keys. When pressed, the On-Screen Keyboard utility will send the Edit Box text along with the Function key pressed. The keyboard utility will also terminate.

4.4 Scanner

The iPAD comes equipped with a 1D laser scanner. The Symbol 923 scanner is connected to the COM3 port of the iPAD. The features of the Symbol 923 scan engine can be found in the Symbol 923 Integration Guide found in the \Documents\SE923_Scanner directory of the "iPAD Documentation & Utilities" CD.

Two interfaces to the scanner are provided. The first is via a wedge that places scanned data in the keyboard buffer and the second is via a set of API's.

4.4.1 Scanner Wedge

Once the scanner wedge is started, any data scanned will be inserted into the keyboard buffer. The data will be presented to the application or CE.NET as though it was typed on a keyboard. As the scanner wedge operates without the applications knowledge (no programmatic interface), no coding needs to be implemented to use the scanner. The negative to this approach is that the application has little control over preventing scan data from being entered into a field that should not except scan data. It is possible for the scanner to send a prefix and suffix code around the data to allow the application to discarded the scan data if not appropriate.

The scanner wedge (SCANWEDG.EXE) is located on the "iPAD Documentation & Utilities" CD in the folder "\code_samples\utils\wedges\scanwedg". The scanner wedge calls the CESCAN.DLL discussed in Section 4.4.2. When the wedge opens the scanner, the scanner is configured via the instructions in SCANPARM.DAT. If the SCANNER.WAV file is found then it is played on a successful scan. The SCANNER.WAV file must either be located in the root of the flashdisk folder or in the path defined in the registry. The SCANPARM.DAT file must be located in the same directory as the SCANNER.WAV file. The CESCAN.DLL must either reside in the directory of SCANWEDGEXE or in the \Windows folder.

4.4.1.1 Scanner Wedge Configuration

Configuration of the Scanner Wedge can be done through an optional text file named SCANWEDG.CFG. The file is only necessary if you wish to change the default Scanner Wedge options. In addition, a Registry Entry is provided to allow for a file path different than the default, which is the root of the FlashDisk (i.e. \FlashDisk\).

Registry Entries

Registry Entries are located in the HKEY_LOCAL_MACHINE\TEAMPADKEY\SCANWEDGE key. The use of these keys are optional.

WedgePath = key will allow you to change the source and destination of files.
WedgeState = key will allow you to Activate, Suspend or Terminate the Scanner Wedge. On startup of the Scanner Wedge, the state is set to ACTIVE, and will remain in this state until the WedgeState is changed by an outside application or manual set through TPREGEDIT. When the state is changed to SUSPEND, the Scanner Wedge will close the Scanner and remain executing looking for the state to change to ACTIVE or TERMINATE. If the state changes to ACTIVE, the Wedge will open the scanner and make scanning possible. When the state is changed to TERMINATE, the Scanner Wedge will terminate and exit. Note that the changing of states will occur after the current scanner read timeouts. By default through the Scanner Wedge, this can be anywhere from immediately to approximately10 seconds.

Configuration File Entries

The optional SCANWEDG .CFG file allows for further customization to the operations of the Scanner Wedge utility. The default location is the root of the FlashDisk. However, this can be changed by using the Registry Entry WedgePath as described above. The file is read on each scan attempt. This allows for re-configuration while the SCANWEDG program is running. Changes to standard default options are the only entries that need to be made and must match the option string before a replacement to the default occurs. The general rule for replacing a default is OPTION STRING=NEW VALUE followed by a carriage return or line feed or end-of-file. The supported operations are as follows.

Option String	Length Range	Default Value
SCAN OUTPUT =	1-20 Alphanumeric	KEYBOARD
SCAN RETRIES=	1-4 Numeric (1-9999)	10
SCAN LED=	1-4 Numeric (0-9999)	100

SCAN OUTPUT=KEYBOARD, FILE, BOTH

KEYBOARD output will stuff the Data into the keyboard buffer.

FILE will output the Data to SCANDATA.DAT located in the root of the FlashDisk or configured path. Mainly used to debug Wedge utility. Data written is for the last good scan .

BOTH will output the Data to the Keyboard buffer and write to the MSRDATA.DAT file located in the root of the FlashDisk or configured path.

SCAN RETRIES=

Controls how many attempts before the CESCAN.DLL will timeout. By default, the value is 10, which equates to roughly 10 seconds. If your desire is to always run the Scanner Wedge in the ACTIVE state, it is recommended that you change the retries to a larger number, so you remain in CESCAN.DLL as often as possible so you are ready for any scanning activity.

SCAN LED=

Controls how many milliseconds the GREEN LED will be illuminated on a good SCAN returned from CESCAN.DLL. By default, the value is 100 milliseconds. A value of 0 will disable the LED illumination.

4.4.2 Scanner API's

The CESCAN.DLL, with OPEN, READ, and CLOSE type functions, is provided to access the scanner. In addition to the DLL, a sample program (ScanTest.exe) written in C is provided that demonstrates the loading and calling of these functions

4.4.2.1 OpenSCAN

The OpenSCAN() function will open the COM3 port and program the scanner to the desired functionality. OpenSCAN(Path) has one parameter which indicates the path of the files needed to configure the scanner. If the path is set to NULL then no programming of the scanner will be attempted. If the path is not NULL then the following files will be processed.

1. Parameter File (SCANPARM.DAT)

If the file SCANPARM.DAT is found then the configuration
data of the scanner will be read. If the file SCANEXP.DAT exists, then the configuration parameters read from the scanner will be compared to the contents of file SCANEXP.DAT. If the parameters match, then no attempt will be made to reprogram the scanner. The scanner is already programmed correctly.

If the file SCANPARM.DAT exists and the file SCANEXP.DAT does not, then the assumption is made the scanner needs to be configured. The scanner is first set to the scanners defaults and then the parameters defined in the file SCANPARM.DAT are sent to the scanners flash. The scanners configuration parameters will be re-read and written to the file SCANEXP.DAT.

If the file SCANEXP.DAT exists but does not match the parameters read from the scanner, then the scanner is configured as though the file SCANEXP.DAT did not exist.

Care should be taken not to delete the file SCANEXP.DAT unless reconfiguring the scanner is desired. The flash memory in the scanner has a limited write life.

A HEX editor is required to build the SCANPARM.DAT file.

2. SCANEXP.DAT

This file is used by the DLL to maintain the expected scanner configuration data. In this case the term expected means the state of the scanner after being set to the factory defaults an applying the parameters defined in the file SCANPARM.DAT. After a successful OPEN, the file SCANEXP.DAT will contain the parameters actually set in the scanner. If desired, the application can compare the file SCANEXP.DAT with a file containing the contents of a SCANEXP.DAT captured during testing. This will confirm that the scanner has been set to the expected configuration. As the format of the SCANPARM.DAT file is the same as the format of the file SCANEXP.DAT file, it can be helpful to review the SCANEXP.DAT file prior to building your own SCANPARM.DAT file. The format of the file is the PARM data defined in the Symbol document minus the protocol information (Param, value, Param, value).

3. SCANLOG.DAT

If this file exists in the defined path then the data sent to and received from the scanner, along with some debug data, will be logged in the file SCANLOG.DAT file. Each time the OpenSCAN() is executed the file will be overwritten. If no SCANLOG.DAT file is found in the path then no logging will occur. If the path was set to NULL then the root is assumed and logging can still be performed. The entries in the log labeled CMD /, RESP/, and DATA/ represent the data sent to and from the scanner. The entries labeled INFO/ are for the use of FJ-ICL. A HEX editor is required to read the SCANLOG.DAT file.

NOTE: None of the Serial Interface parameters (0x9c,0x9e,0x9f,0xee,0x9b,0x9d,0x6e,0xef) or the Event Reporting parameters (0xf0–0x00, 0xf0-0x02, 0xf0-0x03) can be altered via SCANPARM.DAT or the CESCAN.DLL may cease to function.

NOTE: OpenSCAN() can return two non-zero result codes. One to indicate an error indicating the OpenSCAN was not successful and there is no reason to continue or issue the CloseSCAN() function. The other error code indicates that data

can be read from the scanner however the attempt to program the
scanner failed. The CloseSCAN() function must be issued when appropriate to close the scanner. The scanner's behavior is now unpredictable.

4.4.2.2 ReadSCAN

The ReadSCAN() function will wait for and return data from the scanner. The ReadSCAN() function will wait for the number of seconds specified in the first calling parameter before returning a SCAN_OK and zero bytes read status. The ReadSCAN() function will return immediately if barcode is available.

Since the scanner will buffer barcodes, it is recommended that the scanner be configured to return a suffix (terminator), so the application can delineate one barcode from another. If multiple barcodes are scanned prior to the read, both will be returned in a single read.

4.4.2.3 CloseSCAN

The COM3 port is closed along with the SCANLOG.DAT file if logging is enabled.

4.5 Integrated Magnetic Swipe Reader (MSR)

The iPAD comes with a Track 1 and 2 Magnetic Swipe Reader (MSR). Card reading is provided via the Omron 3S4YR-HFNR series MSR. The Omron card reader is connected to the COM2 port of the iPAD. The features of the Omron MSR can be found in the Omron reference guide located in the \Documents\Magnetic_Swipe_Reader directory of the "iPAD Documentation & Utilities" CD.

4.5.1 MSR Wedge

Once the magnetic stripe reader (MSR) wedge is started, any data swiped will be inserted into the keyboard buffer. The data will be presented to the application or CE.NET as though it was typed on a keyboard. As the MSR wedge operates without the applications knowledge (no programmatic interface), no coding needs to be implemented to use the MSR. The negative to this approach is that the application has little control over preventing swiped data from being entered into a field not expecting swiped data. It is possible for the MSR to send a prefix and suffix code around the data to allow the application to discarded the swiped data if not appropriate.

The scanner wedge (MSRWEDGE.EXE) is located on the "iPAD Documentation & Utilities" CD in the folder "\code_samples\utils\wedge\msrwedge". The scanner wedge calls the CEMSR.DLL discussed in Section 4.4.2. If the MSR.WAV file is found, it is played on a successful scan. The MSR.WAV file must either be located in the root of the flashdisk folder or in the path defined in the registry. The CEMSR.DLL must either reside in the directory of MSRWEDGE.EXE or in the \Windows folder.

4.5.1.1 MSR Wedge Configuration

Configuration of the MSR Wedge can be done through an optional text file named MSRWEDGE.CFG The file is only necessary if you wish to change the default MSR Wedge options. In addition, a Registry Entry is provided to allow for a file path different than the default, which is the root of the FlashDisk (i.e. \FlashDisk\).

Registry Entries

Registry Entries are located in the HKEY_LOCAL_MACHINE\TEAMPADKEY\MSRWEDGE key. The use of these keys are optional.

WedgePath = key will allow you to change the source and destination of files.
WedgeState = key will allow you to Activate, Suspend or Terminate the MSR Wedge. On startup of the MSR Wedge, the state is set to ACTIVE, and will remain in this state until the WedgeState is changed by an outside application or manually set through TPREGEDIT. When the state is changed to SUSPEND, the MSR Wedge will close the MSR and remain executing looking for the state to change to ACTIVE or TERMINATE. If the state changes to ACTIVE, the Wedge will open the MSR and make card reading possible. When the state is changed to TERMINATE, the MSR Wedge will terminate and exit. Note that the changing of states will occur after the current MSR read timeouts. By default through the MSR Wedge, this can be anywhere from immediately to approximately10 seconds.

Configuration File Entries

The optional MSRWEDGE.CFG file allows for further customization to the operations of the MSR Wedge utility. The default location is the root of the FlashDisk. However, this can be changed by using the Registry Entry WedgePath as described above. The file is read on each swipe attempt. This allows for re-configuration while the MSRWEDGE program is running. Changes to standard default options are the only entries that need to be made and must match the option string before a replacement to the default occurs. The general rule for replacing a default is OPTION STRING=NEW VALUE followed by a carriage return or line feed or end-of-file. The supported operations are as follows.

Option String	**Length Range**	**Default Value**
TRACK READ=	1-20 Alphanumeric	EITHER
TRACK1 PREFIX=	1-10 Binary (00-FF)	00
TRACK1 SUFFIX=	1-10 Binary (00-FF)	00
TRACK2 PREFIX=	1-10 Binary (00-FF)	00
TRACK2 SUFFIX=	1-10 Binary (00-FF)	00
TRACK SEPARATOR=	1-10 Binary (00-FF)	2C
TRACK TERMINATOR=	1-10 Binary (00-FF)	0D
TRACK OUTPUT=	1-10 Alphanumeric	KEYBOARD
MSR RETRIES=	1-4 Numeric (1-9999)	10
MSR LED=	1-4 Numeric (0-9999)	100

Allowed Options:

TRACK READ=TRACK1, TRACK2, EITHER, BOTH

TRACK1 processes only when a good Track1 Read occurs. Data returned is only Track1 data even if Track2 data is present.

TRACK2 processes only when a good Track2 Read occurs. Data returned is only Track2 data even if Track1 data is present.

EITHER processes only when a good Track1 **or** Track2 Read occurs. Data returned can be Track1 only, Track2 only, or both.

BOTH processes only when a good Track1 **and** Track2 Read occurs. Data returned is Track1 and Track2 data.

TRACK OUTPUT=KEYBOARD, FILE, BOTH

KEYBOARD output will stuff the Data into the keyboard buffer.

FILE will output the Data to SCANDATA.DAT located in the root of the FlashDisk or configured path. Mainly used to debug Wedge utility. Data written is for the last good scan .

BOTH will output the Data to the Keyboard buffer and write to the MSRDATA.DAT file located in the root of the FlashDisk or configured path.

MSR RETRIES=

Controls how many attempts before the CEMSR.DLL will timeout. By default, the value is 10, which equates to roughly 10 seconds. If your desire is to always run the MSR Wedge in the ACTIVE state, it is recommended that you change the retries to a larger number, so you remain in CEMSR.DLL as often as possible so you are ready for any Swipe activity.

MSR LED=

Controls how many milliseconds the GREEN LED will be illuminated on a good card SWIPE returned from CEMSR.DLL. By default, the value is 100 milliseconds. A value of 0 will disable the LED illumination.

Sample Configurations

Example 1:
No file Present will use the Defaults. Defaults are EITHER tracks, Separator is a comma (2C), Terminator is a Carriage Return (0D). Output is to the Keyboard Buffer.

Actual Ouput:
Track1Data,Track2Data,CR

Example 2:
TRACK READ=BOTH
TRACK1 PREFIX=54313A
TRACK2 PREFIX=54323A
TRACK SEPARATOR=00
TRACK TERMINATOR=00
TRACK OUTPUT=FILE

Will only process when both Track 1 and Track 2 are read successfully. A Prefix is added to Track1 and Track 2. No Suffix added, No separator added, No terminator added. Data will be outputted to the MSRDATA.DAT file.

Actual Output:
T1:Track1DataT2:Track2Data

4.5.2 MSR API's

The CEMSR DLL, with OPEN, READ, and CLOSE functions, is provided to access the MSR. An additional function call will return the version of the CDMSR DLL. In addition to the CEMSRDLL, a sample program (MSRTEST.EXE) written in C is provided to demonstrate the loading and calling of these four functions in the DLL. For a definition of the function prototypes, parameters and return codes, please refer to the header include file (MSR.H) for MSRTEST.EXE project.

The MSRDLL will return Track 1 and/or Track 2 data. The Omron reader will not read or return Track 3 data.

4.5.2.1 OpenMSR

The OPEN function will open the COM2 port and prepare the DCB. In addition, if a log file (MSRLOG.DAT) is found in the object store, i.e. "\MSRLOG.DAT", session logging will be performed. This log file will contain messages and status information in the event debugging a MSR session is necessary. Generally, logging should only be performed as a debug tool. A return code of 0 indicates the open was successful. A non zero return code indicates a failure.

4.5.2.2 ReadMSR

The Read function will perform all the initialization and preparatory work required to get the MSR ready for operation. The function takes 3 parameters as input. Parameter 1 defines how many attempts should be tried before an error is return to the calling routine. The ReadMSR function will always attempt one connection to the card reader. In the event no data is available (i.e. No swipe has been performed), the function will Sleep and try again up to the desired attempts. Each query is roughly 500 milliseconds in duration. For example, a parameter 1 setting of 5, will timeout after around 2-3 seconds. Parameter 2 is a pointer to the buffer where the read card data is to be stored. The calling function should define this character buffer with a large enough length to hold the desired data. The third parameter is a length count for the size of the actual read MSR data. A return code of 0 indicates the read was successful. A non zero return code indicates a failure. The data in the return buffer is not filtered with the exception that the protocol and CRC characters are stripped.

4.5.2.3 CloseMSR

The COM2 port is closed along with the MSRLOG.DAT file if logging is enabled. A return value of zero indicates success, while all others indicate a failure.

4.5.2.4 VersionMSR

Returns the MSRDLL version number. The calling routine should define the return buffer with a large enough length to hold the standard Fujitsu Transactions Solution Inc version naming convention ("x.xx.xx").

4.6 Bluetooth

The Bluetooth and 802.11b interfaces will NOT operate at the same time. When the Bluetooth interface is turned on the 802.11b interface is automatically turned off. The program "btstart.exe" in the \windows folder will power the Bluetooth interface on. The control panel applet "Bluetooth Device Properties" can be used to discover other Bluetooth devices.
As an alternative to starting "btstart.exe", an API is provided in POWDLL.LIB which will power on and off the Bluetooth interface.

4.7 802.11b

The 802.11b RF is accessed via standard Microsoft API's. The "iPAD Documentation & Utilities" CD contains three examples of programming through a sockets (Winsock) interface. The following sections describe their functionality and use.

4.7.1 CEPING Project

CEPING is an example program using the WINSOCK interface. After the iPAD has been configured, CEPING can be used to check a TCP/IP network connection. The application will prompt for either a Peer IP address or a computer name. In addition, the number of ping attempts and TTL (Total Time to Live) can also be entered.

CEPING is executed by pressing the PING button. A ping echo request is sent to the entered Peer and a status is returned.

0 = Ping Ok (with the Round Trip Time in milliseconds).
1 = Ping Error on gethostbyname().
2 = Ping Error on gethostbyaddr().
3 = Ping Error invalid Address Name
4 = Ping Error on IcmpCreateFile()
5 = Ping Error on IcmpSendEcho()

The sample code and project can be found on the "iPAD Documentation & Utilities" CD under the \Code_Samples\Winsock\CEPING folder. This example was found on the Internet at a shareware site for CE utilities.

4.7.2 SOCKTEST Project

SOCKTEST is a set of example programs using the WINSOCK interface. SOCKSRVR is a WIN32 application that will run on the server (PC). SOCKCLNT is a WINCE application that will run on the client (iPAD).

After the iPAD has been configured, the 2 sockets applications can be used to check a TCP/IP network connection. The Handheld client application will send a message to the PC server application. On receipt of the message, the PC application will display and send a message back to the HH client application after acceptance of the message box. The HH application will display the return message and then exit on acceptance of the message box. These test suites are an example of a full send and receive between two connected devices.

The server SW must be started first otherwise the client side will timeout out the connection. You must select an open channel for the Winsock connection.

On the client side, the user must enter the Peer's computer name (i.e. not IP address). In addition, the channel number must be the same as entered on the server side.

Error conditions are displayed to the user through message boxes and on acceptance will terminate the application.

The sample code and project can be found on the "iPAD Documentation & Utilities" CD under the \Code_Samples\Winsock\SOCKTEST folder. This example was found on the Microsoft Embedded Visual C++ Toolkit's HELP. Supporting documentation can be found under the WINSOCK keyword description.

4.7.3 CHATTEST Project

CHATTEST is a set of example programs using the WINSOCK interface. CHATSRVR is a WIN32 application that will run on the server (PC). CHATTER is a WINCE application that will run on the client (iPAD).

After the iPAD has been configured, the 2 sockets, application can be used to check a TCP/IP network connection, simulating a chat session. The Handheld client application will send a message to the PC server application. On receipt of the message, the PC application will display and send the same message back to the HH client application's main window.

The server SW must be started first otherwise the client side will timeout out the connection. You must select an open channel for the Winsock connection.

On the client side, the user must enter the Peer's computer address (i.e. not computer name). In addition ,the channel number must be the same as entered on the server side.

The sample code and project can be found on the "iPAD Documentation & Utilities" CD under the \Code_Samples\Winsock\CHATTEST folder. This example was found on the Microsoft Embedded Visual C++ Toolkit's HELP. Supporting documentation can be found under the CHATTER/CHATSRVR keyword description.

5 Utilities and Code Samples

The following sections discuss a set of code samples available on the "iPAD Documentation & Utilities" CD. These code samples can be used as a starting point for applications developers needing similar functions.

5.1 Signature Capture

The "iPAD Documentation & Utilities" CD contains an example of how to capture an ink drawing. This sample utility can be used as a means to capture a signature. In order to give the largest screen area possible, the drawing is displayed and captured in "landscape" mode. Once the SAVE button is pressed, the image is rotated 90 degrees and saved to a bitmap file named SIG.BMP (default location is root).

The sample code and project for TestRichInk.EXE can be found on the "iPAD Documentation & Utilities" CD under the \Code_Samples\SIGCAP folder.

5.2 Desktop Lockdown

In some environments it may be desired to have a set of ICON's on the desktop that start a specific set of approved applications. In this case access to Windows Explorer via the desktop must be prevented. Two sample programs are provided on the "iPAD Documentation & Utilities" CD to help achieve this. The REMOVEMY.EXE program edits the registry to remove the "My Computer" ICON on the next boot. It is important to remove the "My Computer" ICON to prevent navigation to other programs. The REMOVEBIN.EXE program removes the "Recycle Bin" from the desktop. These two programs are found in the "\Code_Samples\utils" directory of the "iPAD Documentation & Utilities" CD.

Removing "My Computer" along with preventing access to the Task Bar (see next section) help prevent the end user from running unauthorized programs.

5.3 Task Bar

While the iPAD can be configured to prevent access to the Task Bar, it is sometimes desired to programmatically grant control of the Task Bar to the end user.

Three sample programs having to do with controlling access to the Task Bar can be found on the "iPAD Documentation & Utilities" CD. The HIDETBAR.EXE program is a sample of how to programmatically hide the Task Bar. The SHOWTBAR.EXE program is a sample of how to programmatically show the Task Bar. The TASKBAR.EXE program is a sample of how to place the access of the Task Bar under password control. These programs can be found on the "iPAD Documentation & Utilities" CD in folder \Code_Samples\Utils.

6 Setting Up Your Development Environment

Microsoft's "eMbedded Visual C++ 4.0" development product must be used to develop code for the iPAD, to which the Fujitsu iPAD SDK must be added. The "embedded Visual C++ 4.0" tools cannot be loaded onto the same disk partition as "embedded Visual Tools 3.0". At the writing of this document the "eMbedded Visual C++ 4.0" product could be downloaded free from Microsoft, or a CD obtained by paying the shipping and handling charge of $11.95. The "eMbedded Visual C++ 4.0" product could (at the writing of this document) be ordered or downloaded from the following Microsoft's web site:

http://www.microsoft.com/downloads/release.asp?ReleaseID=37662&area=search&ordinal=3

SP1 must be added to "eMbedded Visual C++ 4.0" and the service pack can be found at:

http://www.microsoft.com/downloads/release.asp?ReleaseID=40558&area=search&ordinal=2

Initially the iPAD only supports applications written in C++. The comments in this section referencing the loading of Microsoft products are meant as hints to be used when following the Microsoft instructions. Knowledge on how to install and use the Microsoft products is outside the scope of this document.

The following Microsoft products are required to develop for the iPAD, along with the knowledge on how to use them:

1. Microsoft® Windows 2000 (SP2) or Windows XP on the development PC.
2. Microsoft® Windows® CE.NET Operating system (Provided in the iPAD)
3. Microsoft® eMbedded Visual C++ 4.0 (SP1) on the development PC.
4. Microsoft® Activesync 3.5 on the development PC.
5. iPAD SDK located on the "iPAD Documentation & Utilities" CD.

6.1 Embedded Visual C++ 4.0

1. Before you start, make sure you have write access to the Registry. You may need to be logged in as administrator to run the install.

2. Place the "Microsoft eMbedded Visual C++ 4.0" CD into the CD drive. The install program should start automatically.

3. The install program will ask what components you wish to install. Make sure you both "eMbedded Visual C++ 4.0" and "Standard SDK for Windows CE.NET" checked.

4. When prompted for permission to install "Windows CE Platform Manager", respond "YES".

5. The install program will ask what components you wish to install. Make sure both "eMbedded Visual C++ 4.0" and "Common Components" are selected.

6. Respond with "Complete" during the Standard SDK installation when asked "Complete" or "Custom" install.

7. Add "embedded Visual C++ 4.0 SP1.

8. Install ActiveSync 3.5. This software is needed to ship software to the iPAD from your PC via a cable and/or for remote debugging.

5. Install the iPAD SDK (iPAD_US.EXE) located in the "iPAD Documentation & Utilities" CD in directory "iPAD SDK".

6.2 Installing the iPAD SDK for VC++

1. Be sure to uninstall any existing iPAD SDK's if you are upgrading to a newer version of the SDK. Be sure you have write access to the registry. This may require administrator privileges.
2. Execute the Windows install program "iPAD2_US_SDK.MSI" located in the \eVC_SDK directory of the "iPAD Documentation & Utilities" CD.
3. Start the iPAD2_US_SDK program from a folder on the hard drive.
4. Accept the license agreement.
5. Select the "Complete" install versus a "Custom" install.

6.3 Using ActiveSync to move files to the iPAD

Microsoft's ActiveSync 3.5 provides a method of exchanging data between a PC and an iPAD. ActiveSync is especially useful when used in conjunction with Microsoft's "eMbedded Visual C++ 4.0" for remote debugging.

While it is possible to use ActiveSync to move files to and from a PC, the use of ActiveSync will bypass the file recovery logic achieved when loading the iPAD via the CONFIGHHT file. At the writing of this document the Microsoft "ActiveSync" product could be downloaded from the following Microsoft Web site:

ActiveSync3.5 –
http://www.microsoft.com/mobile/pocketpc/downloads/activesync/as-dl35.asp?submit1=I+Accept+%3E%3E

The ActiveSync product should only be used to install files during the development process and CONFIGHHT should be used to install files during product deployment.

Note that the first time the iPAD is seen by the PC, the PC will indicate it "Found New Hardware" and identify a "USB Device" was found. Select the directory that contains the two iPAD USB driver files and select "OK". The USB device driver files are found in the "USB_DRV" folder of the "iPAD Documentation & Utilities" CD.

Note that if the iPAD is reboot or powered on while in the cradle, the desktop PC will report "USB Device Not Recognized" Remove and then replace the iPAD from the cradle to clear this message and to continue.

6.3.1 Starting the iPAD ActiveSync Client

There are four methods of starting the ActiveSync Client.

1. Start ActiveSync from the "Install the application" entry of the "Setup" menu.
2. Start ActiveSync from the "Programs" folder when in debug mode.
3. Start ActiveSync (Repllog) from the "Run" entry from the "Start" icon.
4. Start ActiveSync by navigating to the Windows folder and starting the program "Repllog".

6.3.1.1 Start ActiveSync from the "Install the Application" function

The "Install the Application" function is the second entry in the "Setup" menu. The "Setup" menu can be reached two ways. The first time the iPAD is powered on the iPAD will enter the "Setup" menu automatically. The "Setup" menu can also be accessed by the "SFT" + "CAN" keys, after resetting the iPAD via the reset switch (warm boot) on the back of the iPAD.

6.3.1.1.1 "Setup" menu from initial power on

Place a fully charged battery in the iPAD and power on the unit. Be sure to close the lock on the main battery door when installing the battery or the unit will not power on. Detailed information on this process can be found in the "iPAD Quick Reference Guide". The first time the unit boots a message will be displayed indicating that no application has been loaded. Press "SFT"+"CAN" to continue.

The operator will then be prompted to calibrate the touch screen. Touch each position indicated by the plus sign. When all five locations have been touched, press the "ENT" key to continue. Detailed information on this process can be found in the "iPAD System User's Guide".

The operator will then be prompted to enter the current date and time. Enter the date and time or depress "SFT"+"CLR" to bypass. Detailed information on this process can be found in the "iPAD System User's Guide".

The "Setup" menu should now be displayed.

Select #2 or "Install the application" and the user will be prompted for whether the system should be loaded via ActiveSync(USB) or from the LAN (802.11b). Note: the LAN install does not use ActiveSync.

Select "ActiveSync(USB)" and select "Next".

A drop down will appear for the selection of the desired hardware interface to be used for ActiveSync. At this time only the USB entry exists and no change is possible. Check the "Install to default path of Cab" if you are installing a CAB file and do not wish to change the destination directory.

Once the USB cable is connected to the PC, and ActiveSync has been started on the PC, select "Connect" to begin the connection to the PC. Once the popup "Connected to Host" is displayed a dialog box will appear on the desktop asking if a partnership is desired. Select NO, which will allow the connection with multiple iPAD's. Use "Windows Explorer" on the desktop to create the folder "Install" in root of the iPAD. Copy the desired files to the "Install" folder via "Windows Explorer".

Note that CEAppMgr is not able send files to the iPAD while it is in the "Setup Menu". CEAppMgr can only be used when the iPAD is at the desktop in debug mode.

Once the file transfer is complete, select "Disconnect" and then "Next". At this time the iPAD will be ready to begin an application install via the CONFIGHHT file. If no CONFIGHHT file was transferred you can terminate the application install by selecting the X in the top right corner of the screen then select "Enter Debug". If a CONFIGHHT file was transferred, selecting "Next" will begin the application install process. When the installation is complete select "End" to return to the "Setup" menu and then select "End" at which time the unit will power off. When the iPAD is powered back on, the application will be started or the desktop will be displayed.

6.3.1.2 "Setup" menu via key sequence

The "Setup" menu is only presented to the operator once. Provided that the charge in both batteries is not lost, the "Setup" menu will not be displayed on subsequent reboots. The iPAD can be forced to enter the "Setup" menu if the "SFT"+"CAN" keys are held simultaneously while the system is rebooted.

It should be noted that every time the "Setup Menu" is exited with the END function, the contents of the RAM based registry is pushed to non-volatile memory. Both the changes made to the registry during this "Setup Menu" session and the changes made to the registry between the first and this "Setup Menu" sessions will be saved.

6.3.1.3 Start ActiveSync via the "Programs" folder in debug mode.

The ActiveSync client can be started from the "Programs" folder if the iPAD entered the debug mode when exiting the "Setup" menu.

If an application was installed then the ActiveSync client will not appear in the "Programs" folder.

6.3.1.4 Start ActiveSync via "Run" from the "Start" icon.

The program "Repllog" can also be started via the "Run" function from the "Start" icon in the task bar. The "Run" function is of particular use when establishing an 802.11b connection.

Select "Run" and browse to the \windows folder. Select "Repllog"

The use of "Run" is particularly helpful in establishing an ActiveSync connection via 802.11b. After creating a partnership via the USB cable, start "Repllog" via "Run" but with the /remote switch added. This will establish an ActiveSync session via 802.11b. A network connection must be previously established.

6.3.1.5 Start ActiveSync by navigating to the Windows folder.

Repllog is the Microsoft CE client for the ActiveSync product.

Use either "Windows Explorer" or "My Computer" to locate the program "Repllog" in the \Windows folder. Double tap the Repllog program to start the ActiveSync client.

7 API's

7.1 Scanner API's

CESCAN.DLL provides a simple Open, Read, Close type interface to the scanner The CESCAN.DLL must first be loaded into memory before the Open, Read, Close functions can be called. A method of loading and unloading the CESCAN.DLL and can be found in the program SCANTEST. The source to the SCANTEST program can be found on the "iPAD Documentation & Utilities" CD in the \Code_Samples\SCANTEST directory.

7.1.1 Open Scanner

The function OpenSCAN(UserPath) is used to both open the scanner and set the scanner to the desired configuration. If the file SCANPARM.DAT is found in the UserPath, then OpenSCAN() will validate that the scanner is configured correctly and configure it if needed. Providing a path name in the open results in the open taking longer, as the current configuration of the scanner must be requested and the results compared to the file SCANEXP.DAT (if the file SCANPARM.DAT exists). If the application wishes to open and close the scanner often, then the open should provide a NULL path that results in the bypassing of the validation of the scanner configuration. The open with a path name should be conducted occasionally to insure the scanner is configured correctly, perhaps when the application is initially started. The path name must be provided if the following event has occurred:

- New Installation.
- The iPAD has been returned from repair.
- A new SCANPARM.DAT file was downloaded.

Note: The scanner is powered once the scanner is opened. While the scanner will enter a low power mode if not used, it will remain ready to scan until a CloseSCAN is issued. To achieve the best battery life close the scanner when not needed.

More information about the files associated with OpenSCAN can be found in Section 4.4.2

1. Files to be used

   SCAN.H
   CESCAN.DLL

2. Function

   DWORD OpenSCAN(TCHAR * UserPath)

3. Parameter's

   UserPath: The full path name to the directory where the files SCANPARM.DAT, SCANEXP.DAT, and SCANLOG.DAT will be located. The path name must be in Unicode format.

   NULL (do not program scanner, logging to root if SCANLOG.DAT exits in root)
   Path (Unicode string of full path name to SCANPARM.DAT minus the term SCANPARM.DAT).

4. Returned value

   0 = SCAN_OK – Open successful, scanner configured if needed.
   1 = SCAN_ERROR – The scanner was not opened due to an error.
   2 = SCAN_NOT_PROGRAMED – Scanner open for read but scanner failed configuring.

7.1.2 Read Scanner

The ReadSCAN() function is used to read all barcodes decoded since the last read. The scanner is powered and constantly able to scan barcode data once the OpenSCAN function has been executed. While the scanner will place itself in a low power mode when not being used, the scanner will not be powered off until the CloseSCAN function is issued. It is recommended that the scanner be configured to return a suffix at the end of a barcode so the application can delineate one barcode from another.

1. Files to be used

   SCAN.H
   CESCAN.DLL

2. Function

   DWORD ReadSCAN(
   DWORD dwRetry,
   unsigned char *RecBuf,
   DWORD *dwRecSize)

3. Parameter's

   dwRetry: The number of seconds that should be waited prior to returning a no data response.

   *RecBuf: A pointer to a buffer in which to put the barcode data.

   *dwRecSize The value must be set to the size of the read buffer when the read is executed and the size of the data read is returned when the read is complete. A value of 0 is returned indicating a timeout or no data is available.

4. Returned value

   0 = SCAN_OK – Open successful, scanner configured if needed.
   1 = SCAN_ERROR – The scanner was not opened due to an error.

7.1.3 Close Scanner

This function closes the scanner port and powers off the scanner. The SCANLOGDAT file is closed if logging was enabled.

1. Files to be used

   SCAN.H
   CESCAN.DLL

2. Function

   DWORD CloseSCAN();

3. Parameter's

   None

4. Returned value

   0 = SCAN_OK – Open successful, scanner configured if needed.
   1 = SCAN_ERROR – The scanner was not opened due to an error.

7.1.4 Read CESCAN.DLL Version

This function reads the version of the CESCAN.DLL and returns a string with the version information in Unicode format.

1. Files to be used

   SCAN.H
   CESCAN.DLL

2. Function

   DWORD VersionSCAN(TCHAR *Ver);

3. Parameter's

   *Ver = A pointer to a location to store the Unicode version string of the CESCAN.DLL.

4. Returned value

   0 = SCAN_OK – Open successful, scanner configured if needed.
   1 = SCAN_ERROR – The scanner was not opened due to an error.

7.2 Magnetic Swipe Reader (MSR) API's

CEMSR.DLL provides a simple Open, Read, Close type interface to the MSR. The CEMSR.DLL must first be loaded into memory before the Open, Read, Close functions can be called. A method of loading and unloading the CEMSR.DLL and can be found in the program MSRTEST.EXE. The source to the MSRTEST.EXE program can be found on the "iPAD Documentation & Utilities" CD in the "\Code_Samples\MSRTEST" directory.

7.2.1 Open MSR

The function OpenMSR() is used to open the iPAD's Integrated MSR

Note: The MSR is powered once the OpenMSR function is issued. While the MSR will enter a low power mode if not used, it will remain ready until a CloseMSR function is issued. To achieve the best battery life, close the scanner when not needed.

1. Files to be used

   MSR.H
   CEMSR.DLL

2. Function

   DWORD OpenMSR()

3. Parameter's

   None

4. Returned value

0=MSR_OKAY	MSR Opened Okay
1=MSR_OPEN_ERROR	Port Open Failed
2=MSR_BUFFER_ERROR	Couldn't allocate Internal Buffers
3=MSR_GET_TIMEOUTS_ERROR	Couldn't retrieve comm. port timeouts
4=MSR_SET_TIMEOUTS_ERROR	Couldn't set comm. port timeouts
5=MSR_GET_DCB_ERROR	Couldn't retrieve comm. port DCB structure
6=MSR_SET_DCB_ERROR	Couldn't set comm. port DCD structure

7.2.2 Read MSR

1. Files to be used

   MSR.H
   CEMSR.DLL

2. Function

   DWORD ReadMSR
   DWORD dwRetry,
   char *RecBuf,
   DWORD *dwRecSize)

3. Parameter's

   dwRetry: The number of attempted retries to query MSR prior to returning an error condition.

   *RecBuf: A pointer to a buffer in which to put the MSR data. The data returned is not filtered with the exception that the protocol and CRC characters are stripped.

   *dwRecSize Size of the data read is returned when the read is complete. A value of 0 is returned indicating a timeout or no data is available.

   The Orion's header is the first 16 bytes of data returned in *RecBuf. The actual card data follows the header. For a complete description of the prefix header, refer to the Orion Card Reader Specification.

Element	Byte	Location Description	Good Read Values
1	1-1	Positive Acknowledgement	"P"
2	2-3	Multi-track read	"6A"
3	4-5	Status	"00"
4	6	Track Type	Refer to Specification
5	7-8	Track 1 Read Status	"00"
6	9-10	Track 2 Read Status	"00"
7	11-13	Track 1 Length	Non-zero length
8	14-16	Track 2 Length	Non-zero length

4. Returned value

0=MSR_OKAY	MSR Read Okay
7=MSR_CTS_ERROR	MSR Clear To Send Error
8=MSR_WRITE_COMMAND_ERROR	Error writing data block to MSR
9=MSR_READ_DLEACK_ERROR	Error retrieving acknowledgement from MSR
10=MSR_WRITE_DLEENQ_ERROR	Error writing enquiry to MSR
11=MSR_READ_DATA_ERROR	Error reading MSR data (ReadFile)
12=MSR_DATA_ERROR	Error with MSR data integrity (no "P" acknowledgment).
13=MSR_RETRIES_ERROR	Exceeded Parameter 1 retry attempts
14=MSR_STEP_ERROR	Internal Step error (should not happen)
15=MSR_BCC_ERROR	MSR data failed CRC check

7.2.3 Close MSR

This function closes the MSR port and powers off the MSR. The MSRLOG.DAT file is closed if logging was enabled.

1. Files to be used

   MSR.H
   CEMSR.DLL

2. Function

   DWORD CloseMSR();

3. Parameter's

   None

4. Returned value

   0=MSR_OKAY MSR Close Okay

7.2.4 Read CEMSR.DLL Version

This function reads the version of the CEMSR.DLL. A string is returned in ASCII format.

1. Files to be used

   MSR.H
   CEMSR.DLL

2. Function

   DWORD VersionMSR(char * Ver);

3. Parameter's

   *Ver = A pointer to a location to store the ASCII version string of the CEMSR.DLL.

4. Returned value

   0=MSR_OKAY MSR Version Okay

7.3 Power Managment

Samples of code, which call these API's, can be found on the "iPAD Documentation & Utilities" CD in directory \Code_Samples\APITEST.

7.3.1 Read Battery Status

This function provides a method of reading the Main and Backup battery status. As you will notice this function returns a number of values that have no meaning to the iPAD. This format has been used to provide a consistent API with the TeamPad 7500/500. The main reason to use this feature is to determine the state of the Main and Backup batteries as well as to determine if the unit is on AC (Cradle).

1. Files to be used

   PowDll.h Header file for power control DLL
   PowDll.lib Library for Power control DLL

2. Function

```
BOOL TeamPadBatteryGetStatus2(
        PSYSTEM_POWER_STATUS_EX2 pSystemPowerStatusEx,
        UINT16 *pwCharge,
        DWORD *pdwErrDetail);
```

3. Parameter

   pSystemPowerStatusEx : Pointer to a structure in which to store the battery status information .

```
typedef struct _SYSTEM_POWER_STATUS_EX2
{
        BYTE    ACLineStatus;
        BYTE    BatteryFlag;
        BYTE    BatteryLifePercent;
        BYTE    Reserved1;
        DWORD BatteryLifeTime;
        DWORD BatteryFullLifeTime;
        BYTE    Reserved2;
        BYTE    BackupBatteryFlag;
        BYTE    BackupBatteryLifePercent;
        BYTE    Reserved3;
        DWORD BackupBatteryLifeTime;
        DWORD BackupBatteryFullLifeTime;
        DWORD BatteryVoltage;
        DWORD BatteryCurrent;
        DWORD BatteryAverageCurrent;
        DWORD BatteryAverageInterval;
```



```
        DWORD BatteryAverageCurrent
        DWORD BatterymAHourConsumed;
        DWORD BatteryTemperature;
        DWORD BackupBatteryVoltage;
        BYTE    BatteryChemistry;
    }

ACLineStatus; Status of the AC Line.
        0x00(AC_LINE_OFFLINE)           ; AC off line
        0x01(AC_LINE_ONLINE)            ; AC on line
        0xFF(AC_LINE_UNKNOWN)           ; Unknown


BatteryFlag;Status of the main battery
        0x01(BATTERY_FLAG_HIGH)         ; Good
        0x02(BATTERY_FLAG_LOW)          ;Low battery
        0x04(BATTERY_FLAG_CRITICAL)     ; Critical
        0xFF(BATTERY_FLAG_UNKNOWN)      ; Unknown Status

BatteryLifePercent ; Remaining capacitor of main batter.
        Returned value is 0 to 100 (%)
        0xFF(BATTERY_PERCENTAGE_UNKNOWN) ; Unknown

BatteryLifeTime ; Remaining time of main battery.
        It always returns 'BATTERY_LIFE_UNKNOWN'.
        0xFFFFFFFF(BATTERY_LIFE_UNKNOWN)        ; Unknown

BatteryFullLifeTime ; Operating time of fully charged main battery.
        It always returns 'BATTERY_LIFE_UNKNOWN'.
        0xFFFFFFFF(BATTERY_LIFE_UNKNOWN)        ; Unknown

BackupBatteryFlag ; Status of backup battery
        0x01(BATTERY_FLAG_HIGH)                 ;Good
        0x02(BATTERY_FLAG_LOW)                  ; Low battery
        0x04(BATTERY_FLAG_CRITICAL)             ; Critical
        0xFF(BATTERY_FLAG_UNKNOWN)              ;Unknown

BackupBatteryLifePercent ; Remaining life of backup battery
        It always returns '0'.

BackBatteryLifeTime ;Remaining operating time of backup battery
        It always returns 'BATTERY_LIFE_UNKNOWN'
        0xFFFFFFFF(BATTERY_LIFE_UNKNOWN)        ;Unknown

BackupBatteryFullLifeTime ;Operating time of full charged backup battery
        It always returns 'BATTERY_LIFE_UNKNOWN'.
        0xFFFFFFFF(BATTERY_LIFE_UNKNOWN)        ;Unknown Status
```


BatteryVoltage; Main Battery Voltage in millivolts.
Returned value is 0 to 65535 mV.

BatteryCurrent; Instantaneous current drain (mA).
Returned value is always 0

BatteryAverageCurrent; Short-term average of device current drain (mA).
Returned value is always 0.

BatteryAverageInterval; Time constant (mS) of integration used in reporting
Returned value is always 0.

BatterymAHourConsumed; Long-term cumulative average DISCHARGE (mAH).
Returned value is always 0.

BatteryTemperature; Reports Battery temperature in 0.1 degree C.
Returned value is always 0.

BackupBatteryVoltage; backup-battery voltage in millivolts.
Returned value is always 0.

BatteryChemistry;
Returned value is always 255.

pdwErrDetail: Buffer pointer to the return error code

Set to NULL if no error code is desired.

If the return value is FALSE then error codes are as follows.

101(POW_INVALID_PARAMETER)	;Parameter error
102(POW_SYSTEM_ERROR)	; System error (Lack of resource or others)

4. Returned value

TRUE : Success
FALSE: Failure

7.3.2 Force A Suspend

This API can be used to suspend the device under program control.

1. Files to be used

   PowDll.h Header file for power control DLL
   PowDll.lib Library for Power control DLL

2. Function

   BOOL TeamPadSetPowerDown (
   DWORD dwResume, Command
   DWORD *pdwErrDetail); Results

3. Parameter

   dwResume Must be a zero
   0 POW_SYSTEM_RESUME

   pdwErrDetail Buffer pointer to the return error code

   Set to NULL if no error code is desired.

   If the return value is FALSE then error codes are as follows.

101 (POW_INVALID_PARAMETER)	Parameter Error
102 (POW_SYSTEM_ERROR)	System Error (Short of System resources. etc)
103 (POW_SYSTEM_BUSY)	System Busy (Can not suspend now- retry operation)
104 (POW_KEY_DISABLE)	Power Key Disabled (Power Key must be enabled before calling this API.)

4. Returned Value

   TRUE: Success
   FALSE: Failure

7.3.3 Restart The iPAD

This function will reboot the iPAD. Assuming the function is successfully completed, then control will not be returned from this API as the system will have rebooted. Be sure to close all open files prior to issuing this API, otherwise data will be lost. This function can be used to reboot the system after a new application is downloaded from a host.

1. Files to be used

PowDll.h	Header file for power control DLL
PowDll.lib	Library for Power control DLL

2. Function

```
BOOL TeamPadRestartSystem (
        DWORD *pdwErrDetail);          Results
```

3. Parameter

pdwErrDetail Buffer pointer to the return error code

Set to NULL if no error code is desired.
If the return value is FALSE then error codes are as follows.

101 (POW_INVALID_PARAMETER) PARAMETER ERROR
102 (POW_SYSTEM_ERROR) SYSTEM ERROR

4. Returned Value

TRUE	Success
FALSE	Failure

7.3.4 LCD Brightness

This function provides a method of getting and setting the brightness of the iPAD 's LCD.

1. Files to be used

   PowDll.h Header file for power control DLL
   PowDll.lib Library for Power control DLL

2. Function

   BOOL TeamPadGetSetDispControl(
   DWORD dwGetSetCmd, Command
   DWORD *pdwValue, Value
   DWORD *pdwErrDetail); Results

3. Parameter

   dwGetSetCmd : Command which indicates get (read) or set (write) Brightness

   0x12(GET_BRIGHTNESS); Get status of Brightness value
   0x10(SET_BRIGHTNESS); Set Brightness value

   pdwValue :Buffer pointer to back light status or contrast value.

   When getting (reading) the value, the value is returned in this buffer.
   When setting (writing) the value, the value is written from this buffer.

   Followings are the values that can be used with the brightness control,
   0(BRIGHT_LO) ; Brightness LO
   1(BRIGHT_MID_LO); Brightness MID-LO
   2(BRIGHT_MID_HI); Brightness MID-HI
   3(BRIGHT_HI); Brightness HI

   pdwErrDetail: Buffer pointer to the return error code

   Set to NULL if no error code is desired.
   If the return value is FALSE then error codes are as follows.
   101(POW_INVALID_PARAMETER) ; Parameter error
   102(POW_SYSTEM_ERROR) ; System error(Lack of resource or others)

4. Returned value

   TRUE: Success
   FALSE: Failure

7.3.5 Device State

This function provides a method of getting and setting the iPAD's Green LED state and controlling the Bluetooth interface.

1. Files to be used
   PowDll.h Header file for power control DLL
   PowDll.lib Library for Power control DLL

2. Function
   DWORD TeamPadDeviceState(
   DWORD dCmd, Command
   DWORD *pdwErrDetail); Results

3. Parameter
   dwCmd : Command which indicates which operation to perform

   0x0100(POW_GREENLED_STATE); Get current state of Green LED
   0x0101(POW_GREENLED_OFF); Set Green LED off
   0x0102(POW_GREENLED_ON); Set Green LED on

   0x0200(POW_BLUETOOTH_STATE); Get current state of Bluetooth
   0x0201(POW_BLUETOOTH_OFF); Set Bluetooth off
   0x0202(POW_BLUETOOTH_ON); Set Bluetooth on
   0x0203(POW_BLUETOOTH_RESET); Reset Bluetooth

   pdwErrDetail: Buffer pointer to the return error code

   Set to NULL if no error code is desired.
   If the return value is FALSE then error codes are as follows.
   101(POW_INVALID_PARAMETER) ; Parameter error
   102(POW_SYSTEM_ERROR) ; System error(Lack of resource or others)
   -1(POW_DEVICE_STATE_ERR), Device State Error.

4. Returned value
   When getting (reading) the state, the value is dependent on which Query.
   0x0101(POW_GREENLED_OFF); Green LED is off
   0x0102(POW_GREENLED_ON); Green LED is on

   0x0201(POW_BLUETOOTH_OFF); Bluetooth is off
   0x0202(POW_BLUETOOTH_ON); Bluetooth is on

   When setting (writing) the state, the return value is
   TRUE: Success
   FALSE: Failure

8 Upgrading The Firmware

The iPAD firmware is upgradeable by the customer. A set of files can be placed in the \flashdisk folder via ActiveSync or the network and the program DUPFLAGEXE executed. The iPAD should be in the cradle to insure there is enough voltage to complete the firmware upgrade. The following files MUST be both present and in the structure defined as follows.

\flashdisk\DubStart.exe
\flashdisk\DUPFLAGEXE
\flashdisk\DUP\UPDATE.INF
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_000.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_001.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_002.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_003.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_004.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_005.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_006.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_007.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_008.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_009.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_010.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_011.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_012.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_013.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_014.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_015.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_016.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_017.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_018.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_019.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_020.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_021.bin
\flashdisk\DUP\os_files\F-VXXXXXXXXXX_022.bin
\flashdisk\DUP\os_files\manifest.dub

After placing the files listed above into the flash disk execute DUPFLAGEXE. At the completion if a software install had been conducted prior to the firmware update, the user will be prompted to recover the application.

9 Trouble Shooting The iPAD

9.1 iPAD will not power on

The two most common reasons the iPAD will not power on are either the "Main Battery" does not contain a sufficient charge to power on the unit or the "Battery Door Lock" is open.

First, confirm the "Battery Door Lock" is closed. Place the iPAD with the display facing down on a table with the battery door towards the bottom of the unit. The right battery door lock contains a switch that is used by the operating system to indicate the battery door is open. If this switch is open, the unit will not power on. If the switch is open, close the switch and try to power on the unit.

If the "Battery Door Lock" is closed, the "Main Battery" may be below operational voltage. Place the iPAD in a cradle and try to power on the unit. The LED on the top of the iPAD should turn orange when the iPAD is placed in the cradle, indicating the unit is charging from AC power. The LED should turn green as the unit starts to power on.

If the unit still does not power on, press the reset switch on the back of the unit just above the battery door. While the reset switch may result in some data being lost (open file handles), the RAM Object Store should remain intact. If the warm boot process works, the unit was hung.

If the unit will still not power on, open the battery door and remove the battery. Press the reset button labeled "Full Reset" to perform a cold boot. All data in the object store will be lost. Place the iPAD in a cradle and try to power on the unit. If the unit does power on, the unit was locked up.

9.2 TeamPad-Explore

The use of the TeamPad-Explorer to copy or load files onto the iPAD will bypass the recovery process established by CONFIGHHT. Files loaded by the TeamPad Explorer will probably not be recovered during a backup/restore or a recovery process.

9.3 ActiveSync Will Not Start

If the iPAD is powered on while in the cradle the message "USB Device Not Recognized" will be displayed on the desktop. Remove and replace the iPAD in the cradle to continue.

On rare occasions the desktop will have to be rebooted to get ActriveSync to connect.

Remove and replace the iPAD into the cradle. Make sure the iPAD is locked into the cradle.

The ActiveSync client does not automatically start in CE.NET. Make sure Repllog.exe has been started.

