BSQUARE® CORPORATION

Applications Designs Guidelines

for the Cassiopeia BE-300

Document No. Application Design Guide_1264

CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

1 INTRODUCTION

This document sets guidelines for the development of applications for the Casio BE-300.

The BE-300 uses a platform based on Windows CE 3.0 Embedded, so applications developers will need to be familiar with the basic techniques for developing applications for the Palm-size PC and the Pocket PC.

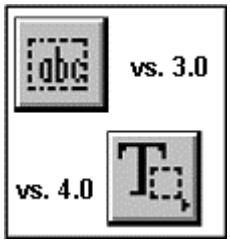
The BE-300's unique specifications differ from those of the Palm-size PC and the Pocket PC, and the programming implications of this fact are covered herein. This document provides the developer with the minimum general information required to develop applications for the BE-300.

Documentation Information

Copying Sample Code

Using this document requires the Adobe Acrobat® Reader™. The Reader opens with its Hand Tool enabled, allowing you to easily navigate through the document. To select and copy text such as sample code, you must enable the Text Select Tool. Figure one illustrates the Text Select Tool button as it appears on the toolbar in versions 3.0 and 4.0 of the Adobe Acrobat Reader.

Figure 1. Text Select Tool



IMPORTANT: Because the Reader copies each line of text with a hard return, it is important, when pasting from the Reader, to remove any hard return that separates a single line of code.

Conventions Used

The following typographic conventions are used in this guide.

Table 1. Conventions Used

Style	Convention
Italic font	References to sections or titles of documents.
[CAPS_ITALIC_ BRACKETED]	Generic file names.(Example [CPU_NAME] or [USER_NAME]
`Courier New font`	Font used to signify code text.
Bold face font	Font used for function names, menu items, and a command or action taking place in procedures.
CAPS	Used to signify keys (Example: ENTER, ESC, CTRL-SHIFT)

Additional Documentation

Help documents for developing applications for the BE-300 can be found by clicking on the Help option in eMbedded Visual Tools 3.0. Specifications for the supported Standard Windows CE APIs as well as custom BE-300 APIs will be listed here. Additional information may be found in the following documents:

- *The Cassiopeia BE-300 Development Environment Manual*
- *Developers Specification for Casio Be-300*
- *SDK Installation Guide*

2 OVERVIEW

The BE-300 uses an Embedded platform employing the Microsoft OS, Windows® CE 3.0.

The basics of applications development for the BE-300 are more or less the same as for the Palm-size PC and the Pocket PC. If you are already conversant with the basic techniques for developing applications for these devices, you will be able to develop applications for the BE-300 using similar methods. However, some aspects of the BE-300's unique specifications have implications for programming. This document provides information on these considerations, enabling the developer to create applications for the BE-300.

Unique Features of the BE-300

The BE-300 differs from the Palm-size PC and the Pocket PC in the following ways.

- The BE-300 uses an original shell known as Casio Original Shell, or **COShell** for short. DestTop is not displayed..
- The BE-300 uses an original memory configuration and memory control system.
- There are some differences in terms of the input/output devices installed (the BE-300 has no action control wheel, speaker or IrDA port).
- Original controls are provided for the BE-300.
- The BE-300 uses a subset of the Win32 API, plus the original Casio API (the Pocket PC's original API cannot be used).
- The BE-300 does not support the full run-time libraries of the Pocket PC and Palm-size PC.
- The BE-300 uses an original help system.
- The BE-300 uses an original installation system.

Required vs. Recommended Rules

The following sections cover the BE-300's unique specifications, and explain their implications for writing applications. Each consideration is classified as either "Required" or "Recommended".

Applications should always follow the rules given in “Required” items. Following the specified rules of “Recommended” items is not absolutely required, but will allow applications to make the best use of the BE-300’s special features.

3 SHELL AND PROCESSES

This section explains the programming implications of the BE-300's original shell and its functions. More detailed information can be found in the COShell API specifications included in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

Shell

The BE-300 uses an original custom shell, Casio Original Shell (or "COShell"). COShell offers the following UI functions:

- Hardware icons
- Taskbars
- Charge warning display
- Menu launching by pressing the Menu Button

COShell offers the following system functions:

- Application running control
- Memory shortage monitoring
- Charge monitoring

Application Categories and Running Control

Four application categories, labeled 0 through 3, are defined for the BE-300. Applications are placed in categories using settings in the registry. Each category has characteristics defining how the applications behave in certain situations. Control processing of various sorts is carried out when applications are launched and when memory shortages occur.

Table 2. Application Categories

Category	Description
0	Fully permanently-resident applications. Applications that have to be permanently resident, such as menus. (These applications have a high priority and stay resident unless a high priority, clear termination command is issued)
1	Permanently-resident applications Applications which require quick switching, such as mail, PIM applications (Calendar, Tasks, Notes, Contacts) and MP3 players (These applications stay resident if possible, but will terminate if there is insufficient memory.)
2	Ordinary applications Regular applications, such as browsers and extra applications supplied by independent software vendors. (These applications terminate in priority order when there is insufficient memory.)
3	Sub-applications Pop-up type, supplementary applications, such as calculators, clocks and settings menus. (These are terminated instantly by launching or switching between other applications.)

Launching Applications and Creating Processes

CoshExecute() [Required]

NOTE: Use **CreateProcess()** for launching Help(PegHelp.exe. See Chapter 11, *Online Help*, for more details

[When launching programs and creating processes, use the COShell API **CoshExecute().**

Do not use the Microsoft APIs, **CreateProcess()** or **ShellExecute()**. Running applications should use **CoshGetCommandLine()** to get the command line when receiving a COSH_EXECUTEAPP message (received when a call attempts to launch a second instance).

Using **CoshExecute()** will enable **COShell** to exert control over the launched program or process and other programs currently running.

If an application from category 0, 1 or 2 is launched using **CoshExecute()**, the shell uses **PostMessage()** to post a WM_HIBERNATE message to applications that are currently active. All programs receiving WM_HIBERNATE should save their data and free up all releasable memory.

Double Launch Prevention Processing

Applications for the Palm-size PC and Pocket PC are normally equipped with processing that prevents a program currently running from being launched again. Using **CoshExecute()** to launch applications on the BE-300 automatically prevents all category 0,1, and 3 applications from being double-launched. Category 2 applications can be launched multiple times unless defined in the registry as a Not Double Execute application.

Task Manager

Using the task manager supplied by the shell, programs currently being executed can be displayed and terminated. For display purposes, the name of the main window (the top-level window without an owner) is used. The task manager can be accessed by running the **Settings | Memory application**.

Memory Shortage Monitoring Control

At regular intervals, the shell checks the amount of memory left available for running programs. If the amount of memory remaining falls below the threshold value, the shortage of a memory is canceled by terminating the low application of a priority by the application of category 1 and 2 started now.

PostMessage() an application -- an end carries out WM_CLOSE.

(When a fixed time end is not carried out, it ends compulsorily by TerminateProcess().)

Handling Responseless Windows

When a window is generated that elicits no response from the OS on the termination of the application from the low-memory window, the shell does not display a window prompting the user to choose what to do about the application. In such cases, the situation is always handled as if the OS had sent a response to the effect that the application should be forcibly terminated.

4 MEMORY ARCHITECTURE & MANAGEMENT

This section explains the programming implications of the BE-300's original memory architecture and memory management.

System Memory Configuration

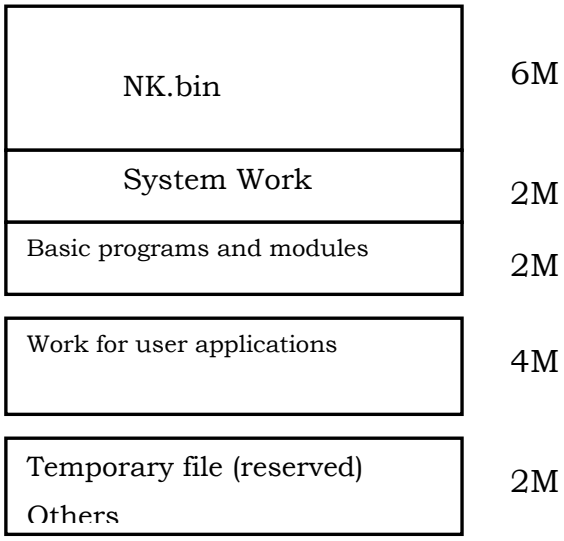
The BE-300 operates with a small amount of memory, so its memory configuration differs considerably from that of the Pocket PC. The system memory of BE-300 consists of RAM, Flash and ROM. The differences between the BE-300 and the Pocket PC are as follows:

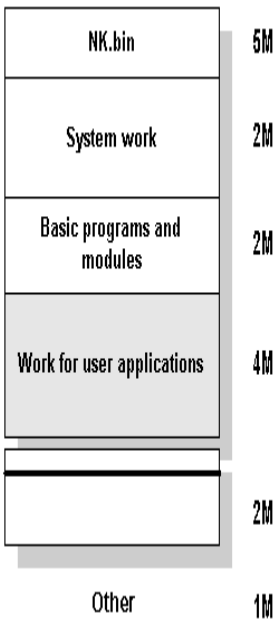
- On the BE-300, applications are installed on Flash.
- On the BE-300, the RAM is basically only used for work memory.
- On the BE-300, applications' user data has to be saved in the Flash area.

RAM Configuration

The RAM configuration is shown in the diagram below.

Figure 2. Ram Configuration





[Required] For standard applications installed in the BE-300, the size of the work memory for the program code, stack, heap etc., must be no greater than 4MB combined.

Work Memory and User Data

In the BE-300, the RAM area is not used as a storage area. Although the RAM can be used to store temporary files created while working, completed user files must ultimately be stored on the Flash disk.

The folders to be used to store user data can be acquired by means of functions supplied by the BE-300 System Disk API.

Also, the folders for temporary data can be acquired by means of **GetTempFolder()** supplied by the System Disk API. If the relevant folders do not exist when these functions are called, they are created automatically. More detailed information can be found in the System Disk API specifications found in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

Object Store

When the BE-300 is reset, all files are initialized. Files and databases created by applications in the user memory on RAM are all erased. In this respect, the BE-300 differs significantly from the Palm-size PC and Pocket PC.

This means that the object store (the disk system in the RAM) can only be used for temporary purposes. Also, when the BE-300 is reset, the object store is always initialized to erase temporary files.

[Required] Applications must use the Flash disk and the CF disk to store all data that has to be saved permanently. Use the folders acquired using the System Disk APIs such as **GetUserDiskName().** When terminated, each application must delete the temporary files it has created.

5 STORING FILES AND DATA

File System

On the BE-300, applications can save data to files in the RAM, but it should be borne in mind that data on RAM is treated as temporary. The data in the RAM disappears completely when the battery runs out, so applications need to be designed so that their files and data are stored on CF or Flash. More detailed information can be found in the System Disk API specifications included in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

[Required] Applications must be designed so that data written to RAM is ultimately stored on CF or Flash.

An application can obtain a user disk folder on CF or Flash using the System Disk API **GetUserDiskName()**

[Required] Applications must be designed so that their user data is stored to the memory folder destination acquired by the **GetUserDiskName()** function.

Folder structure

The folder structure is shown in the diagram below.

Figure 3. Folder Structure

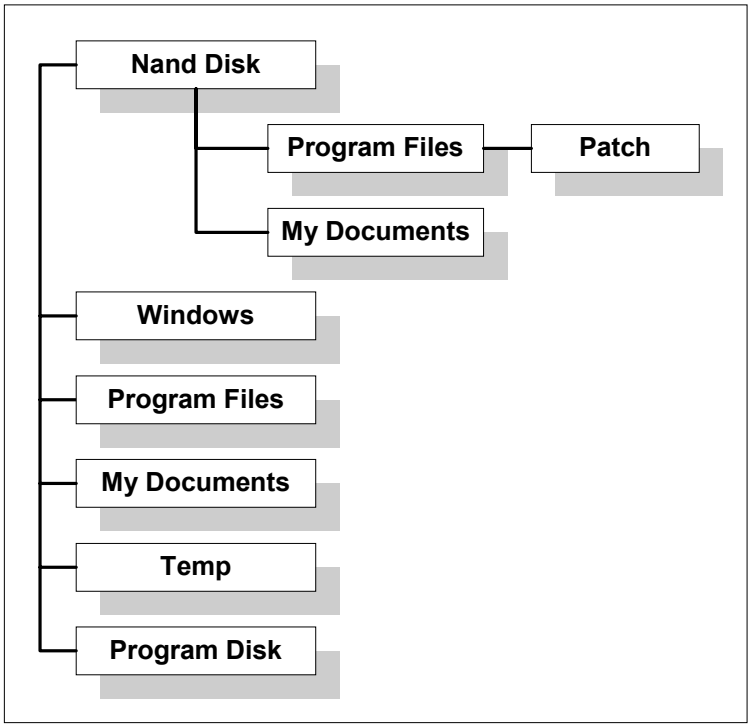


Table 3 provides descriptions for each of the folders.

Table 3. Folder Structure Descriptions

Folder	Description
Nand Disk\Program Files	For application files (Flash, compressed)
Nand Disk\Program Files\Patch	Patch files for NK.bin (include add-in drivers)
Nand Disk\My Documents	For user data (Flash)
Windows	Files in nk.bin(Flash) are seen as uncompressed files (Driver, Library, Menu.exe, etc.)
Program Files	not used (RAM)
My Documents	not used (RAM)
Temp	work folder for applications(RAM)
Program Disk	Applications(RAM) are seen as uncompressed files (Read Only)

- Application files are stored in \Nand Disk\Program Files as compressed files after installation. They can be accessed as uncompressed files via \Program Disk when reading.
- To add a file that will persist in the Windows directory after device reset, put it in \Nand Disk\Program Files\Patch

- Get the folder name for user data (e.g.\Nand Disk\My Documents, \Storage Card\My Documents) using **GetUserDiskName().**
- Get the folder name for an inserted CF card (\Storage Card) using **FindFirstFlashCard().**
- Get the work folder name for applications using **GetTempFolder().**
- When launching an application, the system searches for the application in the following order:

  Windows > *[ROOT]* > Program Disk
- When calling **LoadLibrary,** the system searches for the library in the following order:

  The folder where EXE resides > Windows > *[ROOT]* > Program Disk
- Specifying the full path is therefore not necessary when launching an application or using LoadLibrary.

Registry, Database(Alarm)

When registry operations are carried out on the BE-300, the registry is saved in the RAM therefore making data written to the registry temporary. The registry must be saved to flash in order for registry modifications to be permanent.

This is also true for the alarm database.

[Required] To save the registry permanently, **CGDFFlashRegistry()** must be called. To save the database(alarm) permanently, **CGDFlushAlarm()** must be called.

IMPORTANT: The above functions should be used efficiently, as frequent use will affect system performance. If a series of registry or alarm modifications are made, call the functions after all changes are complete.

Other Fixed Memory Areas

For information on the Flash card and other fixed memory areas, see "Specification for Developers".

6 POWER MANAGEMENT

On the BE-300, power management processing is carried out when the power is switched on, and when the battery runs low. For details, see "Specification for Developers". More information on power management can be found in the Power Management API specification included in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

7 GRAPHICS AND MULTIMEDIA

Although it has a buzzer, the BE-300 does not have a speaker and therefore cannot directly output sound files (WAV and other formats). However, sound files can be played back using earphones or speakers plugged into the device.

[Recommended] To play and stop buzzer files or wavefiles, use the following APIs:

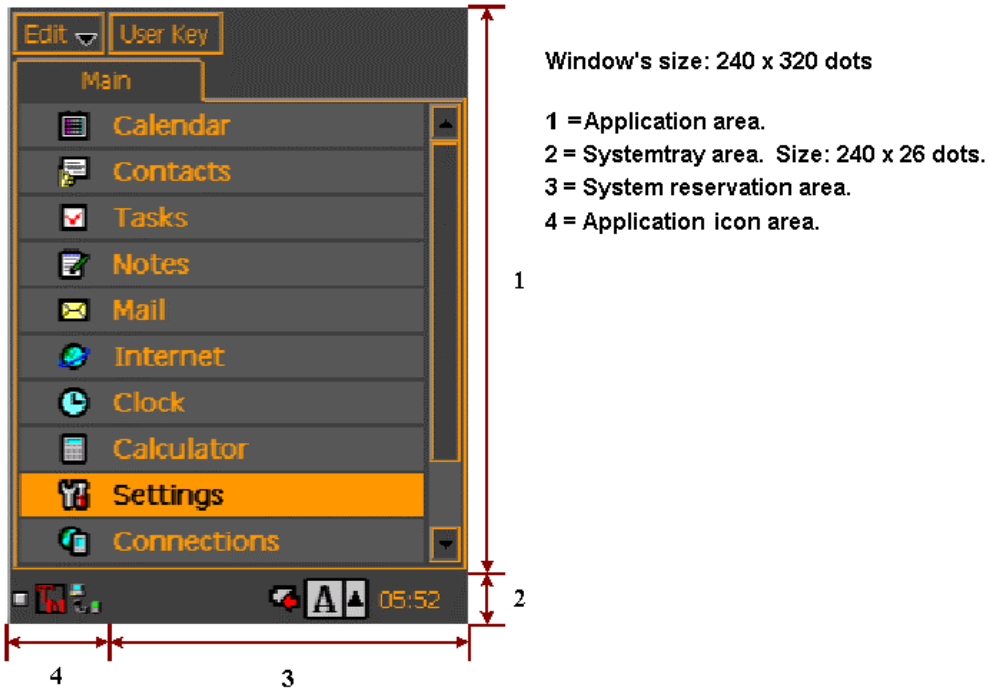
- **PlayBuzz()**
- **StopBuzz()**

8 User Interface Design

Window

The following diagram illustrates BE-300's window.

Figure 4. The BE-300 Window



Command Bars and Menus (Original Controls)

The following original controls are provided for creating command bars and menus:

- **CSOButton:** A functionally-expanded version of the "Button" control in the standard API
- **CSOBar:** A functionally-expanded version of the "CommandBar" control in the standard API

[Recommended] Instead of using the standard command bar control, use **CSOBar**. Create menus by using **CSOBar** in conjunction with **CSOButton**.

NOTE: Pocket PC Shell functions are not available on the Casio BE-300.

Standard Controls

The Windows CE 2.X Common Controls can be used on the BE-300. However, certain controls are subject to the following restrictions.

- MessageBox and MessageBeep (Windows CE standard controls) cannot be used to output sounds.
- Do not use the functions PageSetupDlg() and PrintDlg(), as the BE-300 has no printer system.

[Recommended] The Casio BE-300 original API, MessageBuzzer(), should be used in conjunction with the MessageBox() API in order play the appropriate buzzer sound when a message box is displayed.

Moreover, the following guidelines should be noted with regard to list views.

[Recommended] Please use the following settings for list views:

- Specify LVS_SINGLESEL during **CreateWindow()** to limit the number of selections in the list view to 1. If multiple selections are required, execute **ListView_SetExtendedListViewStyle()** with LVS_EX_CHECKBOXES.
- Allow all items in a row to be highlighted when selected by adding the LVS_REPORT style when calling **CreateWindow()** to create the list view. In addition you should call **ListView_SetExtendedListViewStyle()** with the LVS_EXFULLROWSELECT parameter.

Termination Buttons

[Recommended] Please provide each application with a termination button. The code that launches the menu **(CoshMainMenuExec())** should be added to the processing to be carried out when the termination button is pressed.

Colors

[Recommended] On the BE-300, designs can be changed (i.e. the colors of controls can be changed) by Settings. Applications should be designed using standard colors in order to support this. For example, you can set the **FaceColor** field of a **CSOButton.btnExStyle** to CLR_INVALID to use the default color system. Doing this allows the color of the button face to change whenever the user decided to change the color scheme on the device.

9 APIs

NOTE: The Pocket PC original APIs cannot be used.

The BE-300 comes with Windows APIs (a subset of the WIN32 API), plus Casio's original APIs. For further details on the purpose of these APIs, see *Specification for Developers*. Specifications for each original Casio API can be found in eMbedded Visual Tools Help after the CASIO BE-300 SDK is installed.

Windows APIs

The standard APIs included in eMbedded Visual Tools 3.0 are supported. Multimedia APIs are also supported.

For details, see Appendix A: Typical APIs Supported.

Libraries

The BE-300 contains the MFC libraries but do not contain any of the VB runtimes. The Cassiopeia's proprietary libraries have also been included in the BE-300. For further details, see *Specification for Developers*.

10 MISCELLANEOUS FEATURES

Dialup

For details, see Com Composer API specifications included in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

Module Monitor

In the BE-300, the module monitor function can be used. For details, see the Module Monitor API specifications included in the eMbedded Visual Tools help files after the CASIO BE-300 SDK has been installed.

Message

The Casio shell includes the two original messages. These are the messages COSH_EXECUTEAPP and COSH_NOTIFYICON_RESULT. COSH_EXECUTEAPP, which is defined as WM_APP(0x8000)+0x3000, is sent to an active application when it is specified in a call to **CoshExecute()**. COSH_NOTIFYICON_RESULT, which is defined as WM_APP(0x8000)+0x3001, is sent to a window after the function, **Shell_NotifyIcon()** is called.

Other messages that are used by applications in common are defined as WM_APP + 0x2f00--WM_APP + 0x31ff (include WM_APP + 0x3000--WM_APP + 0x30ff (shell)). Do not assign custom messages in an application.

WM_USER is defined as 0x0400. If the application uses an original message that is assigned a large number, make sure message values do not coincide.

11 ONLINE HELP

The BE-300 uses an original online help system displaying text and bitmapped graphics. The following tags are available.

Table 4. Online Help

Tags	Description
Basic tags	
<html></html>	
<head></head>	
<body></body>	Scrollable body text area
<title></title>	String displayed following on from "Help" in the caption
Tags in body text	
<img src=”***.gif”> 	Explanatory image (GIF) displayed (always displayed at top, ignoring options other than "src".
 	Line break (prevents text spilling over on to graphics(<br CLEAR="ALL">))
	Column list control (alternative strategy used when items cannot be aligned using spaces)

[Required] The help files you create must be compatible with this original help system.

HELP File

Put the HELP file in the same folder in which the application .exe file resides. Use **CreateProcess()** to run PegHelp.exe. Before calling **CreateProcess()**, get the folder path of app EXE file, and call **CreateProcess(),** specifying the name of the HTML file as the absolute path.

IMPORTANT: Make sure the path is specified within quotes.

Sample code is shown below.

Sample code

NOTE: You can also create a HELP subfolder beneath the folder in which the application .exe file resides. When placing the HELP file in a subfolder, add the path before the file name.

```
//HELP message processing  "calen_alarm.html" is help file
//name.

case WM_HELP:
{
    TCHAR fileName[MAX_PATH + 3] = TEXT("\"");
    DWORD fineNameSize;
    PROCESS_INFORMATION pi;
    memset( &pi, 0, sizeof(PROCESS_INFORMATION) );
    fineNameSize = GetModuleFileName( NULL, filename+1,
MAX_PATH );
    if ( fineNameSize ){
        FilenameCutFileName( filename+1);
    }
    lstrcat( fileName, TEXT("calen_alarm.html") );
    lstrcat( fileName, TEXT("\"") );
    CreateProcess( TEXT("PegHelp.exe"), fileName,
                   NULL, NULL, FALSE, 0, NULL, NULL, NULL,
&pi );
}
break;

//module for HELP
void FilenameCutFileName( TCHAR *fileName )
{

TCHAR *ptr = fileName + _tcslen(fileName);
while(1){
    ptr--;
    if ((*ptr == TEXT('\\')) || (*ptr == TEXT('/'))) {
        *(++ptr) = TEXT('\0');
        break;
    }
    else if ((*ptr == TEXT('\0')) || (ptr == fileName)) {
        break;
    }
}

return;
}
```

12 PORTING APPLICATIONS

When applications created for the Palm size PC or the Pocket PC are ported to the BE-300, the following should be taken into consideration.

- **Launching applications:** The BE-300's special launch function **CoshExecute()** should be used so that the Casio original shell can control the application.

  (See: *Launching Applications and Creating Processes* in Chapter Three.)

- **WM_HIBERNATE:** When a WM_HIBERNATE message is received, the application needs to save work data and free up releasable memory and resources.

  (See: *Memory Shortage Monitoring Control*, in Chapter Three.)

- **The user data storage location:** In the BE-300, the RAM area is not used as a storage area. Save the user data in a specified folder on Flash or CF.

  (See: *Work Memory and User Data*, in Chapter Four, and *File System* in Chapter Five.)

- **Registry, Database(Alarm):** On the BE-300, the Registry and Database(Alarm) are temporary. To save them permanently, specific functions must be called to store them on Flash.

  (See: *Registry, Database(Alarm)* in Chapter Five)

- **Using Casio's original controls in place of standard controls:** It is recommended that the **CommandBar** and **Button** controls should be replaced with the original controls supplied by Casio. (Shell functions of Pocket PC are not available.)

  (See: *Command Bars and Menus* in Chapter Eight)

- **Restrictions on the Help function:** Although conventional help files can be displayed on the BE-300, the Help function is limited.

  (See: *Online Help* in Chapter 11.)

- **Main unit supports buzzer only:** Although sound can be played back through the earphones using the standard API, the buzzer is the only sound that can be output by the main unit itself. To sound the buzzer, use the original APIs provided.

  (See: *Sound* in Chapter Seven.)

- **Color scheme:** On the BE-300, window colors can be specified as in Windows. BSQUARE recommends that you use standard colors in the Windows style.

  (See: *Colors* in Chapter Eight.)

- **Installer:** The general Windows application installer system cannot be used. The custom BE-300 installation system should be used instead.

  (See: Chapter 13, *Installing Applications*.)

13 INSTALLING APPLICATIONS

The BE-300 uses an original installation system. Generally speaking, the Windows application installer system cannot be used without modification. The tools provided by Casio should be used instead. The tools can be used to create a Setup program to run on a desktop PC which will install an application to a connected BE-300 device. Files and registry settings can be appropriately set.

[Required] Use the BE-300 installation system for installing applications.

Appendix A

TYPICAL APIS SUPPORTED

Please note that the list of APIs supported is subject to change. The following list is correct as of May 22, 2001.

- WinAPIs: Automation, OLE, Message Queuing, Storage, NTLMSSP, GDI, Time, Clipboard, DLLs, Errors, File IO, Process and Thread, Memory Manager, Registry, Exception (Command Processor shell not supported).
- CE APIs: File Mapping, IME, Synchronization, Notify, Database, RAPI (Handwriting Recognition, Input Panel and Project Control APIs are different from the Pocket PC's. ActiveSync, CEUTIL, Pocket Outlook,Voice recorder, Inkx Control, Rich Ink Control, Game API, Today screen API, Transport API, PPC User Interface API and PPC UserInterfaceMessage are not supported.)
- Network and communications: Windows network, comm., Winsock, Win User I/F Service, TAPI (Remote access, IP helper, SNMP, IPC Internet, HTML control and Mail API are not supported.)
- Shell and Common Controls: Progress Bar, Rebar, Status Bar, Tool Bar, Track Bar, Command Bar (Tool Help is not supported.)
- Multimedia and DirectX: Multimedia APIs are different from the PC's. DirectX (except for DirectDraw) and GameX are not supported.
- Security: Crypt API (Smart Card and SSPI are not supported.)
- Drivers: Device IO, NDIS drivers, File System driver