BSQUARE® CORPORATION

Developers Specification
for Casio BE-300

Document No. Developers Specification 1264

CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

1

INTRODUCTION

Objectives

This document summarizes the specifications for the design of the Cassiopeia BE-300 device. The impact the unique design of the device has on writing applications will also be discussed. Methods to be used when writing applications will be introduced taking into the consideration the device design specifications.

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

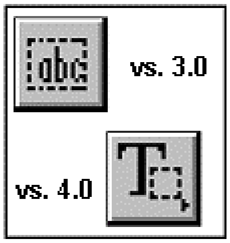
Help documents for developing applications for the BE-300 can be found by clicking on the Help option in eMbedded Visual Tools 3.0. Specifications for the supported Standard Windows CE APIs as well as custom BE-300 APIs will be listed here. Additional information may be found in the following documents:

- *Applications Designs Guidelines for the Cassiopeia BE-300*
- *The Cassiopeia BE-300 Development Environment Manual*
- *SDK Installation Guide*

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.

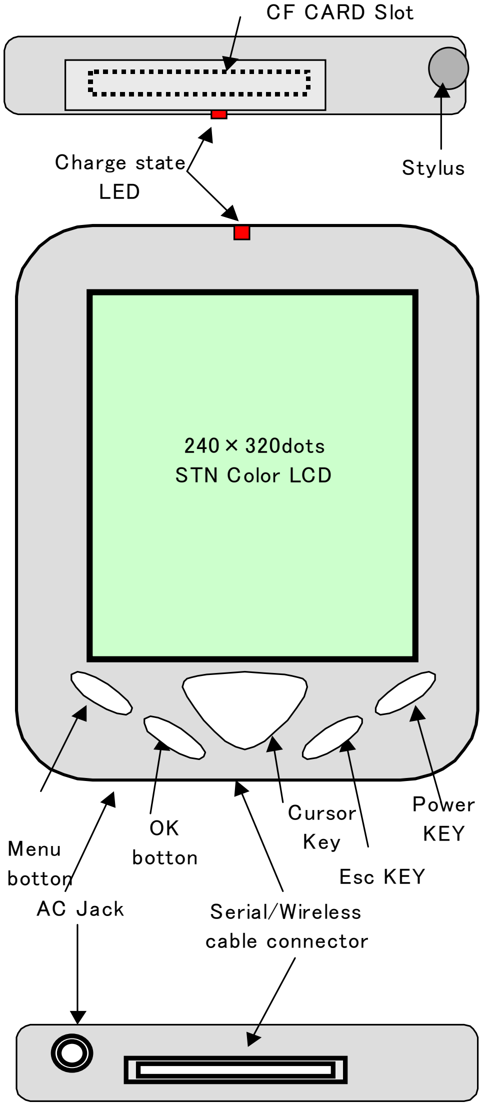
2

BE-300 SYSTEM

Hardware

The following illustration shows the hardware components of a Cassiopeia BE-300.

Figure 2. BE-300 Hardware Components



Touch screen

The *touch screen* is an LCD covered by a *resistive touch panel*. The LCD has a portrait orientation with a 240 x 320 pixel resolution.

Keyboard

Hard keys are located on the front panel

Buzzer

Buzzer alarms may be played out by the device buzzer. Applications can specify buzzer files to play when necessary.

Audio

Sound files can be played and heard through headphones or speakers plugged into the Headphone jack.

Memory

BE-300 has a least 2 MB built-in Flash memory for storage memory and 6 MB RAM for work memory. External memory may be utilized via the CF card slot.

Communication

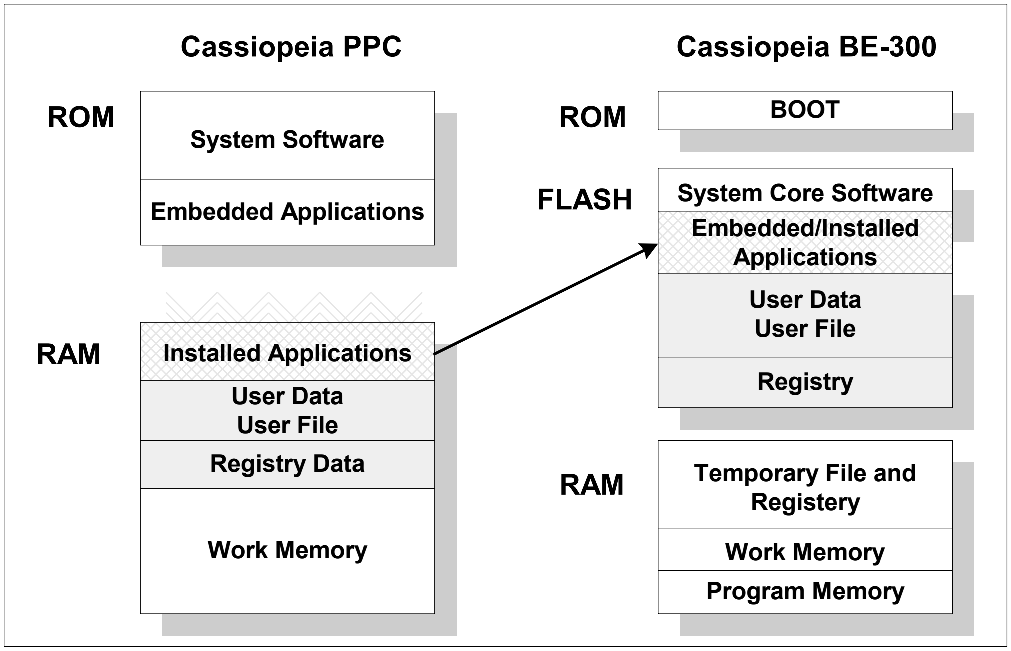
BE-300 can be connected to a PC using a serial, USB or LAN cable. The device can also be connected to a network wirelessly using a CF wireless card.

System memory configuration

Cassiopeia BE-300 has a very different memory configuration from that of Pocket PCs in order to allow it to run on a small amount of memory. (Figure 2-1) BE-300 differs from Pocket PCs in the following respects:

- Applications are installed in Flash.
- The RAM is basically used for working memory only.
- The user data for the application must be stored in the Flash area.

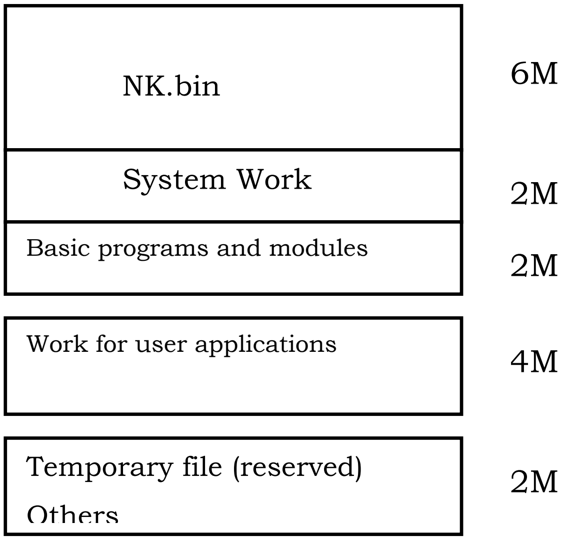
Figure 3. Memory Configurations for Pocket PC vs. BE-300



RAM configuration

In the case of standard applications installed in the BE-300, the total size of the working memory represented by program code, stacks, heaps and so on must be no more than 4MB.

Figure 4. RAM Configuration


NK.bin
6M
System Work
2M
Basic programs and modules
2M
Work for user applications
4M
Temporary file (reserved)
Others
2M

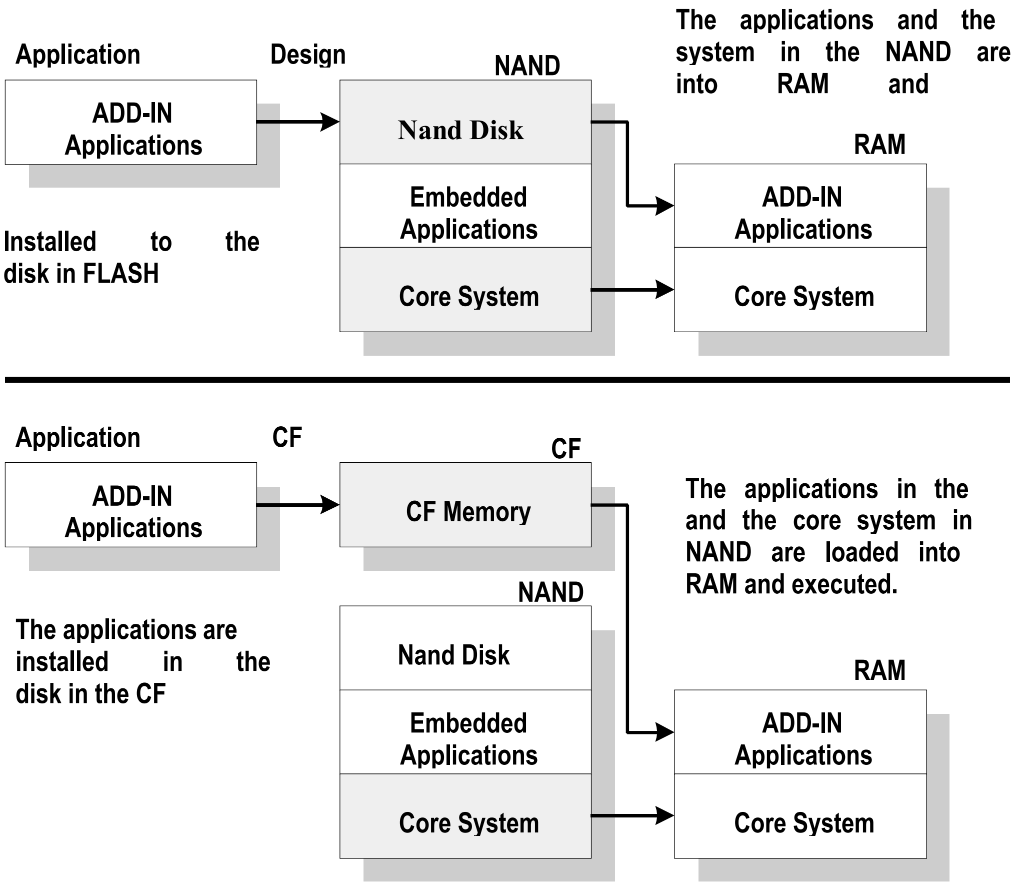
3

PROGRAMMING FOR THE BE-300

Core System

The Cassiopeia BE-300 is designed to allow the development of many, varied applications. To achieve this, a number of libraries must be provided from the system. The size of the memory makes it difficult to pre-embed all of the libraries in the system. Cassiopeia BE-300 is configured as illustrated in Figure 5 to allow various different applications to be implemented.

Figure 5. Installation



Memory Management

Memory Usage

The BE-300 does not allow the RAM area to be used as a storage area. The RAM may be used to store temporary files created during a task, but completed user files must be stored on the Flash disk. The folders in which user data is stored can be obtained by using the functions described below. The relevant folder is created automatically if it does not exist when these functions are called.

Table 2. User Memory Folders

API	Use	Notes
GetTempFolder	Working memory	Max. around 4 to 5 M, global memory, stack memory, heap memory etc.
GetUserDiskName	User storage memory	User data should normally be stored in this memory folder. Returns the user storage folder on the media where the application is stored.
GetInnerUserDisk	User storage memory	Returns the user storage folder on the internal Flash disk.
MakeUserDiskName/ FindFirstFlashCard	User storage memory	Returns the user storage folder on the CF disk specified by FindFirstFlashCard. Dependent on the CF memory size.

Low-Memory Management

If an application on a BE-300 needs more memory, the shell shuts down applications without notifying the user.

Memory Boundary

BE-300 shell fixes the memory boundary that separates RAM area into program memory and user memory. All applications share less than 2M byte of user memory in RAM.

The order in which applications are terminated to free up memory is determined using the following rules.

- Category 2 applications are terminated before category 1 applications
- Category 0 applications are not terminated
- Applications of the same category are terminated sequentially from the bottom of the Z-order
- Category 1 or 2 applications with the top window in the Z-order is terminated last
- Since applications are closed until the shortage of a memory is solved, two or more applications may be terminated simultaneously.
- Message WM_CLOSE is posted to each application's window of style WM_VISIBLE the owner window. The application which received WM_CLOSE should perform end processing correctly. (Note: Windows that are not WM_VISIBLE are not terminated automatically)

Example 1

Top of Z-order	Termination order
------------ A application of category 1	(6)
------------ B application of category2	(3)
------------ C application of category1	(5)
------------ D application of category2	(2)
------------ E application of category0	(no end)
------------ F application of category1	(4)
------------ B application of category2	(1)
Bottom of Z-order	

Example 2

Top of Z-order	Termination order
------------ B application of category2	(6)
------------ E application of category0	(no end)
------------ A application of category 1	(5)
------------ C application of category1	(4)
------------ D application of category2	(2)
------------ F application of category1	(3)
------------ B application of category2	(1)
Bottom of Z-order	

Example 3

Top of Z-order	Termination order
------------ x application of category3	(Closed after start of another application)
------------ E application of category0	(no end)
------------ G application of category2	(6)
------------ A application of category1	(5)
------------ C application of category1	(4)
------------ D application of category2	(2)
------------ F application of category1	(3)
------------ B application of category2	(1)
Bottom of Z-order	

The category classifications for applications become effective by registering the applications with a registry. Applications not placed into a category in the registry are classified as category 2.

- **Category 0:** An application registered into this category is not automatically ended by the shell. It is necessary to register an application that must reside permanently into this category.

  Since it resides permanently, be careful about consumption of memory.

  (The registration to the registry which manages category 0 application is required.)

- **Category 1:** An application registered into this category resides as long as possible. However, if memory is insufficient, and no category 2 applications are available for termination, category 1 applications will be terminated sequentially from the bottom of Z-order.

  (The registration to the registry which manages category 1 application is required.)

- **Category 2:** An application registered into this category is an application that is ended preferentially when a shortage of memory occurs. However, when a window from a category 2 application is at the top of the Z-order, it is the last to be closed to free up memory.

  (A registry does not need to be registered.)

- **Category3:** An application registered into this category is automatically terminated when another application is started. Therefore, category 3 applications can only be at the top of the Z-order.

  (The registration to the registry which manages category 3 application is required.)

Obtaining Program and User Folders

The functions listed in Table 3 can be used to obtain particular folders that are used for storing user data, programs, and other memory needs. The folder obtained from some of the functions depend on whether the application was started from Flash or CF. The relevant folder is created automatically if it does not exist when these functions are called.

Table 3. APIs to Retrieve Memory Folders

API	Started from NAND	Started from CF
GetUserDiskName	\Nand Disk\My Documents	\Storage Card\My Documents
GetProgramDiskName	\Program Disk	\Storage Card \Program Files
GetInnnerProgramDisk	\Program Disk	\Program Disk
MakeProgramDiskName /FindFirstFlashCard	\Storage Card\Program Files	\Storage Card \Program Files

FindFirstFlashCard	\Storage Card	\Storage Card

Application Disk

Design

On the BE-300, all applications are installed to the NAND Flash or the CF. These applications are stored in special folders in a compressed form. A virtual disk system is supported for viewing extracted forms of the applications. Figure 6 demonstrates how the virtual disk system is utilized.

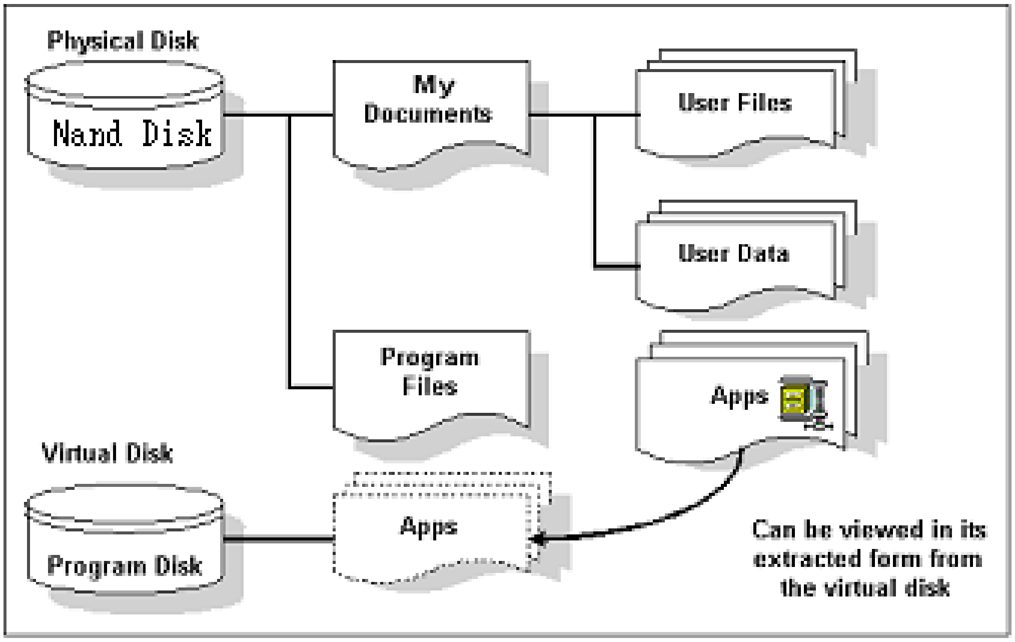
The applications appear as follows on the different types of disk.

- Compressed format on actual disk \Nand Disk\Program Files\Application1.exe.cpk
- Uncompressed format on virtual disk \Program Disk\Application1.exe

Programming Implications

When accessing the program disks on Flash or CF card, use the **GetProgramDiskName()** function rather than accessing the folders directly. The destination folder obtained depends on whether the application was started from Flash or CF card.

Figure 6. Program Files and Virtual Program Disks



User Files

Design

On the Cassiopeia BE-300, it is possible for applications to store files and data in the RAM, but the data in the RAM is treated as temporary. Data in the RAM is completely deleted if the battery runs out.

Programming Implications

Applications should be designed to store their files and data in the CF or the Flash. The destination folders in which user data should be stored depends on the location from which the core is started. To store user data in the Flash, the destination memory folder is determined by **GetSystemDisk().** To store user data to CF, the destination memory folder is determined by **MakeUserDiskName().** The BE-300 supports the standard common dialogues applications may use for end user file manipulation.

Registry and Alarms

Design

If the registry is manipulated on the Cassiopeia BE-300, these registry files are also stored as temporary files in the RAM. If the registry is not saved to the Flash after any registry modifications, the registry will return to its previous state after the device is reset. Similarly, any modifications to the alarms database are stored as temporary files in the RAM.

Programming Implications

The registry must be written back to the Flash after any modification using the **CGDFlushRegistry()** function.

If the alarm settings are changed, they must be written back to the Flash using the **CGDFlushAlarm()** function.

Also, to change the system password, use **CGDSetPassword()** instead of **SetPassword()**. To either enable or disable password query upon startup of the device, use **CGDSetPasswordActive()**.

Flash Card

Design

The BE-300 device is built with a Compact Flash slot. When a memory card is inserted, the device detects it and creates a folder representing the external disk.

Interface

To determine if the device is equipped with an inserted Flash card, call both the **FindFirstFlashCard** and the **FindNextFlashCard** functions. **FindFirstFlashCard** returns a search handle that **FindNextFlashCard** uses. It also returns a pointer to the first Flash card, if any. **FindNextFlashCard** returns a pointer to the next Flash card and a BOOL value indicating if the find was successful. These functions retrieve find data for the Flash card, which includes information such as filename and various attributes.

Using Auto-Run

The Auto-Run feature enables BE-300 software to autorun an application when a Compact Flash card is inserted into the BE-300 CF slot. When the card is detected, the BE-300 automatically loads a specified application from the Compact Flash card into active memory and runs it. Rename the application to be autorun upon CF card insertion to “autorun.exe” and copy the executable to a particular directory on the card as follows.

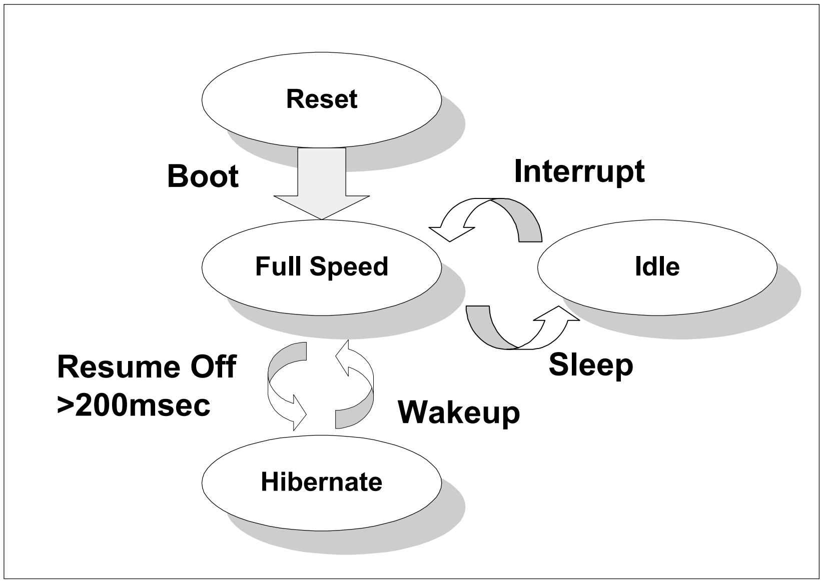
<Storage Card Root> \Ce\R4100\autorun.exe

Power Management State

Power State

The Cassiopeia BE-300 stays at a particular power state based on user activities and programming activities. All applications can enhance power management by using functions that let the operating system block a return to the application. To accomplish this, use the **GetMessage** function, rather than the **PeekMessage** function, because **GetMessage** lets the system block return to the application, and **PeekMessage** does not. Letting the operating system retain control allows it to determine when the system is inactive so that it can select the most efficient operating state for the BE-300.

Figure 7. Power State



- FULL Speed: System or application is running.
- IDLE: System or application is waiting an event. Device driver is working
- Hibernate: Device is off. Device driver are stopped.

Wakeup from the Hibernate state

Applications cannot run in the suspend state. However, some functions can wake up a BE-300 device from the Hibernate state. **CeRunAppAtTime** and **WaitCommEvent** are such functions.

Message for wakeup

Applications do not receive any information while the BE-300 device is turned off. The BE-300 does, however, send a message, WM_POWERBROADCAST, at wakeup. Applications can determine that the device has been turned off by receiving this message.

Device Reset

Design

When the BE-300 is reset, all files in the object store are initialized. This is a major difference between the Cassiopeia BE-300 and the Cassiopeia PPC. All files and databases generated by the application in the user memory in RAM are erased.

Since the data in RAM is temporary, a cold boot is performed in order to ensure that no lock or other such mechanism is activated due to corruption in the system.

Programming Implications

User data must be saved to Flash or CF in a folder obtained by **GetUserDiskName()**. Registry and alarm data must also be saved to Flash after modification using the **CGDFlushRegistry()** and **CGDFlushAlarm()** functions, respectively.

Protect Mode

In Cassiopeia Embedded, applications run in “Full Kernel Mode”.

Thread Scheduler

The system scheduler runs at a resolution of 1 ms. The default for Quantum is set to 100ms.

4

PROGRAMMING INPUT AND OUTPUT

Key Input

Describes the specification for key events and key navigation

Table 4. KeyCodes

Key Name	VKey(down)	VKey(up)
Right arrow	VK_RIGHT down	VK_RIGHT up
Left arrow	VK_LEFT down	VK_LEFT up
Up arrow	VK_UP down	VK_UP up
Down arrow	VK_DOWN down	VK_DOWN up
Top Menu	VK_LWIN down , VK_F12 down	VK_LWIN up, VK_F12 up
OK	VK_F23 down	VK_F23 up, VK_RETURN down, up
Escape	VK_F24 down	VK_F24 up, VK_ESCAPE down, up
Power OFF	VK_OFF	

Table 5. Navigation

Navigation	Effect
Power + Up arrow	Increase contrast
Power + Down arrow	Decrease contrast
Power + Right arrow	Increase bright
Power + Left arrow	Decrease bright
Power + OK	Initiate calibration

Hard Icon

Hard icons are activated in the same way as key events.

Table 6. Hard Icons

Number	Key event
ICON1	VK_LWIN, VK_F1
ICON2	VK_LWIN, VK_F2
ICON3	VK_LWIN, VK_F3
ICON4	VK_LWIN, VK_F4
ICON5	VK_LWIN, VK_F5
ICON6	VK_LWIN, VK_F6
ICON7	VK_LWIN, VK_F7

Buzzer

Design

The buzzer is one of the output devices on the Cassiopeia BE-300. There is no speaker built in the device.

Programming Implications

Applications can play and stop buzzer files for notifications or alarms as necessary using the **PlayBuzz()** and **StopBuzz()** functions. .buz files must be created for play and the buzzer file type specified in the function calls.

Sound

Design

Headphones or speakers can be plugged into the headphone jack at the bottom of the device for playback of wavefiles.

Programming Implications

Applications can play and stop wavefiles through headphones or speakers using the **PlayBuzz()** and **StopBuzz()** functions. .wav files must be created for play and the wavefile type specified in the function calls.

I/O device

The following peripheral circuits are provided, but their interfaces can be accessed using the standard CE APIs.

Table 7. Peripheral Circuits

IO Device	Accessible APIs
Serial	COM, TAPI, RAS, Socket, ActiveSync
USB	COM, TAPI, RAS, Socket ActiveSync
PC/CF Modem card	COM, TAPI, Socket,
CF memory card	File I/O
CF LAN card	Socket

Battery Driver

You can get power information by calling **GetSystemPowerStatusEx**. BE-300 contains the following battery information.

Information	Description or status
ACLineStatus	Offline/Online
BatteryFlag	Hign/Low/Critical/Charging
BatteryLifePercent	10,25,50,75,100%
BatteryLifeTime	Not supported
BatteryFullLifeTime	Not supported
BackupBatteryFlag	Not supported
BackupBatteryLifePercent	Not supported

5

SHELL & USER INTERFACE

Windows

Using Full-Page Child Windows

All applications should remove any non-message dialog boxes and replace them with full-screen views to create data items and change user options. During UI design, care should be taken in the display area around the input panel to allow its use when necessary.

Prohibition Against Double Starting

Applications registered into any CASIO defined category in the registry are automatically checked by the BE-300 shell to ensure that it is not double-started. Such applications therefore do not need to take any measures to prevent multiple execution. If it is not entered into a category, an application will be double-started unless specified as a Not Double Execute application in the registry. See Applications Design Guidelines and CASIO BE-300 SDK online help for more information.

Shell

Common Controls

The BE-300 supports variable controls as follow.

- Common control
- Command bar
- List View
- MessageBox
- MessageView
- Common Dialog

Soft Input Panel

The SIP for the BE-300 works in the same manner as the PPC SIP, without WC_SIPPREF controls.

Help

The BE-300 uses a custom online help system. Please refer to *Applications Design Guidelines for the Cassiopeia BE-300* for more details, including a code sample that illustrates how to integrate help file support into applications.

International

Font

English version: Tahoma, Courier

Support is scheduled for the ink input controls supported by Windows CE 4.0.

Synchronization

The CASIO PC Connect software allows the PIM data on the device to be synchronized with the desktop PC.

6

APIs

Windows APIs

Standard APIs that are included in eMbedded Visual Tools 3.0 are generally supported. BE-300 also supports multimedia API.

WinAPIs

- Automation
- OLE
- Message Queuing, Storage
- NTLMSSP
- GDI
- Time
- Clipboard
- Dlls
- Errors
- File IO
- Process and Thread
- Memory Manager
- Registry, Exception

Not Supported:

- Command Processor shell

CE APIs

- File Mapping
- IM
- Synchronization
- Notify, Database
- RAPI

Different from PPC

- Handwriting Recognition
- Input panel, Project control

Not Supported

- ActiveSync
- CEUTIL
- Pocket Outlook
- Voice recorder
- Inkx Control
- Rich Ink Control
- Game API
- Today screen API
- Transport API
- PPC User Interface API
- PPC UserInterfaceMessage

Network and Communications

- Windows network
- comm.
- Winsock
- Win User I/F Service
- TAPI

Not Supported

- Remote access
- IP helper
- SNMP
- IPC Internet
- HTML control
- Mail API

Shell and Common Control

- Progress Bar
- Rebar
- Status Bar
- Tool Bar
- Track Bar
- Command Bar

Not Supported

- Tool help

Multimedia and Direct X

Different from PPC

- Multimedia

Not Support

- Direct X
- Game X

Security

- Crypt API

Not Supported

- Smart Card
- SSPI

Driver

- Device IO
- NDIS Drivers
- File System driver

Library

Support is provided using MFC as the standard, with Cassiopeia-specific libraries also made available.

Table 8. Library APIs

API	PPC (Rapier)	Standard core	PPC compatible core
VB Runtime library	OK		OK
MFC library	OK	OK	OK
Cassiopeia Library	OK	OK	OK
Crypt Disk Library	(OK)		(OK)
Cassiopeia Embedded Library	OK	OK	OK

Runtime Library

Support is provided for MFC libraries, but not for VB.

Cassiopeia Embedded Library

Core System Loading

Loads the core system from the CF. This means switching from the system that is currently loaded, so the task must be effected using a system that does not allow the currently loaded system to hang when the core system is loaded.

Core Verification

Core verification

Digital Signatures

Support provided

Cassiopeia Corporate Library

Support is provided

Cassiopeia FDB Library

A library for front-end databases. Please refer to the BE-300 custom Database API specifications in eMbedded Visual Tools help after BE-300 SDK installation. Custom APIs and structures are detailed, demonstrating how they may be utilized to easily manipulate pre-defined databases for the BE-300.

7

ACTIVE SYNC

Inhibit Start

If Active is input to the DCD terminal when the application has not been started, ActiveSync is started. On a USB connection, this happens when a DTR signal (DSD response signal) has been set for the USB class drivers. The following method prevents this from happening:

- Do not make the DCD signal Active except when the modem is open. The DCD and DSR signals are shorted out at the main unit if the current cradle is used, and these signals should be handled with care.
- Delete the ActiveSync start settings in your application by calling the functions given below. Caution should be exercised, because, under this method, Active Sync will not be started even if the main unit is introduced into the standard cradle once the settings are created.

```
CeRunAppAtEvent( TEXT("repllog.exe"),
NOTIFICATION_EVENT_NONE );

CeRunAppAtEvent(TEXT("YourApplication.exe"),NOTIFICATIO
N_EVENT_RS232_DETECTED );
```

8

APPLICATION DISTRIBUTION/INSTALLATION

The Environment Started on Installation

Entry in the Casio Menu

The applications included as items in the Top Menu are registered in the registry under the following key:

[HKEY_LOCAL_MACHINE\Software\MENU]

The application installation tools provided by CASIO for the BE-300 will enable developers to easily add their application to the Top Menu upon installation. Therefore, developers are not required to explicitly modify the registry to add their application to the menu.

Entering Start-up Options in the Registry

Applications to be started upon device startup are defined in the registry under the following key:

[HKEY_LOCAL_MACHINE\Software\CASIO\COShell\Startup]

The CASIO Top Menu, menu.exe, is included in the default registry as a startup application.

Application Starting Times

As described in the section on Flash disk systems, applications are compressed into the FLASH. Applications are started in the following sequence upon start-up.

1. The system attempts to load the applications on the virtual program disk
2. The virtual program disk drivers open up the applications on the actual disk
3. The actual program accesses the Flash drivers, which are transferred from the Flash to the RAM buffer at high speed by means of DMA
4. The virtual program disk drivers extract the data read from the actual disk, sending it to the system buffer

Note that this procedure may make the starting times slower, despite the use of DMA transfer from the Flash.

Application Distribution and the Core System

One method of distribution is together with the application, but as Platform Builder modules are common in many of the libraries supported by the system, there is a problem with distribution by ISVs. Implementation is therefore as follows.

Applications Implemented Using Only the Built-in Standard Core System

These present no problems in terms of their creation and distribution, and are to be made available for ISVs to distribute freely.

Applications that Require the Addition of a Core System

The following two methods are allowed for the distribution of applications

1. Sold as a package

   The core system and the application are bundled for sale and distribution as a software package. In this scenario, royalties paid to MS must be included in the package, for which reason sales are licensed under the control of Casio Computer Co., Ltd.
2. Sold as unbundled applications

   Sold to users who own a core system capable of executing the application or sold to users separately. Users must be made aware of the need for a core system when being sold these applications. Such applications may, however, be distributed and sold freely.

The Core System on the CF

The core system is loaded from the CF by calling the LoadCoreSystem function in the Cassiopeia library. It is necessary to the call the above function to the core system installation program in advance.

The Core System and Applications

As a rule, the core systems that guarantee that an application will run must be certified by the application. Casio releases a number of core systems, but there are limits on these core systems. Although we do not provide a means by which to check whether an application is about to be installed on top of an appropriate core, a message is generated to allow you to identify whether this is so on start-up.

Installing Applications

Installing via a PC

The standard arrangement of the Windows application installer cannot be used as it is. Please use the tools provided by Casio Computer Co., Ltd.

Installing via the Web

Casio supports ISVs by providing the following arrangements.

1. Tools are provided to pack the application and resources created by the ISV and convert them into an EXE file so that they can be downloaded from the Web.
2. The libraries that allow installation from this EXE file are pre-embedded into the main unit.

Updating Applications

This function is not supported at present. We intend to offer a function that provides automatic updates on connection to the Cassiopeia portal site in the future.

Cassiopeia Authentication

Cassiopeia Embedded does not require Casio authentication. For the present, the logo is freely licensed for use.

9

DEVELOPMENT ENVIRONMENT

For more information on the BE-300 development environment, refer to the document, *Development Environment*

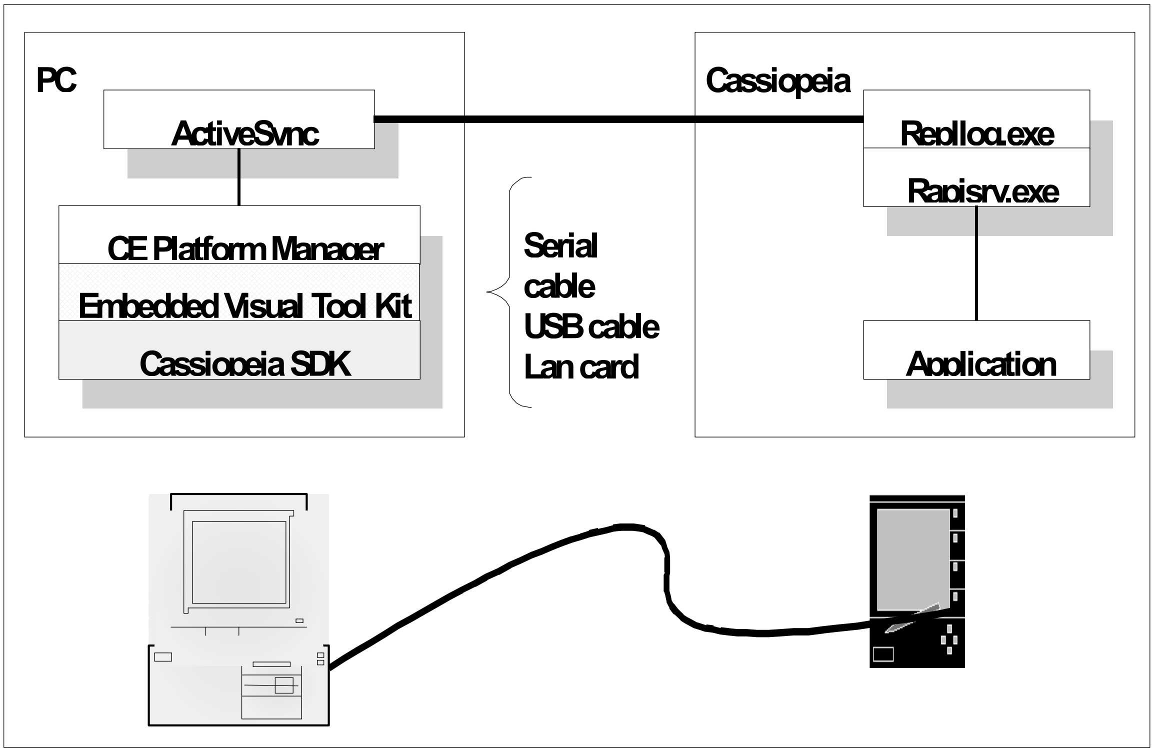
Software Development Tools

In order to develop software for the Cassiopeia BE-300, the following software is required:

- Windows CE eMbedded Visual Tools 3.0
- BE-300 SDK
- Microsoft ActiveSync 3.1

Software Development Environment

Figure 8. Development Environment 1



10

DEVELOPMENT TOOLS AND UTILITIES

To support the many different kinds of applications to be installed and run on the BE-300, the following tools are either currently available or will be available in the future:

- **PC Installer:** A tool for installing applications from a PC
- **Web Installer:** A tool to allow installation to be carried out from the Web.
- **Web Security Installer:** Adds security to the above installers. To be available in the future.
- **Registry Initializer:** A tool that initializes the registry. To be available in the future.
- **Recovery Disk Format Tool:** A tool that returns the system to its original state when the NAND Flash has been corrupted. To be available in the future.

11

POCKET PC VS. BE 300 APPLICATIONS

Differences between PocketPC and BE-300 include:

1. Differences in the peripheral I/O
2. Differences in the memory configuration
3. APIs uniquely supported by PPCs cannot be supported by BE-300
4. The shell package is different
5. Support for run-time libraries was dropped due to memory restrictions

APIs

The original PPC APIs are not supported. The BE-300 SDK provides original APIs for the Cassiopeia BE-300. See Applications Design Guidelines for an overview of these custom APIs. For more details on usage, refer to the custom API specifications found in EVC help after BE-300 SDK installation.

Implementation Design Criteria

Refer to the preceding sections for more details on these design criteria.

Memory Management

Processed according to Casio original algorithms.

Registry Management

If the registry has been modified it must be written back to the Flash area using the CGDFlashRegistry function.

File Management

GetOpenFileName() is supported. External disk access functions such as **FindFirstFlashCard** are also supported

Automatic Start-Up Path

Automatic start-up for the following ROOT is not supported.

`Root\...\<Processor Type1>\autorun.exe %1`

BE-300 will autorun an application named as follows on the CF card upon insertion:

`<CF root>\CE\R4100\aurorun.exe`

The following is not supported.

`SHGetAutoRunPath`

Power Management

- **CeRunAppAtEvent():** Runs start-up at the specified time
- **WaitCommEvent():** Starts an application on Comm connection

Installation

A custom CASIO installation tool is used for setting up an install for add-on applications on a Cassiopeia BE-300 device.

I/O

Support for IrDA is not provided (IrCOMM,COM,IrSock)

Help

A completely different HELP environment is provided

Run Time Libraries

- VB Runtime libraries are not supported.
- MFC Runtime library support has been slightly limited.

IM

The Casio BE-300 supports the Platform Builder 3.0 standard.

VERI test

No VERI test or similar procedure is used for verifying applications.

Packaged software should be tested to conform to the Casio design guidelines as defined in *Applications Design Guidelines.*