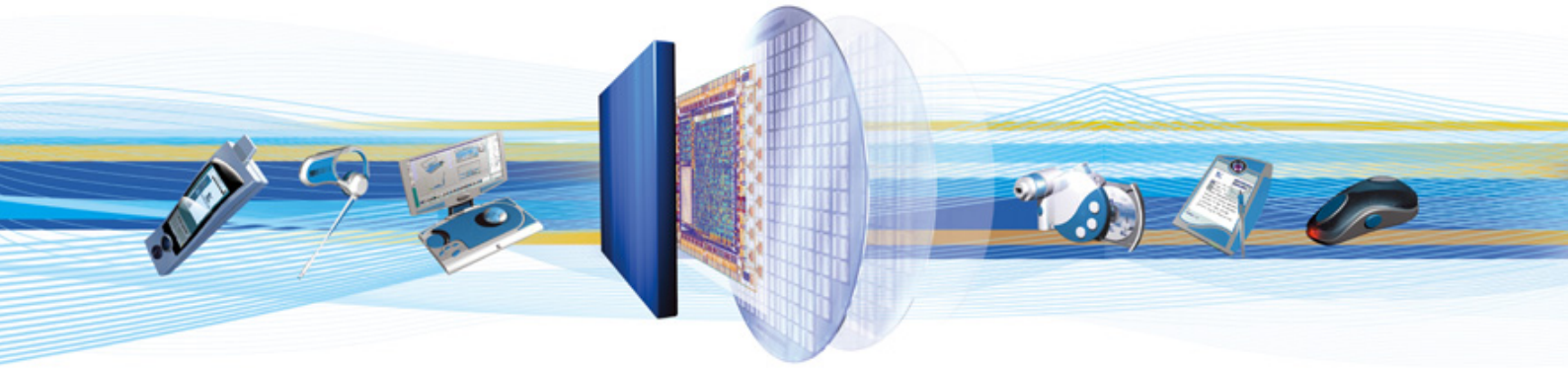
BlueCore™

Understanding and Using Bootmodes

Application Note

July 2005



CSR

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

This document is recommended reading for customers who create products that exhibit multiple modes of operation. Customers who produce single-mode devices such as headsets or mice for HCI dongles do not need to understand the contents of this document.

1.1 Understanding Bootmodes

BlueCore™ firmware exposes a high degree of configurability through the Persistent Store mechanism. The bootmode mechanism builds upon this to allow a device to contain multiple views of the Persistent Store. In the simplest case, this might allow an application to boot using different UART settings. A more complicated example is a device that can switch between operating either as an HCI transceiver or as a specialised profile proxy.

The Persistent Store view is selected when the device comes out of reset, hence the term 'bootmode'. It is not possible to change bootmodes while the device is running (although a mechanism exists to force an immediate reset into a new bootmode).

Each bootmode inherits the keys from the base Persistent Store but can override a limited number of keys. Figure 1.1 shows a device with two bootmodes. In bootmode 0, two keys are overridden, and in bootmode 1, three keys are overridden. These overridden keys are known as 'bootmode keys'. Their values are stored in a dedicated section of the base Persistent Store called a bootmode key table.

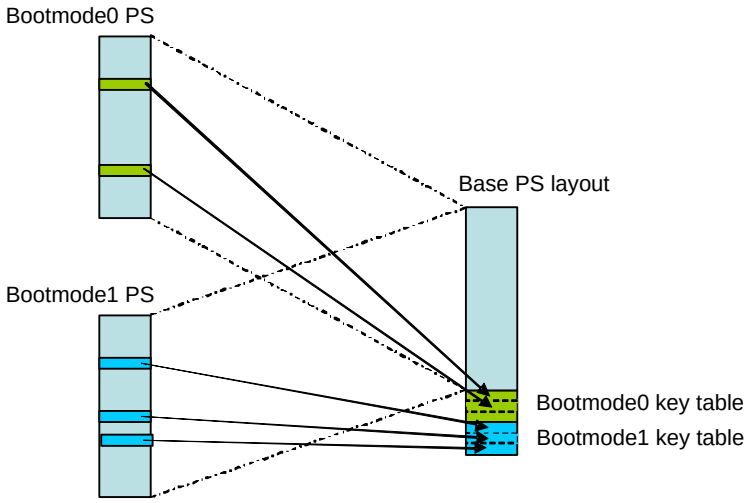


Figure 1.1: Relationship Between Bootmode Keys and Base Keys

The identifiers of bootmode keys are specified in a bootmode key list. Each bootmode has its own key list so bootmodes may override different subsets of the base keys.

There is a close relationship between the bootmode key list and the bootmode key table. The key list is a single PS Key that contains a list of identifiers for bootmode keys. The key table is a set of PS Keys containing replacement values for those keys listed. The replacement value for a bootmode key is obtained by using its position in the key list as an index into the key table.

Figure 1.2 shows a device with two bootmodes. In bootmode 0, keys 216, 434, 546 and 723 are overridden. The replacement value for key 216 is stored in BOOTMODE_KEY_TABLE_0; the replacement value for key 434 is stored in BOOTMODE_KEY_TABLE_0 +1, etc. Replacement values for bootmode 1 are stored in a similar fashion starting at BOOTMODE_KEY_TABLE_1.

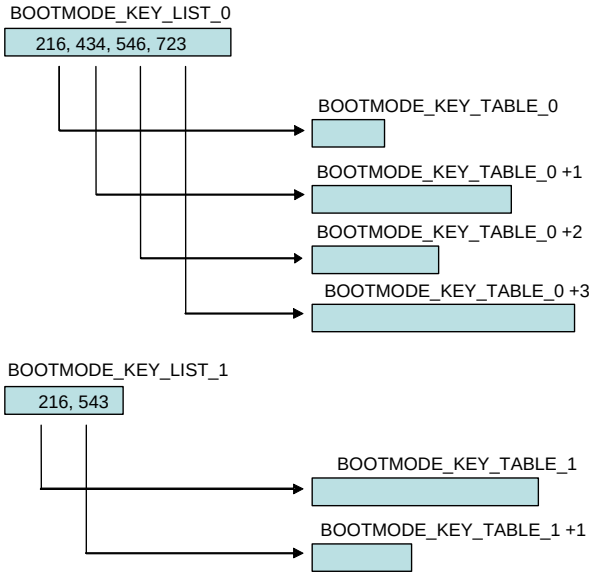


Figure 1.2: Relationship Between Key Lists and Key Tables

1.2 Using Bootmode Keys

It is extremely important to ensure that the key list and key table remain consistent. For example, adding a new value in the middle of the key list will cause all subsequent bootmode keys to refer to incorrect values. Particular care must be taken when reprogramming key lists since the new mapping described by the key list takes effect immediately. CSR strongly recommends using the following sequences.

To add a new bootmode key:

1. Read the key list and calculate the index of the last entry + 1.
2. Write the new value at that index in the key table.
3. Write a new version of the key list with the new key identifier added.

To remove a bootmode key:

1. Write PSKEY_NULL (0) into the key list at the appropriate position.
2. Delete the corresponding value in the key table.

If a value is not present in the key table, it will appear as if the bootmode key is not present; the firmware will not default to presenting the associated base key.

2 Bootmode Key Permissions (DFU and BCCMD)

BlueCore firmware provides mechanisms for controlling access to PS Keys. When the BCCMD_SECURITY_ACTIVE key is set, some keys may no longer be writable over BlueCore Command (BCCMD) and most keys cannot be changed via Device Firmware Upgrade (DFU) unless the image is signed correctly. For security reasons, the replacement values stored in the bootmode key tables inherit the permissions of their associated base keys. This requires the bootmode key list to be correctly programmed, since the firmware can only derive these permissions by performing a reverse look-up through the key list. Key lists are write-protected for both BCCMD and DFU access. This prevents them from being modified to allow inappropriate access to key tables.

BCCMD accesses keys via the active bootmode mapping. This means (in general) that an attempt to modify a base key that has been overridden by a bootmode key will result in the bootmode key being changed. It can sometimes be useful to change this behaviour so that the base keys can be manipulated directly. This can be done by setting bit 7 of the stores parameter passed in the BCCMD.

The DFU process always operates on base key identifiers. Therefore, DFU files should always refer to bootmode keys via the identifiers shown in Appendix A .

3 Changing Bootmodes

When BlueCore boots up after a cold reset (e.g., a power-up), the bootmode will be set to the value stored in PSKEY_INITIAL_BOOTMODE. If this key is not present, the bootmode will be set to 0.

If the bootmode key list is not present (or is present but empty), the bootmode will simply inherit the complete set of base keys.

A virtual machine (VM) application may read the current bootmode using the BootGetMode function. BootSetMode may be used to force an immediate warm reset into the requested bootmode. Subsequent warm resets will use the initial bootmode. (An application can pass one word of state across the warm reset using BootGetPreservedWord and BootSetPreservedWord.)

BCCMD provides an alternate mechanism; the GET/SET_REQUEST commands may be used to retrieve the current bootmode or to force a reset into a new bootmode.

Note:

A BCCMD_GET_RESPONSE acknowledgement may not be returned to the host when a new bootmode is set.

4 Bootmodes During DFU

4.1 Loader Build IDs 1372 and Above

The information in this section applies to boot loaders with a build ID of 1372 or greater. The loader build ID may be retrieved from BlueCore using BCCMD commands, described in [BCCMDS].

Note:

On 18.x and Unified18x builds only, the build ID of the loader differs from that of the stack with which it is released. To avoid confusion, use BCCMD to check which loader is present in the release you are using.

When BlueCore is rebooted to start the boot loader at the beginning of a DFU operation, it enters bootmode 0, regardless of the setting of PSKEY_INITIAL_BOOTMODE. Customers who wish to support DFU should ensure that the host transport settings in this bootmode are suitable for their preferred DFU method.

At the end of a successful DFU operation, the host-side DFU Wizard (see [DFUIF]) performs a warm reboot of BlueCore to start the newly downloaded stack. The DFU Wizard then attempts to make an HCI connection to BlueCore to check that the new stack is running. On this first warm reboot after the DFU download, the new stack on BlueCore will enter bootmode 0, regardless of the setting of PSKEY_INITIAL_BOOTMODE. The DFU Wizard gives an error if it cannot establish the HCI connection, so customers who wish to support DFU using DFU Wizard should ensure that the new stack allows an HCI connection when in bootmode 0.

A common use of bootmodes is to provide a USB dongle that presents itself to the host as a USB Human Interface Device (HID) in one bootmode and as a Bluetooth® device using the Host Controller Interface in another bootmode. To support DFU as described above, customers should configure BlueCore to present the HCI interface in bootmode 0 and the HID interface in bootmode 1. This arrangement of bootmodes is used in the sample HID dual boot dongle application provided with BlueLab.

4.2 Older Loaders

Customers with products in the field that meet the following conditions may find that BlueCore does not perform DFU over USB correctly:

- Use bootmodes
- Contain loaders with a build ID of less than 1372
- Do not present an HCI interface in the initial bootmode

The symptom of this problem is that BlueCore does not enumerate on the USB bus after it has been rebooted to start the DFU download process.

To work around this problem, customers may need a special DFU process which sets PSKEY_INITIAL_BOOTMODE to an HCI bootmode (using BCCMD, for example) before the DFU begins, and resets it to their desired initial bootmode at the end of the DFU.

Appendix A Key Identifiers

PS Key	Identifier
PSKEY_INITIAL_BOOTMODE	973
BOOTMODE_KEY_LIST_0	1200
BOOTMODE_KEY_LIST_1	1201
BOOTMODE_KEY_LIST_2	1202
BOOTMODE_KEY_LIST_3	1203
BOOTMODE_KEY_LIST_4	1204
BOOTMODE_KEY_LIST_5	1205
BOOTMODE_KEY_LIST_6	1206
BOOTMODE_KEY_LIST_7	1207
BOOTMODE_KEY_TABLE_0	1208
BOOTMODE_KEY_TABLE_1	1272
BOOTMODE_KEY_TABLE_2	1336
BOOTMODE_KEY_TABLE_3	1400
BOOTMODE_KEY_TABLE_4	1464
BOOTMODE_KEY_TABLE_5	1528
BOOTMODE_KEY_TABLE_6	1592
BOOTMODE_KEY_TABLE_7	1656

Table A.1: Key Identifiers

Appendix B Worked Example

Consider a hypothetical device based on the serial-cable replacement application supplied with BlueLab. This application has been modified so that it will communicate with any Bluetooth Serial Port Profile (SPP) device, exposing the data stream over either the serial port or the USB bus, where the device will enumerate as a virtual com-port. In addition, the device can operate as a standard USB HCI dongle. There are three main modes of operation:

- Mode 0: VM application is inactive, HCI is exposed to host over USB
- Mode 1: VM application controls radio, transport is USER
- Mode 2: VM application controls radio, transport is USB

Note:

As this application supports HCI over USB in bootmode 0, it is suitable for DFU over USB, as described in Section 4.

At a minimum, each of these bootmodes needs to set the following keys according to Table B.1 (values for symbolic names are shown in brackets).

Key	Mode 0	Mode 1	Mode 2
HOST_INTERFACE (505)	USB (2)	USER (4)	USB (2)
ONCHIP_HCI_CLIENT (972)	FALSE (0)	TRUE (1)	TRUE (1)

Table B.1: Bootmode Keys

This results in a set of keys that appear as follows:

```
// Mode 0 keys
BOOTMODE_KEY_LIST_0 (1200) = {HOST_INTERFACE (505) , ONCHIP_HCI_CLIENT (972)}
BOOTMODE_KEY_TABLE_0 (1208) = USB (2)
BOOTMODE_KEY_TABLE_0+1 (1209) = FALSE (0)

// Mode 1 keys
BOOTMODE_KEY_LIST_1 (1201) = {HOST_INTERFACE (505) , ONCHIP_HCI_CLIENT (972)}
BOOTMODE_KEY_TABLE_1 (1272) = USER (4)
BOOTMODE_KEY_TABLE_1+1 (1273) = TRUE (1)

// Mode 2 keys
BOOTMODE_KEY_LIST_2 (1202) = {HOST_INTERFACE (505) , ONCHIP_HCI_CLIENT (972)}
BOOTMODE_KEY_TABLE_2 (1336) = USB (2)
BOOTMODE_KEY_TABLE_2+1 (1337) = TRUE (1)
```

However, it is possible to minimise the number of keys used by moving some of the common values in the base set:

```
//Common base keys
HOST_INTERFACE (505) = USB (2) //default to using USB
ONCHIP_HCI_CLIENT (972)= TRUE // default to ONCHIP_HCI_CLIENT

// Mode 0 keys – just override routing of HCI
BOOTMODE_KEY_LIST_0 (1200) = {ONCHIP_HCI_CLIENT (972)}
BOOTMODE_KEY_TABLE_0 (1208) = FALSE (0)

// Mode 1 keys – just override transport
BOOTMODE_KEY_LIST_1 (1201) = {HOST_INTERFACE (505)}
BOOTMODE_KEY_TABLE_1 (1272) = USER (4)

// Mode 2 keys no override necessary: empty or non-existent key list means that all values will be inherited
from base Persistent Store.

BOOTMODE_KEY_LIST_2 (1202) = {}
```

Document References

Document ID	Document Title	CSR Reference
[BCCMDS]	BCCMD Commands	bcore-sp-005P
[DFUIF]	Device Firmware Upgrade Sample User Interface	bc01-an-096

Terms and Definitions

BCCMD	BlueCore Command
BlueCore™	Group term for CSR's range of Bluetooth chips
BlueLab™	CSR's development toolset for building applications to run in the firmware's VM
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
DFU	Device Firmware Upgrade
HCI	Host Controller Interface
HID	Human Interface Device
Persistent Store	Storage of BlueCore's configuration values in non-volatile memory
PS Key	Persistent Store Key
SPP	Serial Port Profile
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VM	Virtual Machine

Document History

Revision	Date	History
a	11 Jun 03	Original publication of this document.
b	07 Aug 03	Amended Changing Bootmodes instructions in Section 3, Changing Bootmodes.
c	20 May 04	Added Section 4, Bootmodes During DFU. Amended Appendix B , Worked Example, so bootmode 0 is suitable for DFU.
d	28 Jul 05	Added a note that the build ID of the loader differs from that of the stack on 18.x and later builds. Minor formatting changes.

BlueCore™

Understanding and Using Bootmodes

Application Note

bcore-an-019Pd

July 2005