U-Boot Boot Device Design Specification

Author: Jared Lewis
Email: jared.lewis@radisys.com

002-xxxxx-0001
Revision 0.3
July 31, 2023

Note: To verify signoff, see the on-line repository.

Name	Title/Department	Sign-Off Date
Approvers:		
Michael Hu	Architect	
Himanshu Parikh	Project Manager	
Nilan Naidoo	Architect	
Reviewers:		
Brian Dominy	Software Engineer	
Kim Kempf	Software Engineer	
Dave Lyons	Software Engineer	
Carl Thomson	Software Engineer	
Haile Seifu	Software Engineer	
Alan Anderson	Software Engineer	

Document control

This is a copy of a RadiSys controlled document. This document is valid only if it is a copy of the latest version available in the on-line repository or from the document owner. To validate the latest version of this document, refer to the on-line repository or contact the document owner.

On-line repository: Agile
Source filename: U-Boot_Boot_Device_Design_Specification_v0.3.doc

Preface

Revision History

All revisions that start with zero (i.e., 0.5, 0.65, etc.) are preliminary drafts for internal reviews.

Table 1. Revision history

No.	Date	Author	Description
0.1	05/11/2009	Jared Lewis	Initial draft.
0.2	05/21/2009	Jared Lewis	• Corrected Figure 2 – arrows were going in the wrong direction • Updated Figure 1 to reflect new implementation. • Added a new configuration option to allow customers to select whether or not U-Boot should try booting from the boot device list forever. • Removed SCM3 UShell application references. • Updated the DHCP section based on implementation changes. • Updated the configuration options based on changes to implementation. These were most focused on the Ethernet and BootDevice list changes.
0.3	07/22/2009	Jared Lewis	• Updated figure 1 per changes in implementation (adding the bootlist environment variable instead of using predefined boot device values). • Added information about the ability to configure which ports on the bmd0 devices are used. • Added information about support for getting the ushell application from JFFS. • Updated DHCP Option 61 request information to include latest implementation. • Changed all references of NetBoot to UShell.

Contents

Figures

Tables

1. U-Boot Boot Device Process

1.1 Overview

The RadiSys implementation of booting a PowerQUICC-based payload processor to an operating system is based on software entities called boot devices. Boot devices are software entities that define where an operating system will be found and how it will be retrieved. For RadiSys boards, the following boot device types are supported.

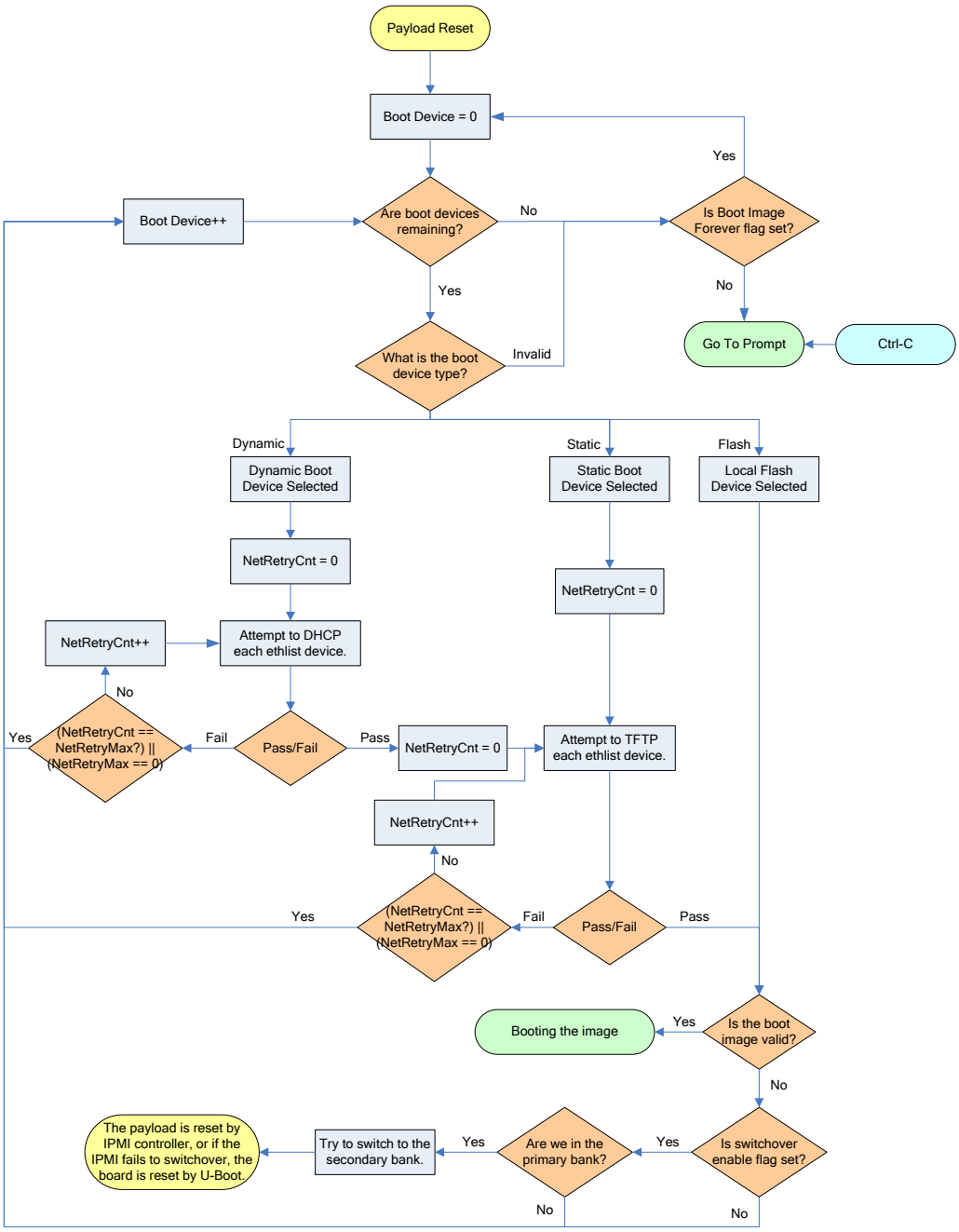
Table 2. Boot Device Types

Default Boot Order	Boot Device	Description
1	Local Flash	This boot device attempts to load the “active” image from flash. RadiSys blades have two flash banks where the currently selected flash bank is called the “active” flash bank.
2	Dynamic Network	This boot device dynamically retrieves and loads an image based on a DHCP request and response. The image name, server IP address, and board IP address are provided by the DHCP server. The image is retrieved using the TFTP protocol. See the **DHCP Request** section for information about the DHCP request format.
3	Static Network	This boot process statically retrieves and loads an image based on statically defined U-Boot environment variables. These variables include the server IP address, local IP address, and image name. The image is retrieved using the TFTP protocol.

Boot devices are managed using an environment variable that represents a list of boot device types. The order and selection of boot devices can be changed. When U-Boot needs to boot an image, it will step through each boot device and attempt to boot from that device. The number of network retries can be configured. Ethernet devices (i.e. eTSEC0, eTSEC1, bmd0, etc) are managed using environment variables. For more information about configuration options, see the **Configuration Options** section.

The figure below describes the image-boot process in more detail.

Figure 1. U-Boot Boot Device Flowchart



For boot devices using the network to retrieve an image, only specific network ports are used. By default, all TSEC interfaces are included. For a list of a board's TSEC options, refer to the board's reference manual. For node blades, the maintenance Ethernet ports and base interface channels 1 and 2 are supported. For RadiSys switch blades, only the TSEC interfaces are supported. More information about which network ports are used can be found in the **UShell Application Design** section.

1.2 DHCP Request

U-Boot uses a simple DHCP request format that can include the DHCP Client Identification option (DHCP Option 61). When a DHCP request is performed, U-Boot will send a "Get Device ID" IPMI command to the shelf manager to determine what kind of shelf manager is being used. The IANA value in the response is used to determine the type of shelf manager. For Pigeon Point Shelf Managers that might have custom shelf address information depending on the vendor, the third auxiliary byte is used to determine the vendor type (i.e. NSN).

Once the shelf manager type has been learned, U-Boot will construct a shelf-specific Option 61 section that will be included as part of the DHCP request. If the shelf manager type is not a recognized kind, or if U-Boot fails to create the Option 61 section, the DHCP packet will be sent without the Option 61 fields being included.

1.2.1 RadiSys-specific DHCP Option 61

The format of the basic DHCP Option 61 is described in the figure and table below.

Figure 2. RadiSys-specific DHCP Option 61 Format

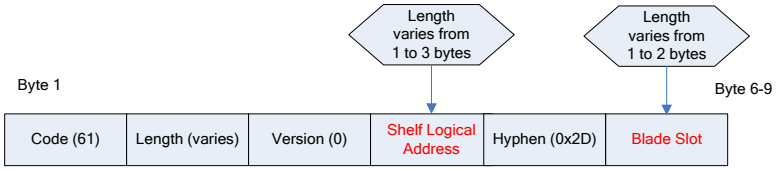


Table 3. RadiSys-specific DHCP Option 61 Fields

Field	Description	Type	Range	Default Value	Data Source
Code	Option Code	Decimal	61	61	Fixed
Length	Length of the Option Code	Decimal	4-7	4-7	Varies
Version	Version of the Option 61 Code	Decimal	0	0	Fixed
Shelf Logical Address	The decimal expansion of the Logical Shelf portion of the Shelf Address value (i.e., 1, 2, or 3 ASCII characters in the range 30H through 39H representing a 1-, 2-, or 3-digit number withno leading zeroes)	Hex	30-39h	varies	Get Shelf Address Info command – This is the equivalent of the Logical Chassis Number.
Hyphen	A hyphen to seprate the shelf's logical address and blade slot	Hex	2Dh	2Dh	Fixed

Table 3. RadiSys-specific DHCP Option 61 Fields

Field	Description	Type	Range	Default Value	Data Source
Blade Slot	The decimal expansion of the Physical Slot value (i.e., 1 or 2 ASCII characters in the range 30H through 39H representing a 1- or 2-digit number with no leading zeroes)	Hex	30-39h	varies	Get Address Info command

An example of a DHCP Client Identifier is 3D-06-00-32-34-35-2D-37, based on a Logical Shelf value of 245 and a Physical Slot value of 7.

1.2.2 NSN-specific DHCP Option 61

The format of the NSN-specific DHCP Option 61 is described in the figure and table below.

Figure 3. NSN-specific DHCP Option 61 Format

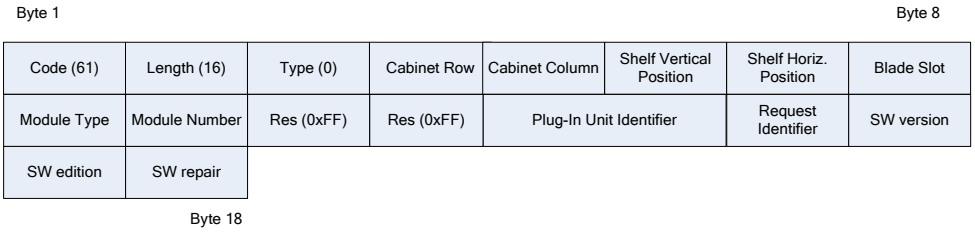


Table 4. NSN-specific DHCP Option 61 Fields

Field	Description	Type	Range	Default Value	Data Source
Code	Option Code	Decimal	61	61	Fixed
Length	Length of the Option Code	Decimal	16	16	Fixed
Type	Version of the Option 61 Code	Decimal	0	0	Fixed
Cabinet Row	Cabinet row location in the central office. Number is one of the coordinates used to indicate where the cabinet is installed in the central office.	Hex	00..FEh	FFh	Get Shelf Address Info command
Cabinet Column	Cabinet column in the central office. Number is the second coordinate used to indicate where the cabinet is installed in the central office.	ASCII	A…Z	FFh	Get Shelf Address Info command

Table 4. NSN-specific DHCP Option 61 Fields

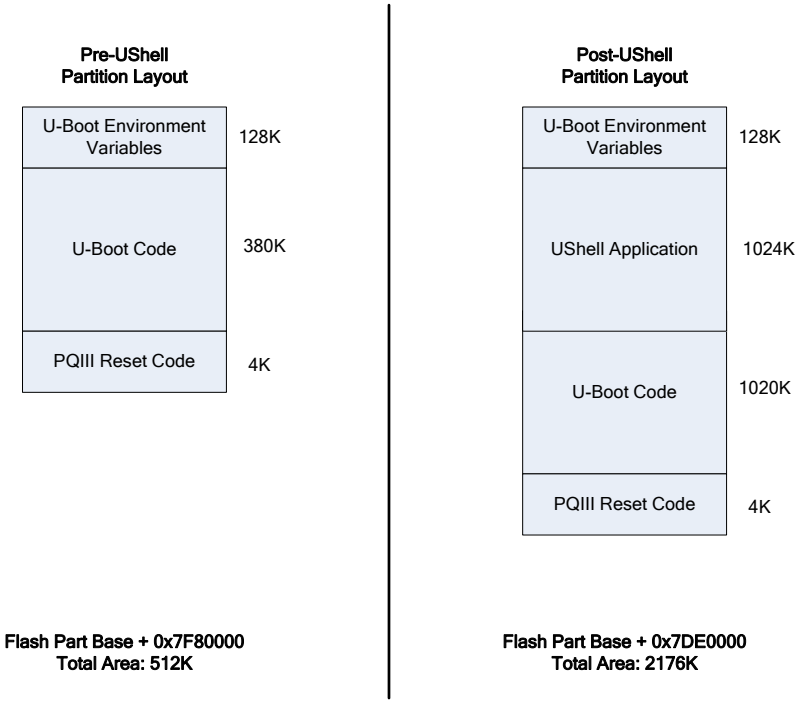
Field	Description	Type	Range	Default Value	Data Source
Shelf Vertical Position	Vertical position of the shelf in the cabinet	Hex	00..FEh	FFh	Get Shelf Address Info command
Shelf Horizontal Position	Horizontal position of the shelf in the cabinet.	Hex	00..FEh	FFh	Get Shelf Address Info command
Blade Slot	Blade's slot position in the chassis.	Hex	00..FEh	FFh	Get Address Info command
Module Type	Contains Site Type value as defined in PICMG 3.0	Hex	00..FFh	FFh	Get Address Info command
Module Number	Contains Site Number value as defined in PICMG 3.0	Hex	00..FFh	FFh	Get Address Info command
Logical Cabinet	Logical Cabinet Address as defined by the Shelf FRU information.	Hex	00..FFh	FFh	Get Shelf Address Info Command
Logical Shelf	Logical Shelf Address as defined by the Shelf FRU information.	Hex	00..FFh	FFh	Get Shelf Address Info Command
Plug-in Unit Identifier	Blade's Product Identification Value	Hex	0000… FFFFh	FFFFh	Retrieved from FRU information at fixed offset **0x013D**.
Request Identifier	Indicates the processor or interface inside the module that requests an IP address.	Hex	00..FEh	01h	Defaults to 1 in U-Boot.
Software Version	SW version of the SW that requests an IP address.	Hex	00..FEh	FFh	Not used by U-Boot. Set to the default value.
Software Edition	SW edition of the SW that requests an IP address.	Hex	00..FEh	FFh	Not used by U-Boot. Set to the default value.
Software Repair	SW repair of the SW that requests an IP address.	Hex	00..FEh	FFh	Not used by U-Boot. Set to the default value.

1.3 Flash Partition Changes

All current RadiSys blades must go through onboard Broadcom switches to gain access to the base channel interfaces. In order to configure and pass traffic through these switches, U-Boot makes use of the Broadcom Mini-Driver Kit (MDK), which is a set of libraries provided by Broadcom. RadiSys incorporates the MDK libraries into a standalone application called UShell, which can be included as part of the U-Boot image or retrieved out of the JFFS partition. U-Boot calls the standalone application when needed to create new Ethernet interfaces as well as to send and receive traffic over the switches.

To support the UShell application from flash, the size and partitions of U-Boot were changed. U-Boot now includes partitions for the UShell standalone application, environment variables, U-Boot code, and PowerQUICC reset code. The figure below shows the pre and post UShell layouts.

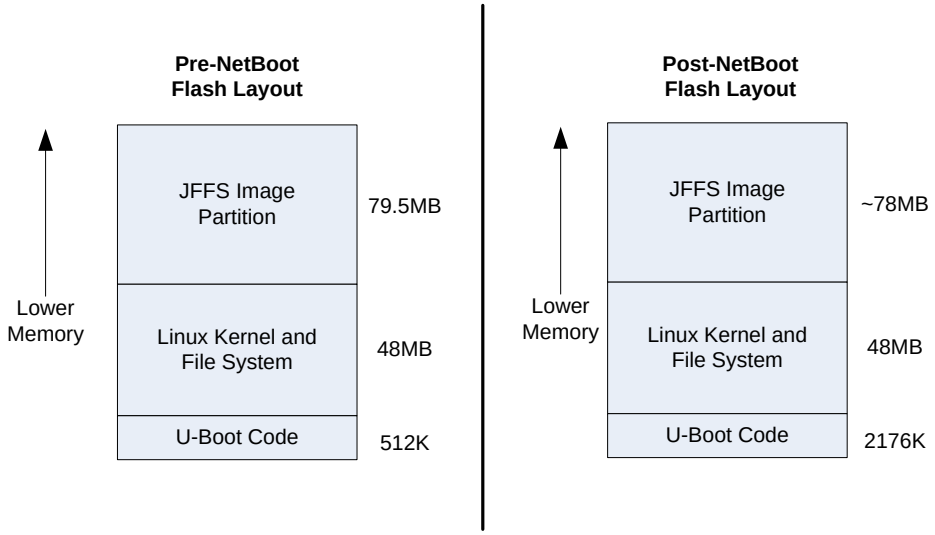
Figure 4. U-Boot Partition Changes



** Note: The Pre-UShell figure is not accurate for the ATCA-9100 blade, which had a U-Boot size of 1024Kbyte before UShell was updated.*

The size and location of each board's flash partitions have been updated to work with the latest version of U-Boot. The following diagram shows the pre and post UShell flash memory layouts.

Figure 5. Flash Memory Device Changes



Due to the change in flash partition sizes, the entire flash must be updated when upgrading to or downgrading from UShell software.

2. UShell Application Design

2.1 Overview

This section describes the features and design of the UShell application.

2.2 UShell Description

The UShell application is a standalone U-Boot application that incorporates the Broadcom Mini-Driver Kit (MDK). Its primary purpose is to configure Broadcom switches on a board, and to provide U-Boot with an API to send and receive packets over the Broadcom ports.

The UShell application is designed to support multiple RadiSys blades including; ATCA-1200, ATCA-9100, ATCA7220, and ATCA-2210. The application learns which board it is running on by reading an environment variable called "boardtype". The "boardtype" environment variable is set internally by U-Boot.

UShell is called by U-Boot in two situations; during U-Boot's initialization routine, and when U-Boot needs to communicate over the Ethernet switch. Users may also manually access the application by using the "go" command at the U-Boot prompt.

The UShell location can be stored in two different locations; as part of U-Boot or as part of the JFFS file system. Users can specify the location of UShell using an environment variable called "ushellLoc." Users can also define where the UShell application is found in the JFFS file system using the environment variable "ushellfile." The UShell application is stored as a file in both the JFFS file system and flash. The file is created using the Linux utility "mkimage", which provides U-Boot with a way to verify that the file is valid before attempting to jump to it. To save flash space, UShell is compressed using "gzip" before being encapsulated as a file.

When UShell is called during initialization it configures switches on the board, and then registers an Ethernet device with U-Boot using the "eth_register" function. One device is registered for each switch on a board. The application designates each switch with the name "bmd***X***".

Each Ethernet device in U-Boot includes four function parameters: init, halt, send, and receive. The initialization function for UShell only initializes a specific set of ports for a board that are hard coded as part of the application code. All other ports are left in a disabled state to prevent creating unexpected network connections.

The "send" function for UShell Ethernet devices only tries to send a packet on one port per U-Boot send request. This is done to prevent duplicate packets from being sent from a switch. The send function learns which port to use by reading an environment variable called "bmd0port". This variable is set by U-Boot as it steps through a list of ports defined by the environment variable "bmd0portlist". When U-Boot reaches the end of the port list, it will start over again. Users may update the port list to include or exclude ports as needed.

Note: Only the Ethernet device" bmd0" is supported for network booting at this time.

The following table describes which switches and associated ports are used by the UShell application. The port name column in the table below applies to the "bmd0portlist" environment variable.

Table 5. UShell Switch Device Mappings

Board	Switch	Device Name	Broadcom Ports	Port Name
ATCA-1200	BCM56502	bmd0	Base Channel 1 – GE21	bi0
			Base Channel 2 – GE22	bi1

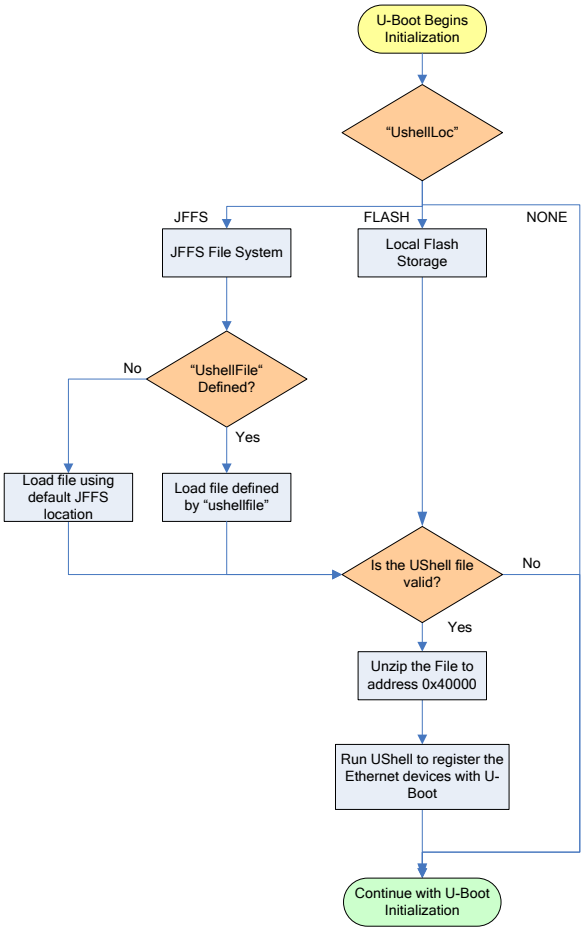
Table 5. UShell Switch Device Mappings

Board	Switch	Device Name	Broadcom Ports	Port Name
ATCA-9100	BCM56500	bmd0	Base Channel 1 – GE0	bi0
			Base Channel 2 – GE1	bi1
	BCM56502	bmd1	Not Used	
ATCA-7220	BCM56317	bmd0	Base Channel 1 – GE0	bi0
			Base Channel 2 – GE1	bi1
	BCM56801	bmd1	Not Used	

UShell assigns a MAC address to each switch's CMIC port. All packets that are received by the switch are forwarded to the payload processor.

The following diagram illustrates how U-Boot loads the UShell application.

Figure 6. U-Boot UShell Initialization Flowchart



E-Key states are not checked as part of the initialization process, because the only supported ports for UShell are base channels, COM-E, or maintenance ports. According to the PICMG 3.0 specification, blades are not required to E-Key base channel ports. Maintenance and COM-E ports do not have associated E-Key descriptors so these ports do not need to be keyed.

3. Configuration Options

Several elements of the U-Boot Boot Device process are configurable to provide customers with more flexibility in their implementation. All configurable options are managed as U-Boot environment variables.

Currently, environment variables can be changed by accessing U-Boot through the serial console, or by using the Linux applications “fw_printenv” and “fw_setenv”. These applications are included as part of RadiSys Linux distributions.

The following table describes all of the boot process configuration options.

Table 6. Boot Process Configuration Options

Configuration Option	Variable	Factory Default
Set the Board Type (**note:** *This field is set by U-Boot code and should not be changed*!)	boardtype	<blade-specific> (i.e. atca1200)
Sets the number of DHCP, ARP, or TFTP retries that should be performed before moving on to the next boot device.	netretrymax	3 (Set to zero for infinite loops.)
If all boot devices fail, this flag indicates whether or not U-Boot should step through the devices again or if it should go to the prompt. This could be used to try to network boot forever.	bootdevforever	n
Sets whether or not the board should switch to the secondary flash bank if a boot image is found to be invalid.	noswitchflash	n
Indicates the board’s boot device list. The list is separated by colons. For example, the list could be “bootlist=dynamic:static”. Three options are supported: “dynamic”, “static”, and “flash”.	bootlist	flash
Indicates the board’s Ethernet list. The list is separated by colons. For example, the list could be “ethlist=bmd0:eTSEC1”. The Ethernet list will vary based on board type, but can include the TSEC devices (i.e. eTSECx) and the Broadcom devices (i.e. bmdx) if supported.	ethlist	bmd0
Indicates which ports to use on Ethernet device bmd0. This setting only applies to sending packets. The order that the ports are used depends on the order of the list.	bmd0portlist	bi0:bi1
Sets the static local IP address.	staticipaddr	10.0.0.1
Sets the static server IP address.	staticserverip	10.0.0.2

Table 6. Boot Process Configuration Options

Configuration Option	Variable	Factory Default
Sets the static image name to retrieve.	staticbootfile	<blade-specific> (i.e. atca1200.bin)
Sets the local MAC address for the board. The MAC address is assigned to each interface when the interface is used to send packets.	ethaddr	<varies> (i.e. 00:E0:0C:00:00:FD)
Sets the location of the UShell application. By default, no location is defined so the standalone application will not get loaded by U-Boot.	ushellLoc	Default: “none” Options: “none” – UShell is not loaded “flash” – Loads UShell from flash. “jffs” – Loads UShell from a file in the JFFS partition.
Sets the file location of UShell application in JFFS. If the JFFS location is not specified and ushellLoc equals “jffs”, a default location of “onboot.data/bindmount/etc/uboot-apps/ushell” is used.	ushellfile	Default: <undefined>