1 JZ4760 Boot ROM Specification

The JZ4760 contains an internal 8KB boot ROM. The CPU boots from the boot ROM after reset.

1.1 Boot Select

The boot sequence of the JZ4760 is controlled by boot_sel[2:0]. The configuration is shown as follow:

Table 1 Boot Configuration of JZ4760

boot_sel[2:0]	Boot method
111	NAND boot @ CS1
100	SD boot @ MSC0
101	SPI boot @ SPI0/CE0
110	USB boot @ USB 2.0 device, EXTCLK=12MHz
000	USB boot @ USB 2.0 device, EXTCLK=13MHz
001	USB boot @ USB 2.0 device, EXTCLK=26MHz
010	USB boot @ USB 2.0 device, EXTCLK=19.2MHz
011	NOR boot @ CS2 (just for FPGA testing)

1.2 Boot Procedure

After reset, the boot program on the internal boot ROM executes as follows:

1. Disable all interrupts and read boot_sel[2:0] to determine the boot method.
2. If it is boot from NAND flash, 4 flags(whose length is 160 bytes) at the beginnig of NAND are read to know the NAND information including bus width(8 or 16 bits), page cycle(2 or 3 cycles) and its page size(512B, 2KB, 4KB or 8KB). Then 8KB code are read out from NAND to cache, if the 8KB reading failed, the next 8KB backup in NAND will be read. Then branch to cache at 160 bytes offset.
   There 8KB backup reading failed, the 8KB backup at 64^{th}, 128^{th}, 192^{th} , …, and finally 1280^{th} page will be tried in consecutive order.
3. If it is boot from MMC/SD card at MSC0, its function pins MSC0_D0, MSC0_CLK, MSC0_CMD are initialized, the boot program loads the 8KB code from MMC/SD card to cache and jump to it. Only one data bus which is MSC0_D0 is used. The clock EXTCLK/128 is used initially. When reading data, the clock EXTCLK/2 is used.
4. If it is boot from USB, a block of code will be received through USB cable connected with host

PC and be stored in cache. Then branch to this area in cache.

Note: The JZ4760's cache is 16KB, its address is from 0x80000000 to 0x80004000.

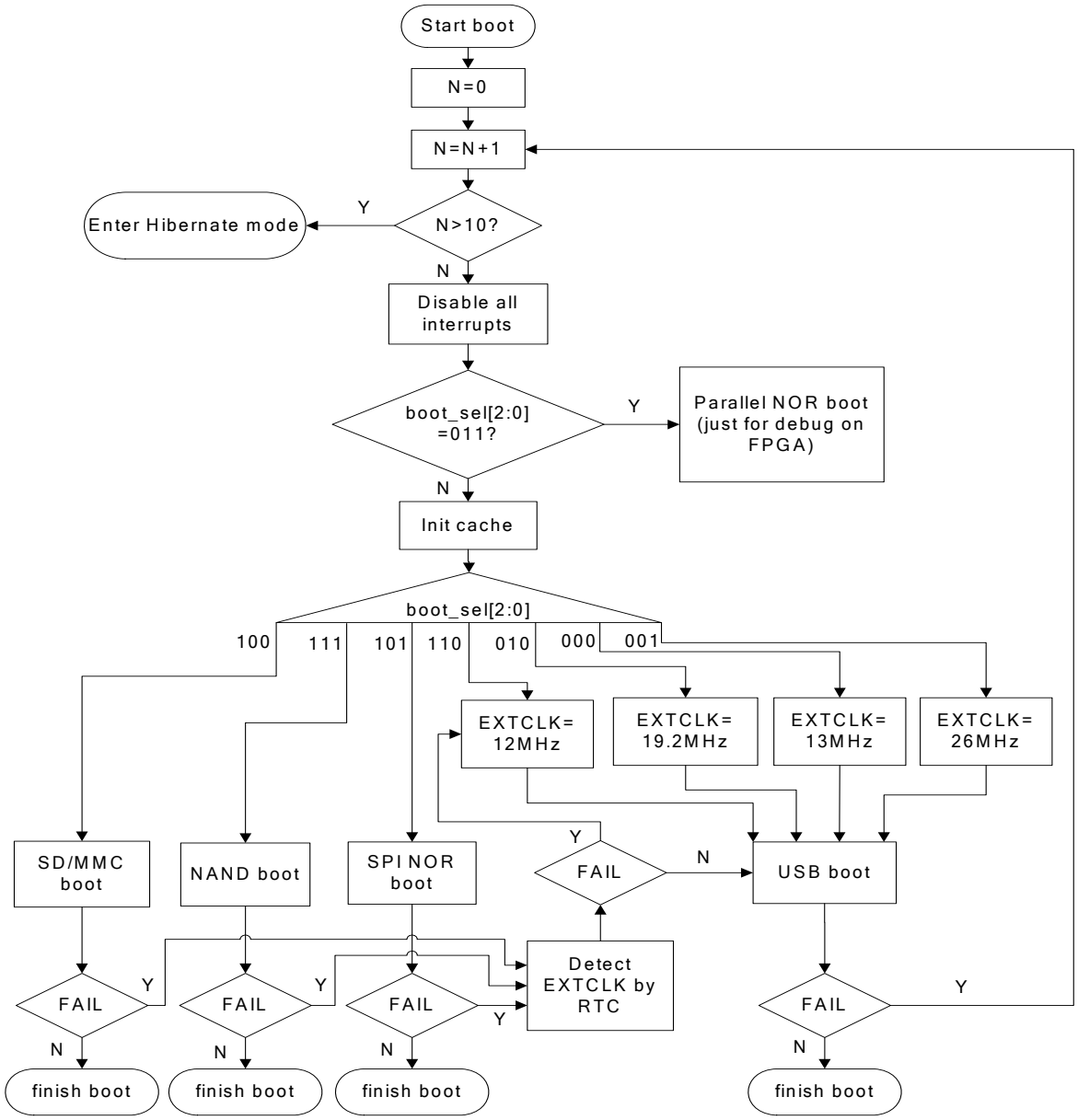

Figure 1 Boot sequence diagram of JZ4760

1.3 NAND Boot Specification

If CPU boots from NAND flash (CS1), the boot ROM will read 4 flags from NAND flash to know the NAND information including bus width(8 or 16 bits), page cycle(2 or 3 cycles) and its page size(512, 2KB, 4KB or 8KB bytes).
The content and definition of the 4 flags are shown as follow:

Table 2 The definition of 4 flags in NAND flash

Name	Location (in byte)	length (in byte)	Value	Description
buswidth_flag	0-63	64	0x55 or 0xaa	Bus width. 0x55: 8bit bus width, 0xaa: 16bit bus width.
rowcycle_flag	64-95	32	0x55 or 0xaa	The number of bytes of row cycles. 0x55: 2-byte row cycles, 0xaa: 3-byte row cycles.
pagesize_flag1	96-127	32	0x55 or 0xaa	pagesize_flag1 pagesize_flag0 pagesize(byte) 0x55 0x55 512 0x55 0xaa 2048 0xaa 0x55 4096 0xaa 0xaa 8192
pagesize_flag0	128-159	32	0x55 or 0xaa	

The buswidth_flag containing 64 bytes locates at the beginning of NAND, if the bus width of NAND is 8 bit, the buswidth_flag should be filled with 0x55 for all 64 bytes, or else it should be filled with 0xaa. The rowcycle_flag containing 32 bytes locates behind the buswidth_flag, if the number of bytes of row cycles is 2, the flag should be filled with 0x55 for all 32 bytes, or else it should be filled with 0xaa. The pagesize_flag1 and pagesize_flag0 each containing 32 bytes locate behind the rowcycle_flag, which value should be filled is determined by the page size of NAND. Please refer to table 2. Totally, 160 bytes are allocated for the 4 flags.

At first, the first 256 bytes (which is not a PN* unit) in NAND containing 4 flags will be read out to a buffer assuming the bus width of NAND is 8 bit. The buswidth_flag will be obtained from the 256-byte buffer to a 32-byte buffer to detect the page size(whether 512B or not) and bus width of nand. If there are more 7 bytes of 0x55 in buswidth_flag, then the bus width is 8 bit; else if there are more than 7 bytes of 0xaa, then the bus width is 16 bit, else, the position stored 8KB codes reading failed, 64^{th}, 128^{th}, 192^{th}, ..., 1280^{th} page will be tried in sequence. If failed at 1280^{th} page, bootrom will jump to usb_boot. If buswidth_flag is valid, the rowcycle_flag will be obtained from the buffer to know the number of row cycles. At last, pagesize_flag1 and pagesize_flag0 will be obtained from the buffer to know precise page size.

8KB codes in NAND will be loaded up to dcache and transferred to icache and branch to icache at 160 bytes offset. Hardware PN and 24-bit BCH ECC will be used for every 256 bytes during reading. The ECC(39 bytes per 256 bytes data) stores in the data area of a NAND page behind the page storing code data. If no ECC error is detected or ECC error is correctable(number of error bits <= 24), NAND boots successfully. If uncorrectable error occurred, next 8KB backup at 64 pages behind will be tried. 64th, 128th, 192th, ..., 1280th page will be tried in sequence. If failed at 1280th page, bootrom will jump to usb_boot.

The distribution and structure of the boot code in NAND is shown as figure 2.
The procedure of the JZ4760 NAND boot is shown as figure 3.

* Note: PN is short for pseudorandom noise which is used for supporting TLC (three-level cell) NAND.

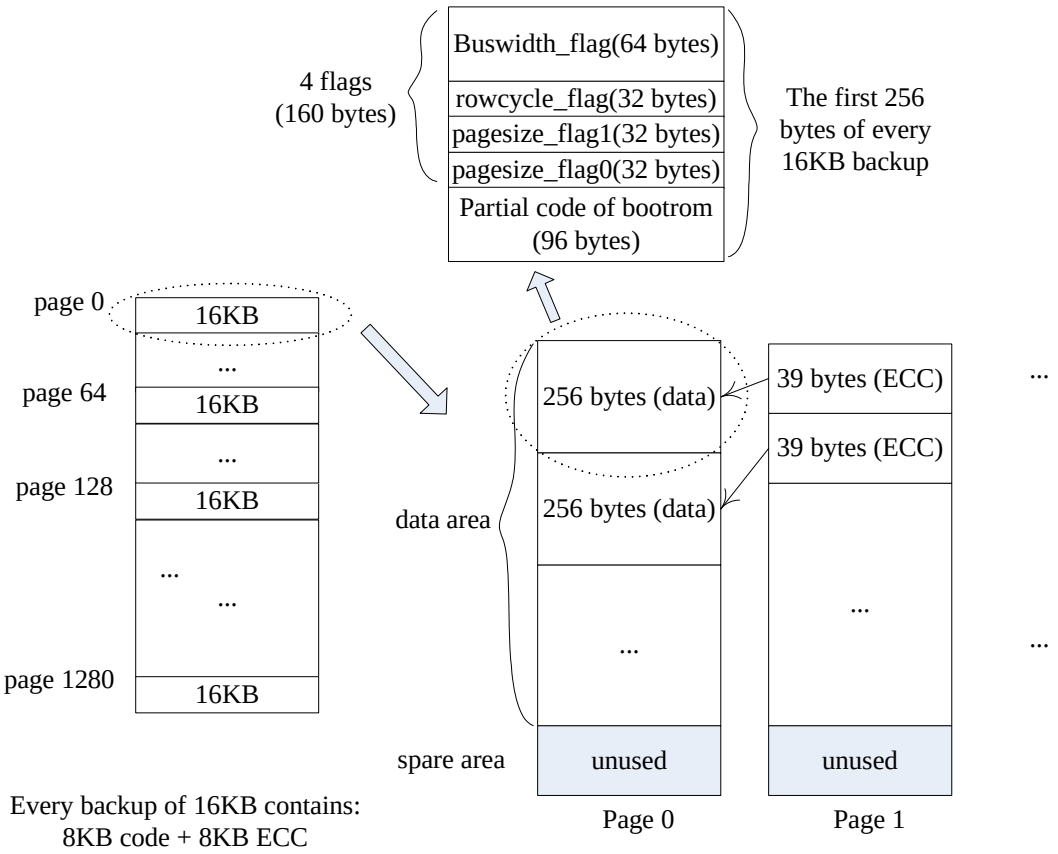

Figure 2 the distribution and structure of the boot code in NAND

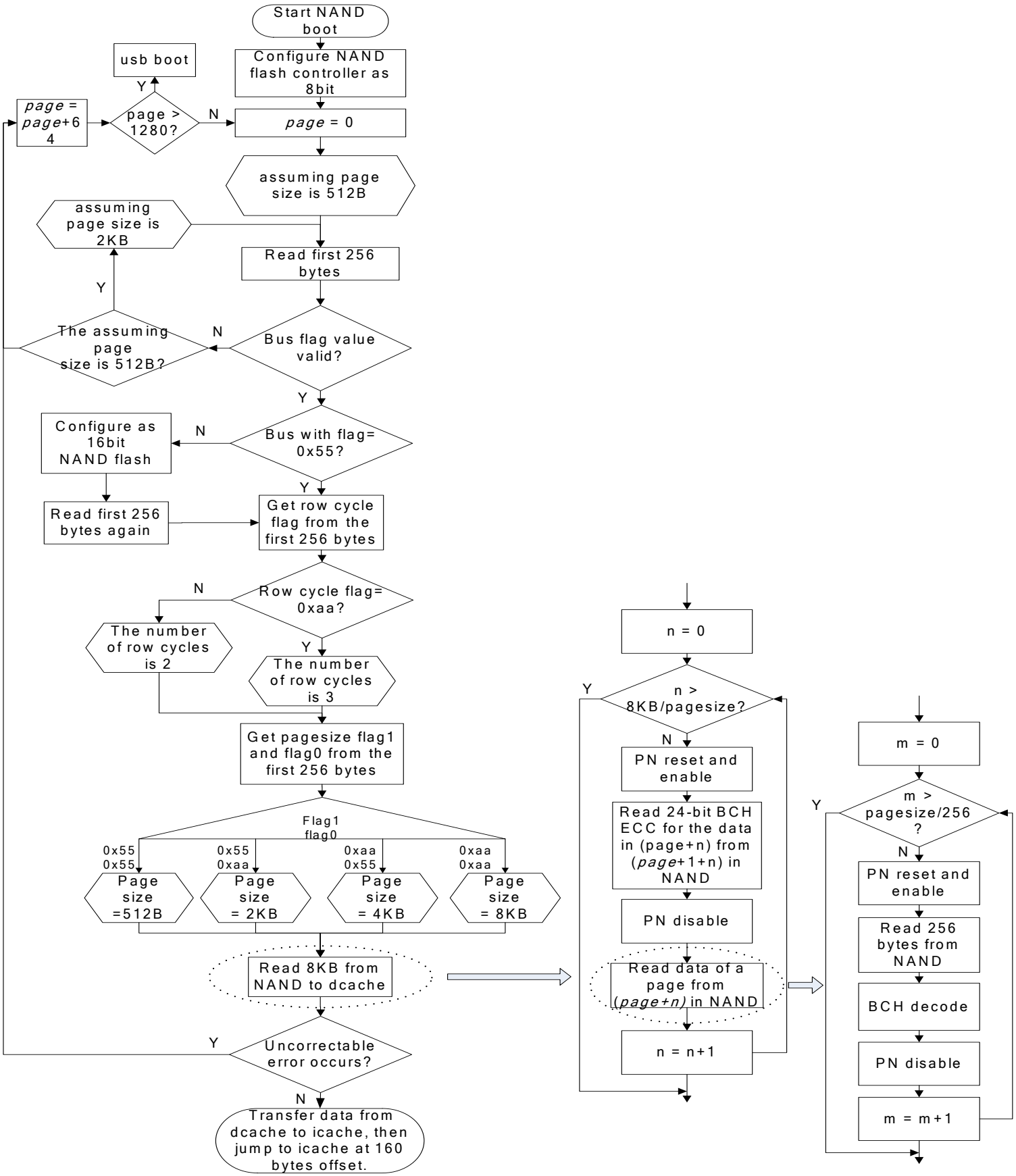

Figure 3 JZ4760 NAND Boot Procedure

1.4 USB Boot Specification

When boot_sel[2:0] is selected as USB boot, the internal boot ROM downloads user program from the USB port to internal SRAM and branches to the internal SRAM to execute the program.

JZ4760 support the external main crystal whose frequency is 12MHz, 13MHz, 19.2MHz or 26MHz.

The boot program supports both high-speed (480MHz) and full-speed (12MHz) transfer modes. The boot program uses the following two transfer types.

Table 3 Transfer Types Used by the Boot Program

Transfer Type	Description
Control Transfer	Used for transmitting standard requests and vendor requests.
Bulk Transfer	Used for responding to vendor requests and transmitting a user program.

The following figure shows an overview of the USB communication flow.

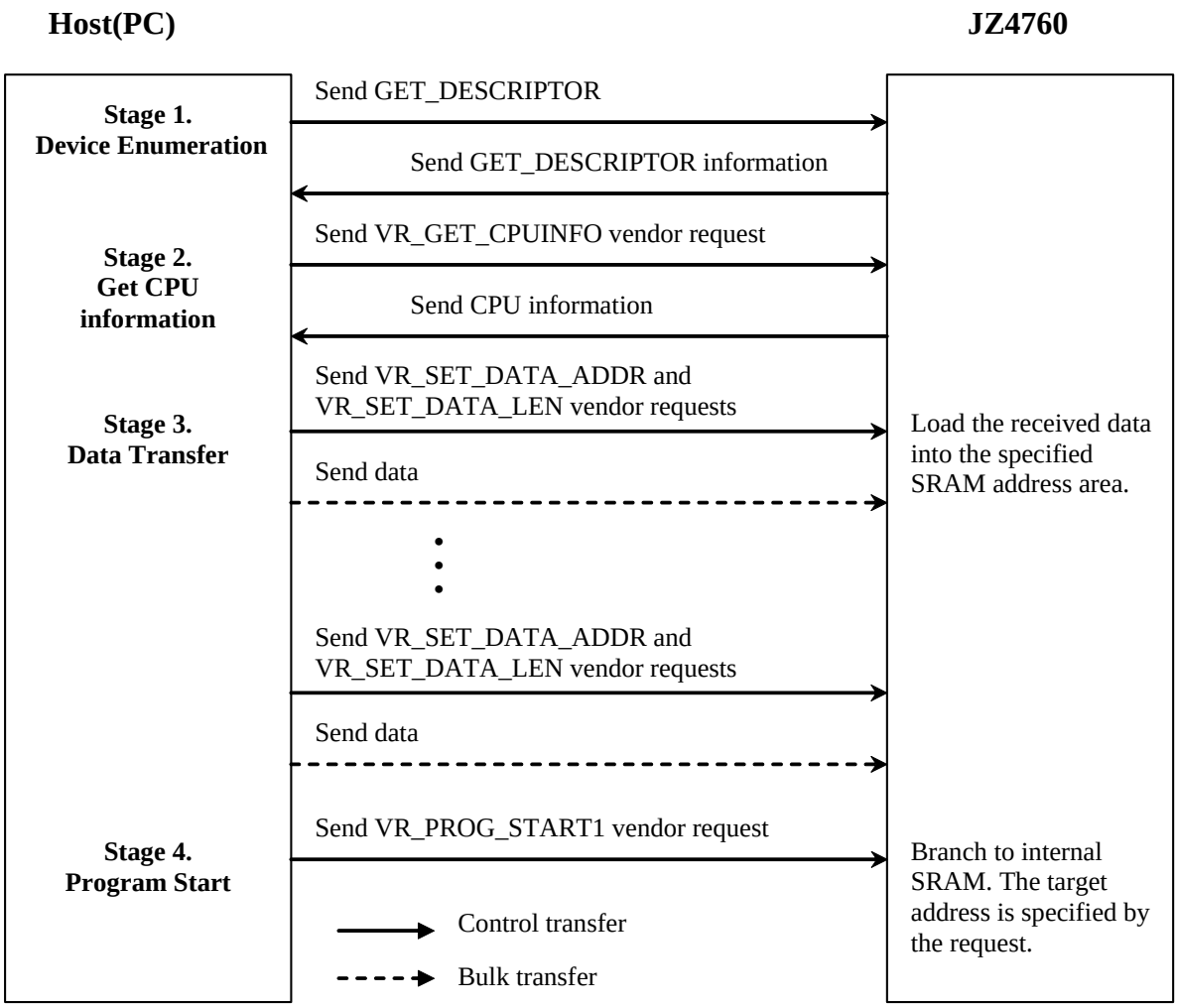

Figure 4 USB Communication Flow

The vendor ID and product ID for the USB boot device are 0x601A and 0x4760 respectively. The Configuration for USB is for Control Endpoint 0 with Max Packet Size equals 64 bytes, Bulk IN at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed, Bulk OUT at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed.

The USB boot program provides six vendor requests through control endpoint for user to download/upload data to/from device, and to branch to a target address to execute user program. The six vendor requests are VR_GET_CPU_INFO (0x00), VR_SET_DATA_ADDRESS (0x01), VR_SET_DATA_LENGTH (0x02), VR_FLUSH_CACHES (0x03), VR_PROGRAM_START1 (0x04) and VR_PROGRAM_START2 (0x05). User program is transferred through Bulk IN or Bulk OUT endpoint.

When JZ4760 is reset with boot_sel[2:0] equals 110b, 000b, 001b or 010b, the internal boot ROM

will switch to USB boot mode and wait for USB requests from host. After connecting the USB device port to host, host will recognize the connection of a USB device, and start device enumeration. After finishing the device enumeration, user can send VR_GET_CPU_INFO (0x00) to query the device CPU information. If user wants to download/upload a program to/from device, two vendor requests VR_SET_DATA_ADDRESS (0x01) and VR_SET_DATA_LENGTH (0x02) should be sent first to tell the device the address and length in byte of the subsequent transferring data. Then data can be transferred through bulk-out/bulk-in endpoint. After this first stage program has been transferred to device, user can send vendor request VR_PROGRAM_START1 (0x04) to let the CPU to execute the program. This first stage program must not greater than 16KB and is normally used to init GPIO and SDRAM of the target board. At the end of the first stage program, it can return back to the internal boot ROM by jumping to ra ($31) register. Thus user can download a new program to the SDRAM of the target board like the first stage, and send vendor request VR_FLUSH_CACHES (0x03) and VR_PROGRAM_START2 (0x05) to let the CPU to execute the new program. Next figure is the typical procedure of USB boot.

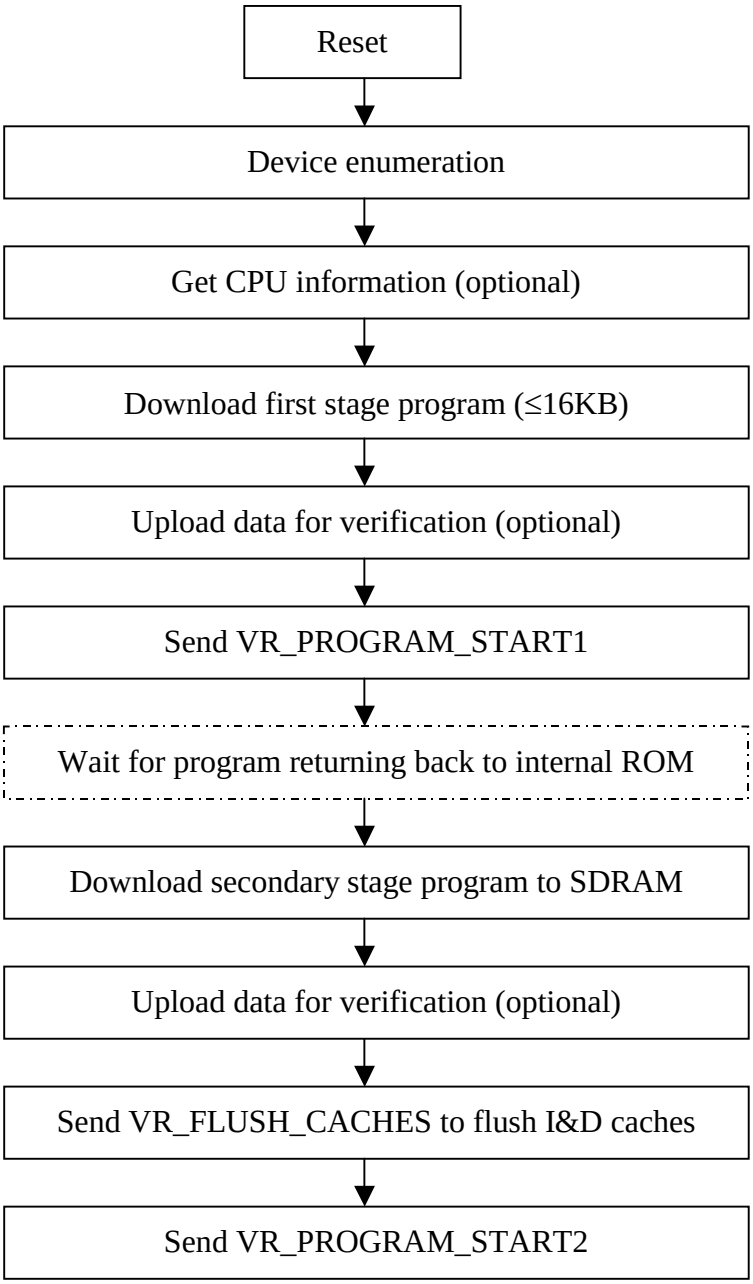

Figure 5 Typical Procedure of USB Boot

Following tables list all the vendor requests that USB boot program supports:

Table 4 Vendor Request 0 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	00H	VR_GET_CPU_INFO: get CPU information
2	wValue	2	0000H	Not in used
4	wIndex	2	0000H	Not in used
6	wLength	2	0008H	8 bytes

Table 5 Vendor Request 1 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	01H	VR_SET_DATA_ADDRESS: set address for next bulk-in/bulk-out transfer
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data address
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data address
6	wLength	2	0000H	Not in used

Table 6 Vendor Request 2 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	02H	VR_SET_DATA_LENGTH: set length in byte for next bulk-in/bulk-out transfer
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data length
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data length
6	wLength	2	0000H	Not in used

Table 7 Vendor Request 3 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	03H	VR_FLUSH_CACHES: flush I-Cache and D-Cache
2	wValue	2	0000H	Not in used
4	wIndex	2	0000H	Not in used
6	wLength	2	0000H	Not in used

Table 8 Vendor Request 4 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	04H	VR_PROGRAM_START1: transfer data from D-Cache to I-Cache and branch to address in I-Cache. Note: After downloading program from host to device for the first time, you can only use this request to start the program. Since the USB boot program will download data to D-Cache after reset. This request will transfer data from D-Cache to I-Cache and execute the program in I-Cache.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point
6	wLength	2	0000H	Not in used

Table 9 Vendor Request 5 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	05H	VR_PROGRAM_START2: branch to target address directly
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point
4	WIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point
6	WLength	2	0000H	Not in used

1.5 MMC/SD Boot Specification

If CPU boots from MSC0, the boot program will load 8KB code starting at sector 0 from MMC/SD card to cache. First the boot program initializes MSC0_D0, MSC0_CLK, MSC0_CMD as function pins. Only one data pin MSC0_D0 is used. Then the boot program sends CMD55 to test if it's SD or MMC card and initializes the card. At last it loads 8KB code from the card to cache and branches to execute the code in cache.

When initializing the card, the clock of EXTCLK/128 is used. And when reading data, the clock of EXTCLK/2 is used.

The procedure of the JZ4760 MMC/SD boot is shown as follow:

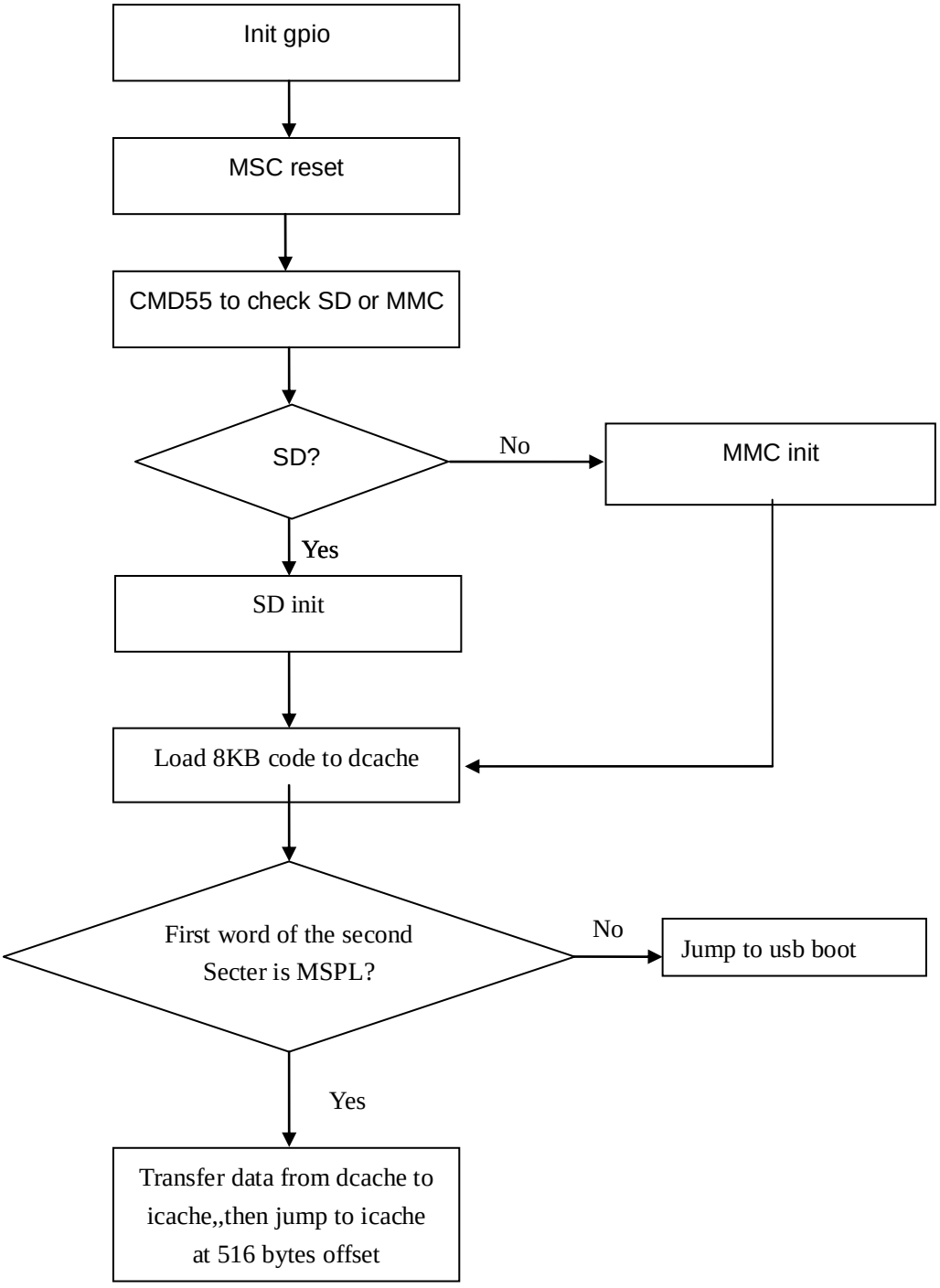

Figure 6 JZ4760 MMC/SD Boot Procedure