1052 Elwell Court, Palo Alto, CA 94303 Tel: 800.683.5943 www.zfmicro.com

The ZF int19handler Overview

The ZF int19handler™ and ZFx86 BIOS allows multiple option-ROM (also called extension-ROM) images to reside in the same Flash that contains the BIOS. Thus, all binary images that comprise an embedded application may use an economical single Flash solution.

This document describes the ZF int19handler built into ZFx86 Phoenix BIOS and its operation. The int19handler intercepts control just before the operating system boots from the disk and scans the option-ROMs in a predefined memory window. At this point in the boot process, the BIOS has fully initialized all the needed devices. By using some special option-ROMs identified at this stage, the ZFx86 launchs the operating system or user programs from the Flash memory.

How The ZF int19handler Works

The BIOS uses int19h vector after power on self test [POST] to load and execute any “bootstrap” code located on a diskette or the hard drive. The int19handler is a standard legacy option-ROM built into the BIOS image and visible during the BIOS run time. This in19handler intercepts the BIOS int19 command and scans for additional option-ROMs using the user-defined memory window chip selects.

Invoking int19h Operation

If the secondary scan finds legitimate option-ROM headers, it transfers control to that object which may now control the boot up process without returning to BIOS. If no “boot up” option-ROMs are found, control returns to the standard BIOS boot up process.

Thus, when the BIOS is ready to boot from disk, it invokes int19h, and the control transfers to the ZF int19handler. Initially, it restores the original BIOS’ int19h boot vector. So that if the int19handler finds no option-ROMs in the special user-defined region scan, it then jumps to the original boot vector address, and a normal disk boot occurs.

The ZF int19handler sets up the segment registers it uses (DS and ES) and also sets up the stack to its own designated memory region.

Setting The mem_cs0 Value

The Int19handler scans the memory according to the mem_cs0 settings defined in Phoenix BIOS Setup Utility’s *Advanced* > *Advanced Chipset Control* > *ISA Memory Chip Select* setup menu. To do this, the handler first reads the Flash PAGE, Window BASE, Window SIZE, and Full ISA values from ZF-Logic registers and saves read values to internal variables for later use and diagnostic printing.

The BIOS requires that the memory window mem_cs0 be set to these minimum values:

- Window BASE = E0000
- Flash PAGE = F00000
- Window SIZE = 128K

Scanning For Option-ROMs

If the int19handler sees only these values then no user scans are required, and the handler invokes the normal boot sequence. Otherwise, the int19handler enters the option-ROM scan loop. The value of ES determines the segment to be scanned.

Before calling the ROM-extension's code, the int19handler verifies that the option-ROM checksum is valid. The checksum algorithm uses the option-ROM's size which is the number of 512-byte blocks stored as a single byte address at OPROMSTART+2.

It calculates the checksum or every block, and stores the sum of all block checksums in the DH register.

- When all blocks are processed and the checksum in DH equals zero, then the handler knows that the option-ROM is good.
- When checksum does not equal zero, then the option-ROM execution is skipped and the search continues.

Prior to calling the option-ROM's code, the int19handler assigns a signature-value of "MORX" stored in the EDX register. Option-ROMs use this value to determine that they have been called by the handler.

When the scan has brought us to the value defined in the ScanLimit variable, then everything is done and the int19handler jumps to the original BIOS int19h address, and normal disk boot resumes. Note that the option-ROMs that contain their own Flash boot algorithms will not return to this point.

ZF Int19handler Flowchart

Figure 1 illustrates the ZF int19handler flow.

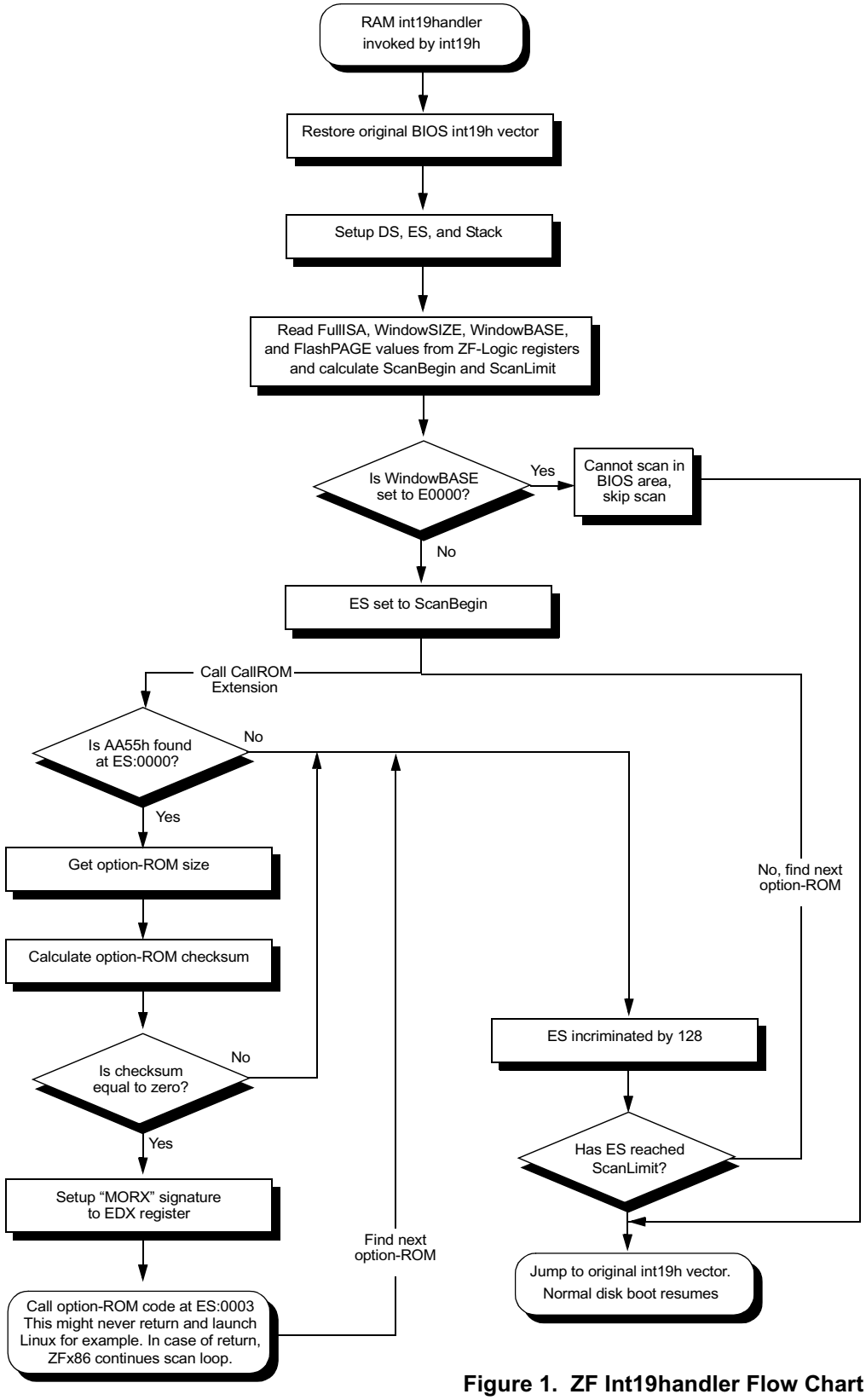

Figure 1. ZF Int19handler Flow Chart

Flash Multi-Boot Example

The int19handler delivers a flexible way of booting different operating-systems from the same Flash simply by changing the mem_cs0 settings thereby changing the Flash mapping to a memory window.

Figure 2 shows an example of a Z-tag Manager's Contents window where the 2Mb AMD Flash chip, which is commonly found on ZFx86 Integrated Development System (IDS), contains both the Linux kernel with the Linux Loader, and also the WindRiver vxWorks binary image and VxWorks loader.

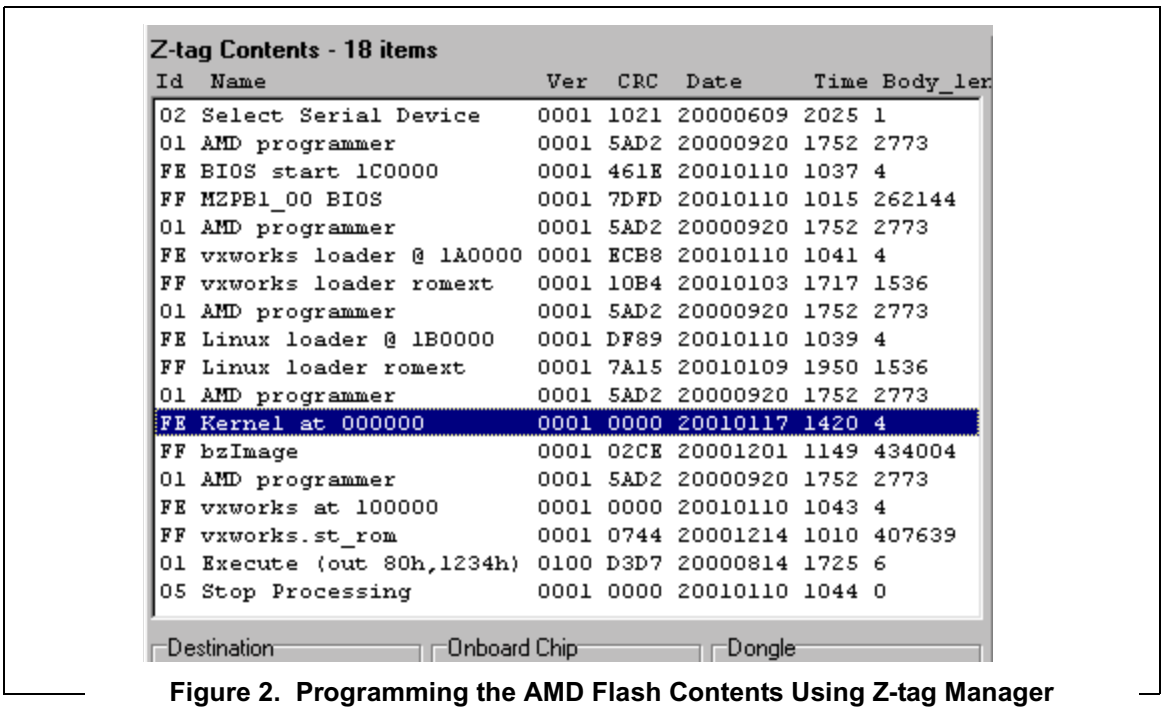

Figure 2. Programming the AMD Flash Contents Using Z-tag Manager

When the Z-tag Manager programs the Flash, it will contain the following items:

- ZFx86 Phoenix BIOS at flash offset 1C0000h
- VxWorks loader at flash offset 1A0000h
- Linux loader at flash offset 1B0000h
- Linux kernel at flash offset 000000h
- VxWorks image at flash offset 100000h

1. Write the Z-tag Manager contents to the Flash by setting up a physical connection between the IDS and the host computer using the dongle.
2. Select PassThrough mode and the appropriate dongle type, then press the Z-tag Manager's Write button.
3. Using the Phoenix BIOS setup utility's *Advanced* > *Advanced Chipset Control* > *ISA Memory Chip Select* setup menu, set the memory window mem_cs0 settings in the BIOS setup. The mem_cs0 setting then determines which operating system boots.

   For booting Linux, set the mem_cs0 memory window settings as follows:

   - Window BASE = C8h (corresponds to C8000)
   - Flash PAGE = E8h (maps Flash from 1B0000 to C8000 in low memory area)
   - Window SIZE = 1h

   Note: The kernel only is booted. The root file system is not found because INITRD image is not in the Flash.

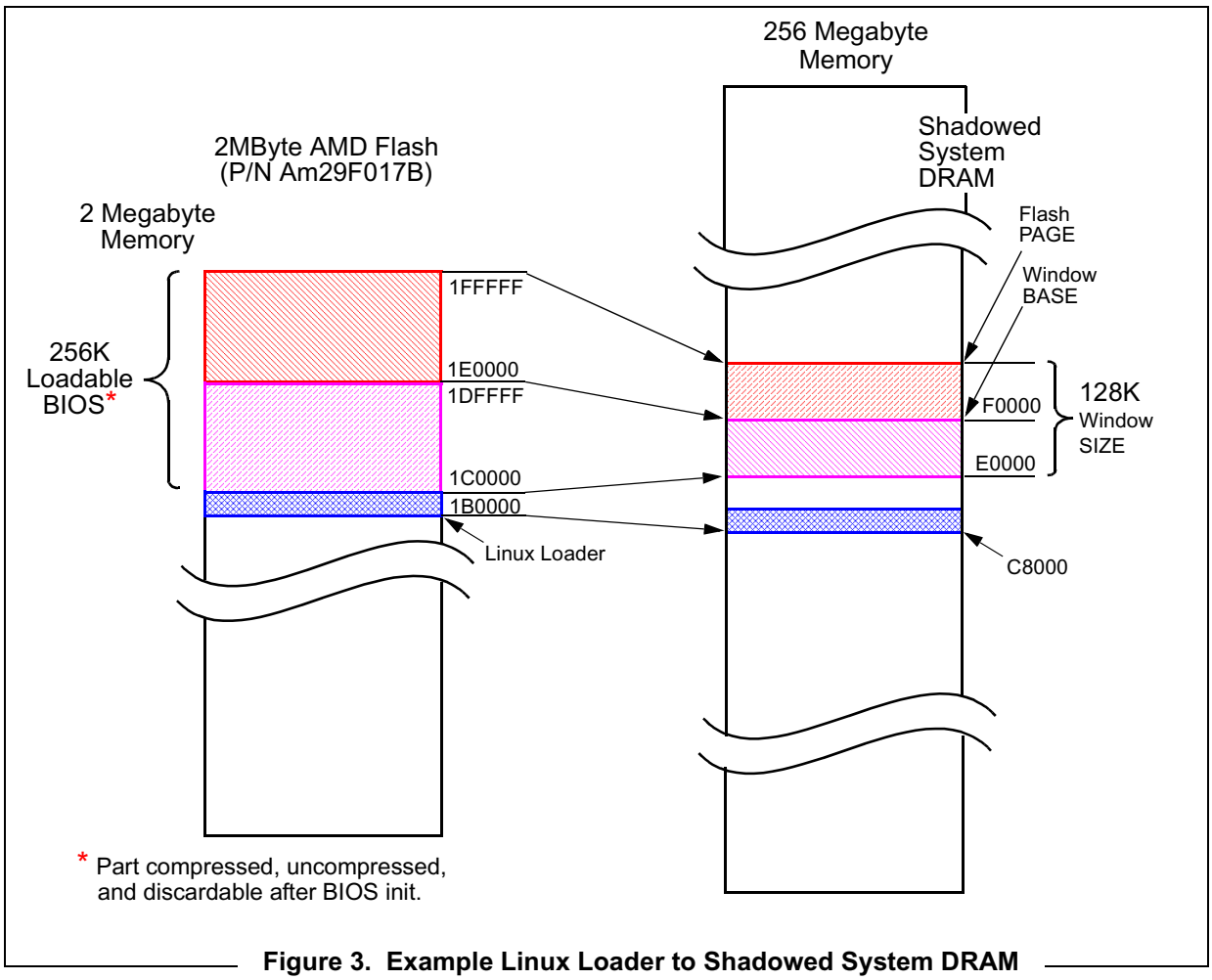

Figure 3. Example Linux Loader to Shadowed System DRAM

For booting VxWorks:

- Window BASE = C8h (corresponds to C8000)
- Flash PAGE = E8h(maps flash from 1A0000 to C8000 in low memory area)
- Window SIZE = 1h

The memory window Data-width value should be set to 8 bits in the previous examples.

Sample Loader Files Available

Figure 4 shows a debug example of the Linux Loader option-ROM's first 32 bytes. Note the "55 AA" signature in bytes 0 and 1. Also, byte 2 indicates a length value of 03 (3x512=1024). The first two instructions in this example move a 70h to register AL, and then outputs to port 80.

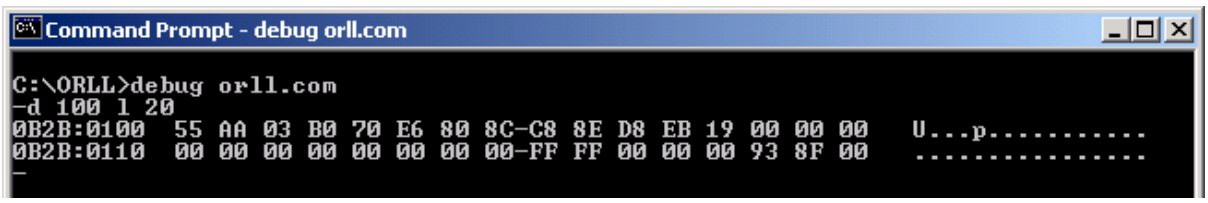

Figure 4. Debug ORLL.COM

Download sample source and build batch files from the ZF Micro Devices website: *http://www.zfmicro.com.* See the following zipped directories: LinuxLoaderOptionROM and VzWorksOptionROM. The *.bat batch files contain the necessary build instructions. View the *.asm files for specific loader examples.

Conclusion

Use the ZF int19handler to boot multiple user applications, or operating systems, or option-ROMs from the same Flash. Make the selection of which item to run by changing the BIOS settings using the Phoenix BIOS setup utility's *Advanced* > *Advanced Chipset Control* > *ISA Memory Chip Select* option.

The int19handler also allows you to run multiple items in series – accomplished when multiple items (for example, option-ROM format programs in the Flash) fit inside one defined memory window (placed into the Flash in specific order). The first item that executes is mapped to the lowest memory address allowing engineers to create a specific sequence of initialization events.