Booting Linux From Flash

Find the following items in the compressed “LinuxFileSet.zip” directory:

- “Ready for Z-tag Manager” directory containing the following
  - INITRD
  - KERNEL
  - LinuxLoader
  - LinuxFromFlash.bin
- 9150-0017-00_Booting Linux from Flash.pdf
- Readme.txt
- AM29xxx.rom
- zdisk-1.87.tar.gz when uncompressed contains the following items:
  - busybox directory
  - doc directory
  - genext2fs directory
  - loadlin directory
  - mklibs directory
  - syslinux directory
  - README.txt
  - zdisk
  - rescue.tgz directory that when uncompressed contains the following directories:
    - bin
    - boot
    - dev
    - etc
    - lib
    - mnt
    - proc
    - root
    - sbin
    - tmp
    - usr
    - var

Using The ZF Linux Loader

This document provides a complete working example of a small Linux system booted entirely from Flash on the ZFx86 Integrated Development System. The Development System has a 2MByte Flash chip that contains the System BIOS in the uppermost 256K RAM. Use the remainder of the chip to store user programs, data, or an Operating System. In this example, we use it to hold all of the elements comprising a small Linux system.

The example OS system used was originally designed to fit on a single 1.44 Mbyte floppy diskette. This matches our available size well, and demonstrates how you can create and package a compressed file system usable by the Linux Loader.

How It Works

As a normal boot process, the Phoenix BIOS scans the system for Option-ROMs. The ZFx86 implements the Linux Loader as one of these Option-ROMs, and gives it control just before the BIOS normally turns control over to the first bootable device.

The Linux Loader performs three relatively simple tasks:

- Copies a Linux kernel from Flash into RAM memory
- Copies a compressed file system from Flash into RAM memory
- Transfers control to the kernel it previously copied into RAM

In order to accomplish these tasks, the various components must be located in specific places within the Flash device. This will impose some restrictions on the size of both the kernel and the compressed file system, but they should be adequate for most situations.

Several helpful documents about Booting From Flash are available with most Linux distributions (for example, How-To and Info files). See the *Bootdisk-HOWTO* for information on creating an Initial RAM Disk (initrd).

Creating the Components

You must create two major components when implementing a booted-by-Flash Linux system:

- The Kernel
- The Compressed File System

The Kernel

Many documents and books exist that describe the process of building or compiling your own kernel. In general, very few differences exist between the “normal” cases described in those texts, and what we need in our kernel that will boot from Flash.

In our case, the kernel requires the ability to use RAM Disks, the ability to use an Initial RAM Disk (initrd) built into the kernel, and the kernel size must not exceed 524288 (80000h) bytes. Use the file generated by the “make zImage” (or “make bzImage”) without a modification. Use the pre-built kernel supplied with this example, named “Kernel”, located in the “Ready for Z-tag Manager” directory.

The Compressed File System

Our example uses a general purpose Rescue Disk as the usable content of the system. It provides many familiar commands within the context of file manipulation, editing, system administration, and so on. While each custom application contains different content, the techniques used to put the File System into the Flash should be essentially the same for all cases.

For the purpose of our example, you need not create the File System, but rather take advantage of one already created and found in the compressed file. The Rescue Disk used here is part of “zdisk-1.87” created by Kent Robotti.

1. Install the zdisk package onto an existing Linux system.
2. Create a directory (location of your choice) into which you extract the File System components.
3. Extract the File System into the new directory (tar xzvpf rescue.tgz -C Rescue), assuming the new directory “Rescue” is located at the same level as “rescue.tgz”.

If you create your own File System, model it after the contents of the Rescue directory. For example, create directories for /dev, /proc, /bin, and so on. The remainder of this procedure applies equally well to this example or a custom system of your own design.

Creating a Device For The File System

We need our File System to occupy an actual device (at least temporarily) so that we can compress it for use as an initrd. You could use several techniques to do this, including a spare Hard Disk partition, a Loop-Back device (treating a disk file as a device), and using a RAM Disk. In our example, we use a RAM Disk.

Since we intend to create a *Compressed* File System, the device should contain only zeros before we populate it. This will allow the compression step to have the greatest effect.

1. To zero out a 4 Mbyte RAM Disk (more than enough space for our needs) type the following:

   ```
   dd if=/dev/zero of=/dev/ram0 bs=1k count=4096
   ```

2. To create the file system (format) on the device, type the following:

   ```
   mke2fs -m 0 /dev/ram0 4096
   ```

   The `-m 0` switch prevents reserving space for the SuperUser.

3. To mount the RAM Disk and populate it, type the following two commands:

   ```
   mount /dev/ram0 /mnt

   cp -a Rescue/* /mnt
   ```

   Be sure to use the -a switch when copying the files into the RAM Disk, as this preserves the special “devices” and so on.

4. Once the RAM Disk is fully populated, you must unmount the device. Note that this is your last chance to make any changes. Type the following:

   ```
   umount /mnt
   ```

5. Then copy the device contents to a single file, and compress it. Type the following:

   ```
   dd if=/dev/ram0 bs=1k count=4096 | gzip -v9 > rootfs.gz
   ```

   The zipped file rootfs.gz contains the contents of our complete File System in a compressed form.

Creating the “initrd” Header

For the Linux Loader to copy the compressed File System into RAM, it must know the file size. The 4 byte Header contains the size information and must be added to the file system. Use whatever technique you are familiar with to create the header.

During development, when the File System contents change often, generating the Header using the following procedure makes it a simple process. However, it might be worthwhile to write a small program to generate this Header, and combine it with the compressed File System. For our example, because we need only do it once, we complete all of the steps manually.

1. First, we need to know the size (in bytes) of the rootfs.gz file. Obtain this information by issuing a “long form” directory listing. Type the following:

   ```
   ls -l
   ```

The system displays the long form directory listing:

```
[~/Zdisk/zdisk-1.87]# ls -l
total 2112
-rw-rw-r--    1 root     root         5169 Nov 15 05:21 README
drwxr-xr-x   14 root     root          267 Mar 30 16:19 Rescue
drwxr-xr-x    2 root     root           81 Oct 30 18:16 busybox
drwxrwxr-x    2 root     root           91 Nov 15 04:36 doc
drwxr-xr-x    2 root     root           89 Aug 30  2000 genext2fs
drwxr-xr-x    2 root     root          115 Aug 16  2000 loadlin
drwxr-xr-x    2 root     root           82 Oct 30 01:19 mklibs
-rw-r--r--    1 root     root       706944 Nov 15 07:42 rescue.tgz
-rw-r--r--    1 root     root       709053 Apr  6 10:15 rootfs.gz
drwxr-xr-x    2 root     root          224 Nov 12 23:21 syslinux
-rwxr-xr-x     1 root      root         17852 Nov 15 07:38 zdisk
```

2. Next, we need a four-byte file in which we place the size information. One way to do this is to create the four-byte file. Type:

   ```
   dd if=/dev/zero of=header bs=1 count=4
   ```

3. Edit this file using a Hex Editor.

   In an x86 processor, the least significant bytes appear at lower addresses, so you need to rearrange the order of the bytes for the size of our file. Converting the size of rootfs.gz in the listing above to hex, we get the following:

   ```
   709053 (base 10) = ad1bd (base 16)
   ```

4. In the Hex Editor, reorder the bytes, and add extra zeros to fill up four bytes:

   ```
   0000:0000        bd d1 0a 00
   ```

5. Save the file with the file name “header”.

6. Now, combine the header file and the compressed File System (rootfs.gz) so that the Linux Loader can use it:

   ```
   cat header rootfs.gz > initrd
   ```

   This combined file, named initrd, is installed into the Flash chip.

Putting the Pieces Together

You have completed the development steps that must be performed in the Linux environment. Perform the remaining steps on a machine running Microsoft Windows™ OS, because the Z-tag Manager software runs on that OS. Copy the two files “kernel” and “initrd” onto media that your Windows machine can access.

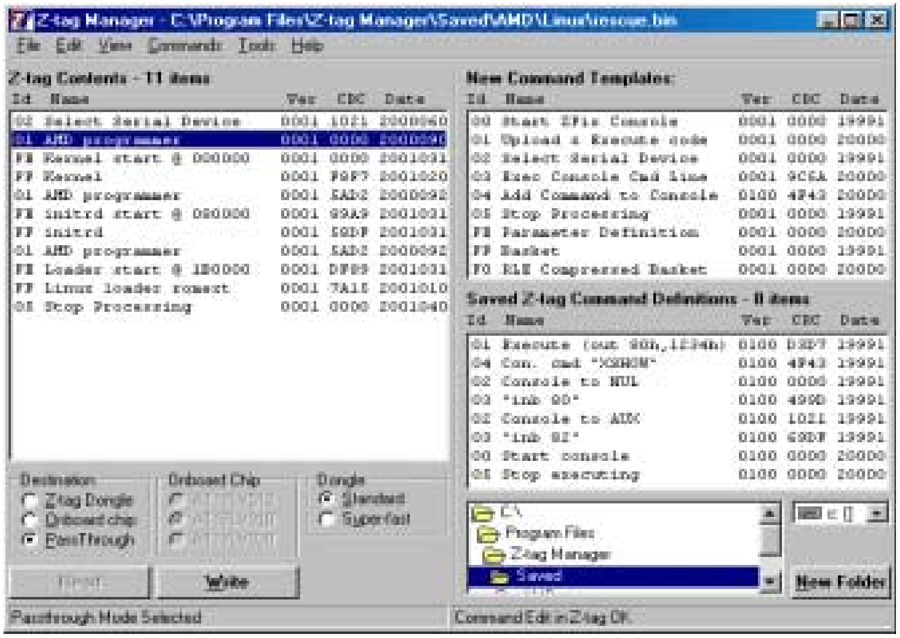
Creating the Dongle Image

Using the Z-tag Manager software (and the Dongle), load the three files into the Integrated Development System's AMD Flash memory device. The following three file are required: the "kernel", the "initrd", and the Linux Loader.

1. Each file requires three Z-tag instructions to complete the Flash upload-process. Use these instructions:

   - **01** – Upload and Execute Code
     This instruction contains the code that writes data into the AMD Flash chip. Locate the file AM29Fxxx.rom in the accompanying file set, and use it to load each of the three files.
   - **FE** – Parameter Definition
     This instruction sets the parameter starting address within the Flash device. The Z-tag Manager loads the file at whatever offset specified using this instruction.
   - **FF** – Basket
     This contains the actual data loaded into the Flash device.

Figure 1 shows how the instruction list appears in the Z-tag Manager window.


Z-tag Manager - C:\Program Files\Z-tag Manager\Saved\AMD\Linux\rescue.bin
File Edit View Commands Tools Help
C:\
Program Files
Z-tag Manager
Saved
Reset
Write
New Folder

Z-tag Contents - 11 items

Id	Name	Ver	CRC	Date
02	Select Serial Device	0001	1021	2000060
01	AMD programmer	0001	0000	2000090
FE	Kernel start @ 000000	0001	0000	2001031
FF	Kernel	0001	F9F7	2001020
01	AMD programmer	0001	5AD2	2000092
FE	initrd start @ 080000	0001	89A9	2001031
FF	initrd	0001	59DF	2001031
01	AMD programmer	0001	5AD2	2000092
FE	Loader start @ 180000	0001	DF89	2001031
FF	Linux loader romext	0001	7A15	2001010
05	Stop Processing	0001	0000	2001040

New Command Templates:

Id	Name	Ver	CRC	Date
00	Start ZFix Console	0001	0000	19991
01	Upload & Execute code	0001	0000	20000
02	Select Serial Device	0001	0000	19991
03	Exec Console Cmd Line	0001	9C5A	20000
04	Add Command to Console	0100	4F43	20000
05	Stop Processing	0001	0000	19991
FE	Parameter Definition	0001	0000	20000
FF	Basket	0001	0000	19991
F0	RLE Compressed Basket	0001	0000	20000

Saved Z-tag Command Definitions - 8 items

Id	Name	Ver	CRC	Date
01	Execute (out 80h,1234h)	0100	D3D7	19991
04	Con. cmd "XONON"	0100	4F43	19991
02	Console to NUL	0100	0000	19991
03	"inb 80"	0100	490D	19991
02	Console to AUX	0100	1021	19991
03	"inb 8I"	0100	69DF	19991
00	Start console	0100	0000	20000
05	Stop executing	0100	0000	20000

Destination
○ Z-tag Dongle
○ Onboard chip
● PassThrough

Onboard Chip

Dongle
● Standard
○ Superfast

Passthrough Mode Selected

Command Edit in Z-tag OK

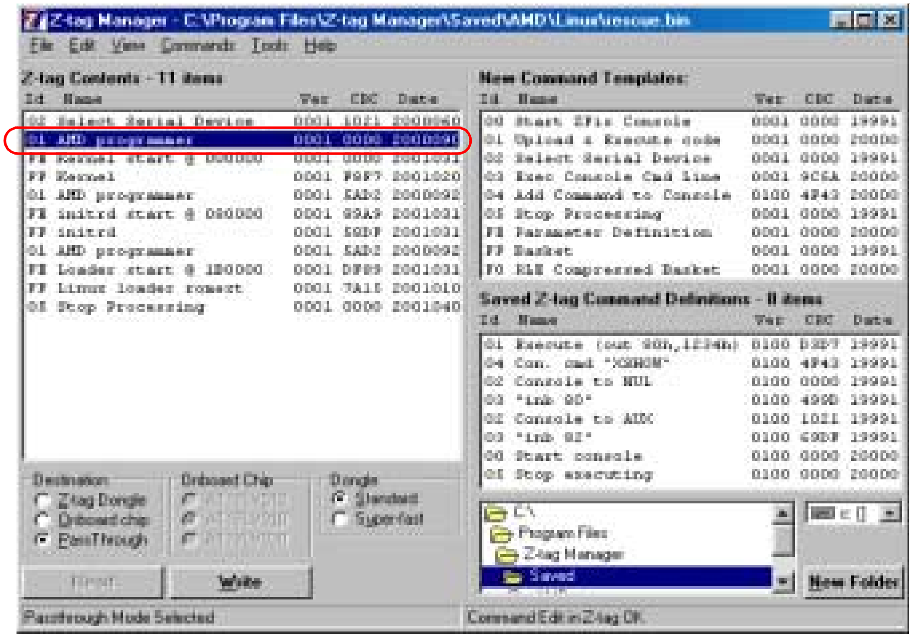
Figure 1. Z-Tag Manager Instruction List

2. *Option*: In addition to the instructions needed to copy files into the Flash device, use the **02** – Select Serial Device instruction to send status messages to the Serial Port. However, you may remove this instruction if the diagnostic or status messages are not required.

3. At the end of the instruction list, use the **05** – Stop Processing instruction to inform the Z-tag Manager that all instructions are complete and to not wait for any more.

Double click on any instruction listed in the T-tag Contents window to open an editing dialog box where you can modify the label and any command parameters.

4. Double click the **01** – Upload and Execute code instruction in the Z-tag Contents window to edit its parameters. See Figure 2. Use this command repeatedly for all three files.



Z-tag Contents - 11 items

Id	Name	Ver	CRC	Date
02	Select Serial Device	0001	1021	2000060
01	AMD programmer	0001	0000	2000090
FE	Kernel start @ 000000	0001	0000	2001031
FF	Kernel	0001	F9F7	2001020
01	AMD programmer	0001	5AD2	2000092
FE	initrd start @ 080000	0001	89A9	2001031
FF	initrd	0001	58DF	2001031
01	AMD programmer	0001	5AD2	2000092
FE	Loader start @ 1E0000	0001	DF89	2001031
FF	Linux loader romext	0001	7A15	2001010
05	Stop Processing	0001	0000	2001040

New Command Templates:

Id	Name	Ver	CRC	Date
00	Start ZFix Console	0001	0000	19991
01	Upload & Execute code	0001	0000	20000
02	Select Serial Device	0001	0000	19991
03	Exec Console Cmd Line	0001	9C6A	20000
04	Add Command to Console	0100	4F43	20000
05	Stop Processing	0001	0000	19991
FE	Parameter Definition	0001	0000	20000
FF	Basket	0001	0000	19991
F0	RLE Compressed Basket	0001	0000	20000

Saved Z-tag Command Definitions - 8 items

Id	Name	Ver	CRC	Date
01	Execute (out 80h,1234h)	0100	D3D7	19991
04	Con. cmd "XGROM"	0100	4F43	19991
02	Console to NUL	0100	0000	19991
03	"inb 80"	0100	490D	19991
02	Console to AUX	0100	1021	19991
03	"inb 82"	0100	69DF	19991
00	Start console	0100	0000	20000
05	Stop executing	0100	0000	20000

Destination
○ Z-tag Dongle
○ Onboard chip
● PassThrough

Onboard Chip
○ [illegible]
● [illegible]
○ [illegible]

Dongle
● Standard
○ Superfast

Passthrough Mode Selected

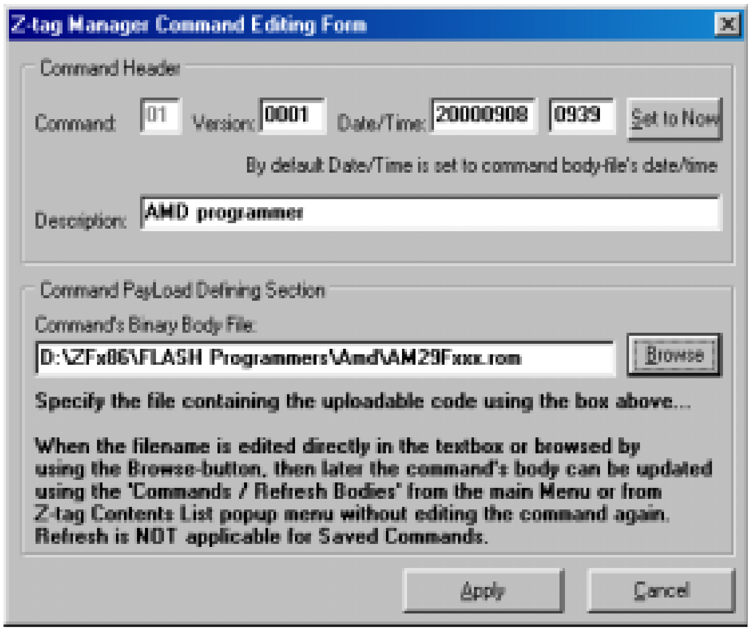
Command Edit in Z-tag OK.

Figure 2. Z-tag Manager Main Menu

5. Type the description “AMD programmer” in the text box.

6. Use the Browse button to modify the Command’s Binary Body File: path so that it points to the correct location on your system for the AM29Fxxx.rom file. The Z-tag Manager uses the **01** routine to write the data into AMD Flash devices. See Figure 3.

7. Click Apply to save your changes.



Z-tag Manager Command Editing Form

Command Header

Command: 01 Version: 0001 Date/Time: 20000908 0939

By default Date/Time is set to command body-file's date/time

Description: AMD programmer

Command PayLoad Defining Section

Command's Binary Body File:

D:\ZFx86\FLASH Programmers\Amd\AM29Fxxx.rom

Specify the file containing the uploadable code using the box above...

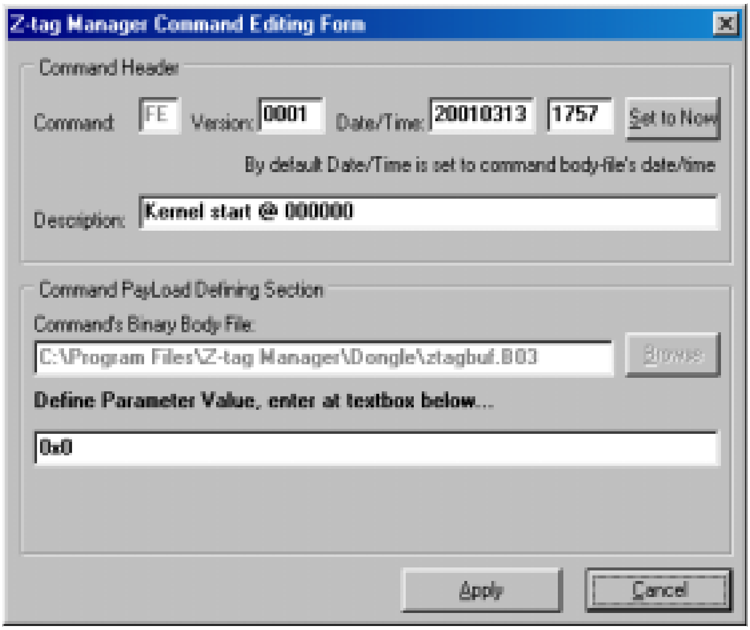
When the filename is edited directly in the textbox or browsed by using the Browse-button, then later the command's body can be updated using the 'Commands / Refresh Bodies' from the main Menu or from Z-tag Contents List popup menu without editing the command again. Refresh is NOT applicable for Saved Commands.

Figure 3. 01 – Upload and Execute Code Form For Flash

The Kernel

Figures 4 and 5 show the other two instructions associated with the Kernel file.

1. From the Z-tag Manager’s Main menu, choose the **FE** – Parameter Definition instruction and specify the location to place the kernel. See Figure 4.



Z-tag Manager Command Editing Form

Command Header

Command: FE Version: 0001 Date/Time: 20010313 1757

By default Date/Time is set to command body-file's date/time

Description: Kernel start @ 000000

Command PayLoad Defining Section

Command's Binary Body File:

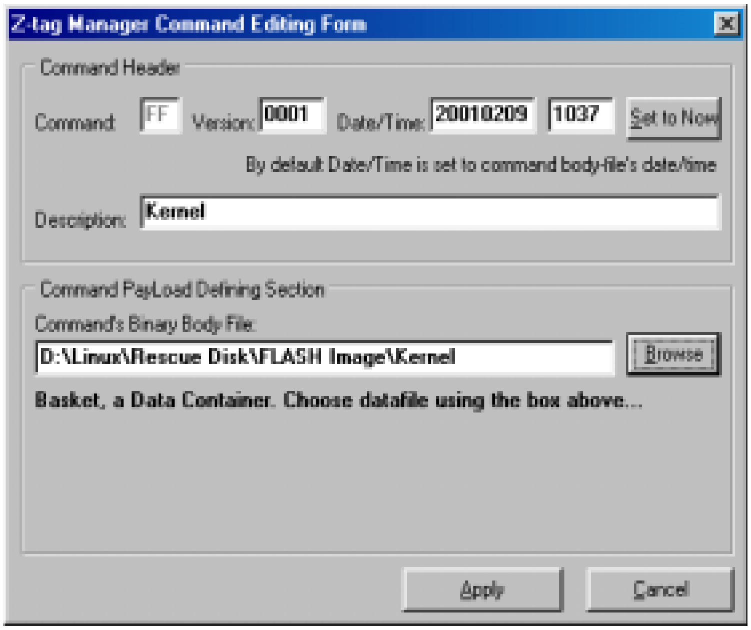
C:\Program Files\Z-tag Manager\Dongle\ztagbuf.B03

Define Parameter Value, enter at textbox below...

0x0

Figure 4. FE – Parameter Definition Editor Form For The Kernel

2. Type “Kernel start @ 000000” in the Description field.
3. Type “0x0” in the Define Parameter Value text field. The Linux Loader expects the kernel to start at the beginning of the Flash device (offset 0).
4. Click Apply to save your changes.
5. From the Z-tag Manager’s Main menu, use the **FF** – Basket instruction to specify the datafile’s location.
6. Use the Browse button to modify the Command’s Binary Body File: path so that it points to the correct system location of the kernel.


Z-tag Manager Command Editing Form
Command Header
Set to Now
Command PayLoad Defining Section
Browse
Apply
Cancel

Command: FF Version: 0001 Date/Time: 20010209 1037

By default Date/Time is set to command body-file's date/time

Description: Kernel

Command's Binary Body File:

D:\Linux\Rescue Disk\FLASH Image\Kernel

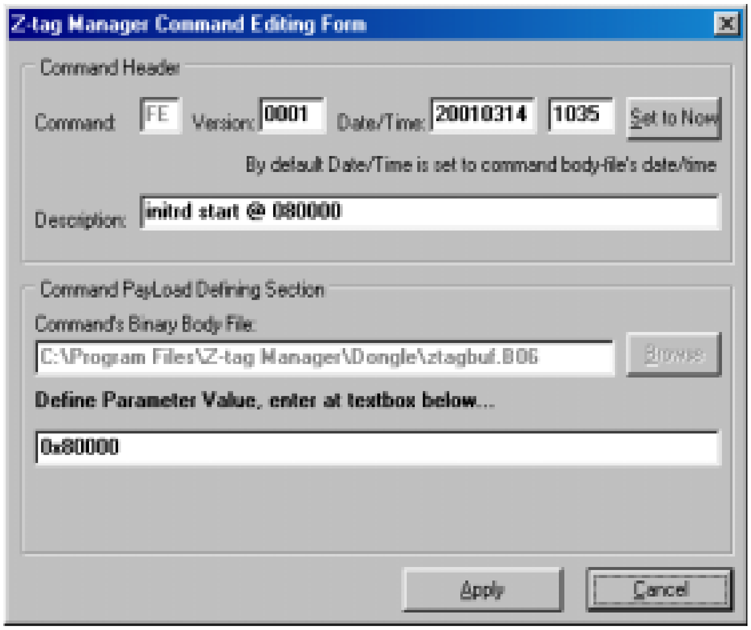
Basket, a Data Container. Choose datafile using the box above...

Figure 5. FF – Basket Editing Form For The Kernel

initrd File

Figures 6 and 7 show the other two instructions associated with the “initrd” file.

1. From the Z-tag Manager’s Main menu, choose the **FE** – Parameter Definition instruction to specify the location to place the initrd file. See Figure 6.



Z-tag Manager Command Editing Form

Command Header

Command: FE Version: 0001 Date/Time: 20010314 1035

By default Date/Time is set to command body-file's date/time

Description: initrd start @ 080000

Command PayLoad Defining Section

Command's Binary Body File:

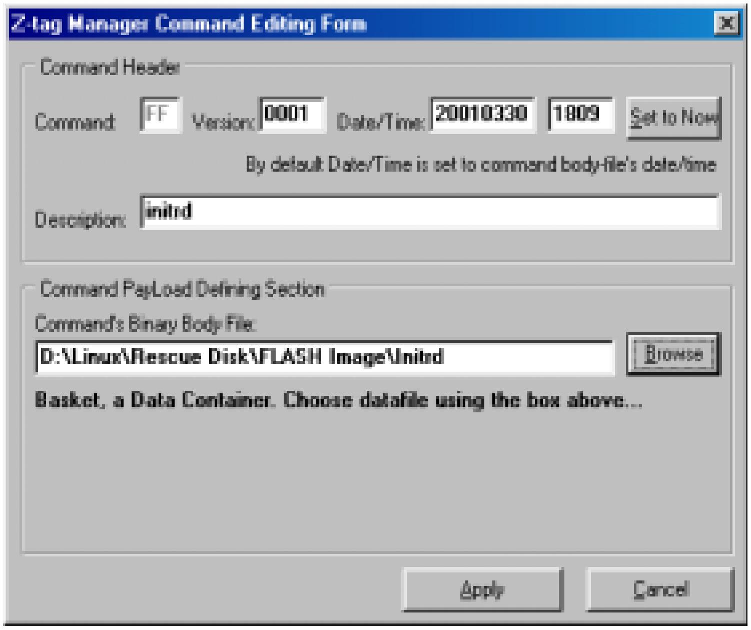
C:\Program Files\Z-tag Manager\Dongle\ztagbuf.B06

Define Parameter Value, enter at textbox below...

0x80000

Figure 6. FE – Parameter Definition Editor Form For initrd

2. Type “initrd start @ 080000” in the Description field.
3. Type “0x80000” in the Define Parameter Value text field. The Linux Loader expects the initrd to start at offset 0x80000 within the Flash device.
4. Click Apply to save your changes.
5. From the Z-tag Manager’s Main menu, use the **FF** – Basket instruction to specify the datafile’s location. See Figure 7.
6. Type “initrd” in the Description field.
7. Use the Browse button to modify the Command’s Binary Body File: path so that it points to the correct system location of the initrd.



Z-tag Manager Command Editing Form

Command Header

Command: FF Version: 0001 Date/Time: 20010330 1809

By default Date/Time is set to command body-file's date/time

Description: initrd

Command PayLoad Defining Section

Command's Binary Body File:

D:\Linux\Rescue Disk\FLASH Image\initrd

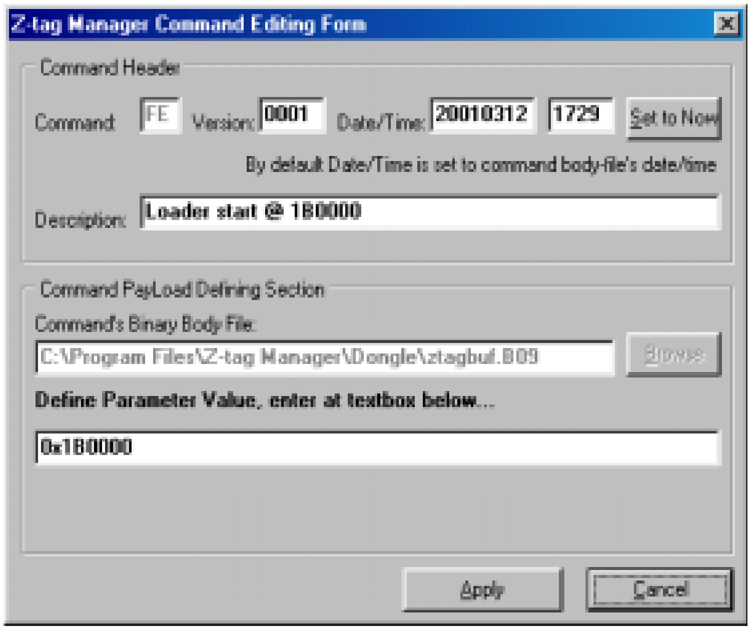
Basket, a Data Container. Choose datafile using the box above...

Figure 7. FF – Basket Editing Form For initrd

Linux Loader

Figures 8 and 9 detail the other two instructions associated with the Linux Loader file.

1. From the Z-tag Manager's Main menu, choose the **FE** – Parameter Definition instruction to specify the location to place the initrd file. See Figure 8.



Z-tag Manager Command Editing Form

Command Header

Command: FE Version: 0001 Date/Time: 20010312 1729

By default Date/Time is set to command body-file's date/time

Description: Loader start @ 180000

Command PayLoad Defining Section

Command's Binary Body File:

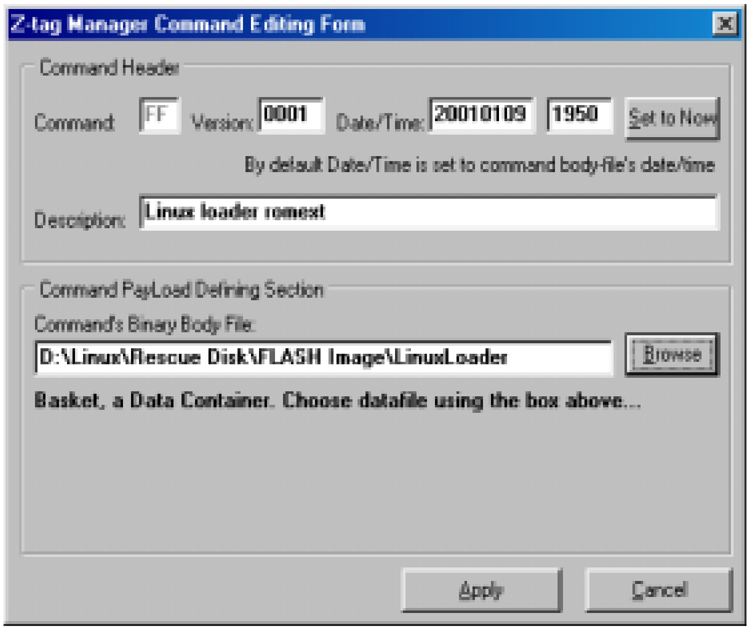
C:\Program Files\Z-tag Manager\Dongle\ztagbuf.B09

Define Parameter Value, enter at textbox below...

0x180000

Figure 8. FE – Parameter Definition Editor Form For Linux Loader

2. Type “Loader start @ 1B0000” in the Description field.
3. Type “0x1B0000” in the Define Parameter Value text field. Locate the Linux Loader at offset 0x1B0000 within the Flash device.
4. Click Apply to save your changes.
5. From the Z-tag Manager’s Main menu, use the **FF** – Basket instruction to specify the Linux Loader’s location. See Figure 9.
6. Type “Linux loader romext” in the Description field.
7. Use the Browse button to modify the Command’s Binary Body File: path so that it points to the correct system location of the Linux Loader.


Z-tag Manager Command Editing Form
Command Header
Command Payload Defining Section
Set to Now
Browse
Apply
Cancel

Command: FF Version: 0001 Date/Time: 20010109 1950

By default Date/Time is set to command body-file's date/time

Description: Linux loader romext

Command's Binary Body File: D:\Linux\Rescue Disk\FLASH Image\LinuxLoader

Basket, a Data Container. Choose datafile using the box above...

Figure 9. FF – Basket Editing Form For The Linux Loader

Writing The Data To Flash

Now that all the Z-tag Manager instructions are loaded and the various parameters are set correctly, use the following procedure to write the data into the Flash device.

1. In the Z-tag Manager's Main menu, set the "Destination" control to "Pass Through".

   The list of instructions previously created contains too much data to fit into the Dongle; therefore, you must connect the system running the Z-tag Manager to the IDS using the Dongle and a parallel extension cable. This cable should be wired "straight through" (do not use a standard printer cable).

2. Configure the Dongle for Pass Through mode by moving the JP2 jumper on the Dongle to pins 2-3.
3. With the two machines connected by the parallel cable and Dongle, click the "Write" button on the Z-tag Manager's Main menu.
4. Turn the IDS board's power ON (or press RESET, if power is already applied).

   A progress bar appears on the Z-tag Machine showing the transfer process.

5. Watch the LEDs on the Dongle that indicate that the transfer is complete. While the transfer processes, the "Status" LED blinks Yellow. If the transfer completes successfully, the "Status" LED turns Green.
6. Remove the Dongle from the IDS, and press the RESET button.

Setting the Memory Chip Select Window

To set the Memory Chip Select setting, follow the procedure below:

1. When the IDS boots the first time after loading the Flash contents, press the "F2" key to enter the PhoenixBIOS Setup program.
2. Select to the "Advanced" submenu.
3. Select the "Advanced Chipset Control" menu item.
4. Select the "ISA Memory Chip Select Setup" menu item.
5. Set the entries for "Memory Window - mem_cs0" as follows:
   - Window Size = 1h
   - Window Base = D8h
   - Window Page = D8h

   Note:The data width must be set to 8 bits for AMD Flash.

6. Exit and save the changes to the BIOS.

Using the new Linux Loader

When the system boots up now, the Linux Loader will be the active OS.

- To allow the Linux OS located in the Flash device to launch, simply do nothing when the Linux Loader message displays.
- If you wish to allow the normal system boot sequence to occur, press the “Esc” key when the Linux Loader message displayes, and the normal boot sequence launches.