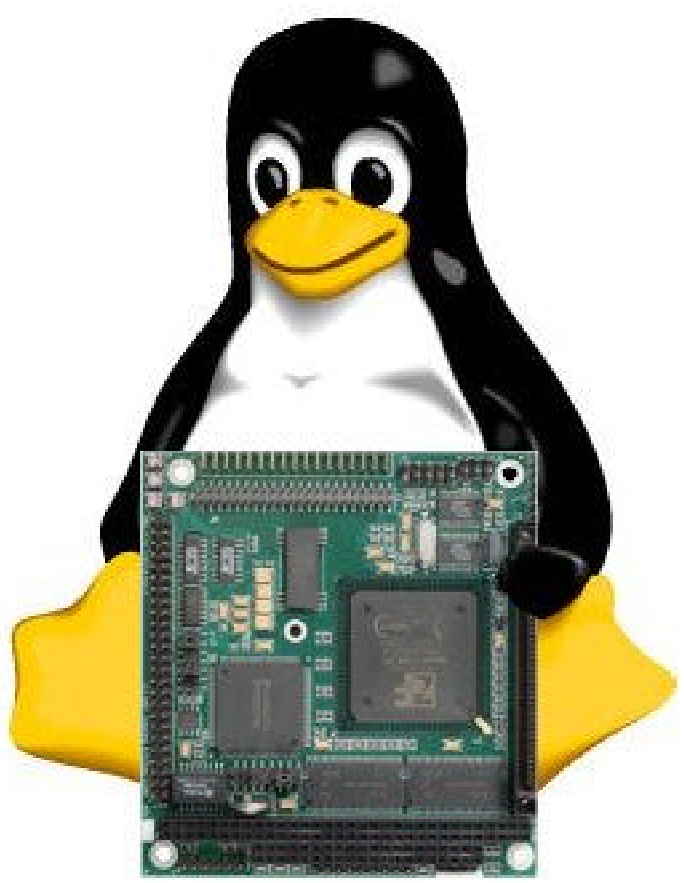
DIAMOND SYSTEMS CORPORATION

# Flash Linux Development Kit

***Embedded Linux Image for the Prometheus SBC***

User Manual V1.1



Diamond Systems Corporation
8430-D Central Ave.
Newark, CA 94560
Tel (510) 456-7800
Fax (510) 456-7878
techinfo@diamondsystems.com
www.diamondsystems.com

## TABLE OF CONTENTS

## 1. DESCRIPTION

The Flash Linux Development Kit targets developers who are embedding Linux on the Prometheus CPU board and do not have the time or background to create their own embedded system install. In particular, creating a 2.4 kernel based system for a 32MB flash module without a swap partition presents a number of challenges which can unnecessarily delay a project. Diamond Systems has created this Flash Linux development kit in response to this demand, so that customers can buy the flash disk with embedded Linux pre-installed and ready for deployment.

The customer can order embedded Linux pre-installed on a flash disk module, accompanied by a CD-ROM which contains the raw Linux file system image which can be used to create additional disks. This CD-ROM is available by itself for customers who already own a flash disk module. Additionally, Diamond Systems offers a 10GB IBM Travelstar hard drive preloaded with a full desktop Linux, including development tools.

### Features

**Miniature Linux Implementation**

Uses 6MB of RAM with Diamond Systems Universal Driver Software included

Uses 10MB of disk space

No swap partition (swap is not supported by flash media)

Boots from power-on to login prompt in 15 seconds

Shuts down in 4 seconds

Stable Minix file system for fast recovery from unexpected reboots

**Management Features**

Serial console enabled

Secure remote access with SSH enabled

Linux system programs provided by the BusyBox project

Full login security and user management provided by the TinyLogin project

Modern 2.4 kernel supports latest Linux features and hardware

Recent libraries - No need to link your application with legacy or custom C libraries.

**Customizability**

LILO installed for easy kernel management

Easy to install custom Linux kernels

Standard Linux system layout. No RAM disks or other tricks used.

**Free Software**

Linux system is free for redistribution and deployment

Source code available to all Linux system software

**Features of Hard Drive Linux Installation**

Graphical X Windows desktop pre-configured

Full development tools including compiler and debugger

Ability to debug software problems directly on the Prometheus CPU

# 2. SOFTWARE INSTALLATION

This section covers transferring the Linux file system image to a blank flash disk module. If you bought the image pre-installed and do not plan to make more copies, or only bought the hard drive version, you can skip to the next section.

This process requires a desktop system with a CD-ROM drive and a Linux distribution such as RedHat. You'll also need the Linux development kit CD-ROM which includes the image file to be copied

The development kit includes a Flashdisk Programming Board (model no. ACC-IDEEXT) which allows the desktop system to talk to the flash disk. Attach your flash disk to this board using the J2 pins. Power off the desktop system and attach the flashdisk programming board as the only device on the secondary IDE chain. The primary IDE chain should have your primary hard drive and CD-ROM drive. This is the configuration which Diamond Systems has determined is the most reliable. The instructions below assume that the flash disk is connected this way. If you set it up differently, take great care to substitute the correct options for your setup or the commands may fail or destroy your existing Linux installation.

Boot your computer and make sure that the BIOS displays a message showing that it has detected the flash disk. If it is missing, make sure everything is connected correctly and consult the troubleshooting tips below. During the Linux boot process a message will show that the flash disk has been detected as /dev/hdc. If a different device is shown, revisit the instructions above on how to add the flashdisk programming board to the system.

Once your development system has booted up, mount the Linux development kit CD-ROM and navigate to the "images" directory of the CD-ROM. You will see a number of images for the different sized flash disks that are supported. For example, if you are using a 32MB flash disk, use the "dsclinux-32.img" file.

Take a moment to be absolutely certain that you know the correct device (/dev/hdc or other) for the flash disk. <u>If you select the wrong one, for example the device for your desktop Linux hard drive, the contents will be destroyed by an incorrect copy</u>. Run the "df" command and make sure that /dev/hdc is not associated with any mounted file systems.

Type the following command to copy the selected image file to the flash disk.

```
dd if=dsclinux-32.img of=/dev/hdc bs=1k
```

Once this process is completed, turn off your desktop system, remove the IDE connector board, and attach the flash disk to the Prometheus CPU module. Read the next section before powering on the Prometheus.

## Troubleshooting Problems

- **The flash disk module is not detected by the BIOS.**

  Make sure the flash disk module is attached correctly to the IDE connector board. The Flash Programmer should be connected to a floppy power cable or other power source. The jumper on the flash disk should be in its default master IDE device position. If your IDE cable does not have a key pin filled, make sure that pin 1 of the IDE cable should match pin 1 on J4 where the cable is connected. Make sure the secondary IDE bus is enabled in your BIOS. If your BIOS is old, try setting the addressing mode for the flash disk in the BIOS to LBA.

- **The flash disk module is not detected by Linux.**

  If you are running a custom kernel, consider running the standard kernel for your Linux distribution. The flash disk appears to be a standard IDE device, so any Linux kernel which includes support for IDE on the secondary IDE bus should detect it. Linux may detect the flash disk but report errors related to DMA which can safely be ignored.

## 3. SERIAL CONSOLE REDIRECTION

Customers planning on using the serial console feature of either the embedded Linux or hard drive Linux images will have to make a single change to the Prometheus BIOS defaults to enable this feature. You can skip to the next section if you do not plan on using the serial console. See the instructions at the end of this section if you would like to turn serial console off.

Attach a null modem cable to COM B on the Prometheus and connect it to your desktop system running a serial terminal program such as Windows Hyperterminal. Configure the terminal program for 115k 8n1 with flow control set to none. Power on the Prometheus with the terminal program enabled.

The BIOS boot messages will be displayed. Hit F2 to enter the BIOS setup screen. In the Advanced screen, select Console Redirection. Change the speed to 9600 baud in the BIOS. Make sure that Continue After Post is set to Off. Save the BIOS changes and reboot. Change the speed of your serial terminal program to 9600 baud and reconnect. You will now see the boot process all the way through to a Linux login prompt in your serial terminal program.

### Troubleshooting Problems

- **Console Redirection does not show the BIOS boot messages.**

  Try a different null modem cable. Double check that the serial protocol settings in the Prometheus match those in your terminal program. Make sure you're using COM B on the Prometheus. As a last resort try resetting the BIOS to defaults and repeating the process above.

- **Console Redirection stops once Linux starts booting.**

  If you have a video board available, make sure that Linux is actually booting on the Prometheus. Repeat the steps in section 2 to write the Linux image to the flash disk as it may not have been copied correctly if Linux does not boot. Reset the BIOS to default settings and repeat the steps above.

- **How do I disable serial console redirection if I don't need it?**

  The LILO configuration file /etc/lilo.conf passes an option to the Linux kernel at startup which enables console on the serial line. Remove the line below from this file to disable serial console in Linux. Keep in mind that serial access to the BIOS only will still be enabled unless you change the BIOS settings for console redirection.

  ```
  append=”console=ttyS1,9600 console=tty0”
  ```

# 4. SECURE REMOTE ACCESS WITH SSH

Customers who do not have a video board and want to use SSH rather than serial console to connect to the Prometheus need to follow these steps when connecting for the first time as the network settings probably are not correct for your LAN. If you do not plan on using SSH, you can skip to the next section, but first see the instructions at the end of this section to disable SSH completely and avoid potential security holes.

Both the embedded Linux and hard drive Linux boot running an SSH server provided by the OpenSSH project. SSH is a secure protocol which supports telnet-style remote access and ftp-like file transfer, as well as advanced features such as tcp port forwarding. For more information see the OpenSSH website at openssh.org. Linux RPMs are widely available for installing an SSH client on your development system if they were not installed by default. There are also free and commercial SSH clients for Windows such as SecureCRT and putty.

The Linux image comes with networking pre-configured with defaults which almost certainly will not work on your LAN. To connect using this default IP address you'll need to connect to the Prometheus on an isolated network such as a solitary hub or directly using a "rollover cable".

The default IP address of the Prometheus is 192.168.1.221. On your isolated mini-LAN you can connect to the Prometheus using SSH from your Linux desktop using the following command.

```
ssh 192.168.1.221
```

The next two sections cover logging into the embedded or hard drive Linux system for the first time and required setup procedures such as configuring networking for your LAN.

## Troubleshooting Problems

♦ **I can't connect to the Prometheus via SSH.**

Check the ethernet link light on the Prometheus and your desktop systems to make sure they are correctly physically connected to the isolated LAN. If you are using a hub, check to make sure there are two link lights showing active connections and try a 10baseT hub if the current one is 100baseT. Otherwise try a different rollover cable. Try to ping 192.168.1.221 from your desktop system.

♦ **I can ping the Prometheus but SSH still fails.**

Your SSH client may be incompatible with OpenSSH. Try a different SSH client. If you are on Windows, make sure your SSH client is actually using the SSH protocol as some may default to telnet.

♦ **How do I disable SSH if I don't need it?**

In the embedded Linux, edit the /etc/rc.sysinit file and remove the reference to starting sshd. In the hard drive Linux, remove the sshd startup files in the /etc/rc.d/ directories. Diamond Systems strongly recommends that customers disable the SSH server once it is no longer needed to avoid any future security holes that may be discovered in the software.

## 5. FLASH DISK MODULE SETUP

This section covers details about the embedded Linux for flash disk modules. Customers only using the hard drive Linux can skip to the next section.

### 5.1 Logging in for the first time

You can login to the Linux system via serial terminal, an SSH client, or directly on the console if you have a VGA board. In any case, the system only has one user, called "root" and the password for root is "welcome". The first thing you should do after logging in is change this password to something more secure.

Configuration files for the system are found in the /etc directory. The file /etc/rc.sysinit performs startup tasks and launches programs at boot time. This file must be customized to the network settings by editing two lines using the vi command.

```
/sbin/ifconfig eth0 NEWIPADDRESS netmask YOURNETMASK
/sbin/route add default gw YOURROUTERADDRESS
```

Additionally, the DNS configuration file /etc/resolv.conf must be updated with the correct IP address for your nameserver.

Reboot the system and connect the Prometheus to your LAN. Login again with your new password and using the "ping" command to attempt IP communication with a device on your network such as your router to confirm that networking is working.

### 5.2 File system layout

This embedded Linux is a simple Linux install based on the system software provided by the BusyBox project and a recent 2.4 Linux kernel. The file system layout will be familiar to anyone who has explored Linux before.

| | |
|---|---|
| **/etc** | **Configuration and startup files** |
| **/bin /sbin /usr/bin** | **System programs** |
| **/lib /usr/lib** | **Libraries and kernel modules** |
| **/boot** | **Kernel and LILO files** |

Exploring the system, you will probably notice that almost all of the software is just a symlink to /bin/busybox. This program contains efficient implementations of the baseline functions of dozens of Linux system utilities. Note that even /sbin/init is provided by BusyBox, and the startup file /etc/inittab uses a format which is somewhat different from the standard Linux inittab. Visit the BusyBox project home page at busybox.net or contact Diamond Systems support for more information on how this system works.

## 5.3 Customizing the Linux Kernel

The embedded Linux system runs a custom 2.4 kernel built by Diamond Systems to include support for the Prometheus hardware and remove unnecessary features to conserve memory. USB is enabled. Kernel modules with driver signing is enabled. The process of running your own kernel is very similar to the normal Linux kernel upgrade steps, except that you need to build the kernel on your development system, and then copy the required files to the Prometheus. Below are basic instructions on building your own kernel. More details on this process are available on the Internet, or by contacting Diamond Systems support.

The first step is to download the source code to the Linux kernel you want to run to the /usr/src/linux directory on your development system. Run the following command in that directory to set kernel options.

```
make menuconfig
```

Make sure to set i486 as the target platform, and compile in support for the National Semiconductor DP83815 Ethernet chip using the natsemi driver if you plan on using Ethernet networking. You also need to compile in kernel support for the Minix file system. Note that in newer Linux kernels the natsemi_old driver should be used instead. Once you're finished setting kernel options, run these commands to compile and install the new kernel on your development system.

```
make depend
make clean
make bzImage
make install
make modules
make modules_install
```

This will create a new /boot/vmlinuz and /boot/System.map on your development system. Copy these files to your Prometheus with SSH using the following commands, substituting the correct IP address for the Prometheus system.

```
scp /boot/vmlinuz root@NEWIPADDRESS:/boot/vmlinuz.new
scp /boot/System.map root@NEWIPADDRESS:/boot/System.map.new
```

The Diamond Systems Universal Driver for Linux requires kernel modules which are compiled for your specific kernel when they are installed. If you switch to a new kernel, you also need to replace to kernel modules that come with the embedded Linux image. Install the DSCUD RPM on your development system and copy the two driver kernel modules in /lib/modules/misc to the same directory on the Prometheus system. Note that you may also have to copy other files from /lib/modules if your new kernel depends on them.

Now that you've copied over your new kernel, you must update LILO so that it will be loaded when you reboot. Edit the /etc/lilo.conf file to add the new kernel as the default and run the lilo command to save the changes. Diamond Systems strongly recommends that you leave the old kernel as a backup option in case your new kernel does not boot correctly.

## 5.4 Customizing the System

If you need any additional programs or libraries, you can simply copy them from your development system to the Prometheus. To conserve space you should be sure to run the "strip" command on these binaries to remove unnecessary symbolic information. When copying make sure that you do not use any programs that depend on i686 optimized libraries. The "ldd" command shows library dependencies if you are unsure. You may have to recompile programs from source to remove this dependency. Customers who also bought the hard drive Linux can simply always copy binaries from that system.

If there are any BusyBox programs which are missing features you require, you can just remove the BusyBox symbolic link for that program and copy over the full version from your development system.

LILO is configured to pause at boot up to give time to select alternate boot options. Before deployment you can edit the /etc/lilo.conf file to remove this pause which will give the fastest boot time.

## 6. HARD DRIVE SETUP

This section covers the hard drive install of Linux. Customers who are only using the flash disk version can skip to the next section.

Diamond Systems is pre-installing a full desktop Linux distribution on 10GB IBM Travelstar 2.5” hard drives which can be mounted directly in the PC104 stack. This is a standard desktop Linux installation, updated with the latest security fixes and upgrades, and including development tools and X Windows configured for the Arcom Controls PC/104 VGA board available through Diamond Systems or directly through Arcom (www.arcom.co.uk).

Installing a desktop Linux including upgrades can be a time and labor intensive process on the 486 Prometheus CPU. The Linux install alone takes an average of 3 hours, and upgrades run overnight. Standard Linux distributions also turn on a great deal of unnecessary software which must be selectively disabled to save resources.

Even if the target Linux system will be deployed on a flash module, it can be very useful to have development tools like the GNU compiler and debugger available on the Prometheus so that applications can be debugged on the target hardware.

### 6.1 Logging in for the first time

Attach the IBM Travelstar to your Prometheus CPU and boot the system. When the system finishes booting, login as the "root" user with the password "welcome" and then use the "passwd" command to change this default password right away.

The first thing you should do is configure the system for networking. The following configuration file must be edited to set the correct IP address for this system.

**/etc/sysconfig/network-scripts/ifcfg-eth0**

Next, the file below should be updated to set the system hostname and gateway IP address.

**/etc/sysconfig/networking**

Finally, you must edit the following configuration file to specify the IP address of the nameserver to be used for DNS queries on the system.

**/etc/resolv.conf**

## 6.2 System layout and configuration

The 10GB IBM 2.5" HDD has been partitioned so that roughly 1.6GB of space is allocated by Linux. The remaining unpartitioned space can added if needed, or ignored. For example, if logging is required by the application, customers may consider creating a separate partition for /var to protect the rest of the system against growing log files.

The Arcom VGA board which accompanies the development kit supports X Windows which is ready to run but not loaded by default. If a GUI is desired, type "startx" command to launch X Windows. A simple window manager called FVWM2 is installed which provides a good balance of speed and features. The X server is configured for 1024x768 at 8bit color with a 17" monitor.

The install includes all the packages one would expect from a standard desktop install, including the complete set of development tools, and applications can be developed and debugged directly on the Prometheus.

## 6.3 Changes to system standard

The Linux install is nearly identical to the standard distribution defaults. Changes are documented here.

The initialization files in /etc/rc.d/ have been modified to remove services which are not needed, to conserve memory. This also protects the system from security holes which may be present in these unnecessary programs. Each directory which has been modified has been backed up in case a service should be re-enabled. For example, the original /etc/rc.d/rc3.d files can be found in /etc/rc.d/rc3.d.DISABLED.

The cron files in /etc/cron.daily and /etc/cron.weekly have been modified to remove scheduled tasks which are not needed and would cause excessive file system activity. These directories have also been backed up so that these tasks can be re-enabled. For example, the original /etc/cron.daily can be found at /etc/cron.daily.DISABLED.

The latest release of the Diamond Systems Universal Driver has been pre-installed and will load at system startup.

# 7. REAL-TIME LINUX

The definition of real-time is fuzzy, and depends a lot on the specific requirements of the customer application. However, Linux is not a real-time operating system by any definition.  Testing is the key to determining if the standard Linux kernel meets your performance needs.  If you experience problems such as frequent FIFO overflows or system instability at high interrupt rates, this section provides tips on possible solutions.

Diamond Systems has found that Linux is capable at driving most of our hardware at peak performance so long as other resource intensive tasks are not running on the system.  If your application performs well in testing, you can skip to the next section.

## 7.1 Low interrupt latency kernel patches

The main problem with Linux real-time is that very high latencies (100ms or greater) can occur during normal activity such as disk access or screen activity such as scrolling text on the console. A very widely used solution to this to apply patches to the standard kernel which fix the parts of the kernel which are causing these delays. There is no "best" one, so the customer must research the patches below to select one or more which are ideal for their application. All of these solutions are available at no cost.

"Ingo Molnar's low-latency patch"
http://people.redhat.com/mingo/lowlatency-patches/

"Andrew Morton's scheduling latency patch"
http://www.zip.com.au/~akpm/linux/schedlat.html

MontaVista software's preemptible kernel patch"
http://www.linuxdevices.com/articles/AT4185744181.html

Additionally, the following article at O'Reilly on this subject is highly recommended
http://linux.oreillynet.com/pub/a/linux/2000/11/17/low_latency.html

## 7.2 FSM Labs Real-Time Microkernel

FSM Labs sells a unique solution for Linux real-time applications called **RTLinux**. This patented technology is a microkernel which runs all of Linux as a thread which can be pre-empted when a real-time task needs to run, which results in excellent performance. It is available as both a distribution called RTLinux and a patch to the standard Linux kernel. FSM Labs charges licensing and runtime feeds for commercial use of this technology. Customers needing the very best real-time performance from Linux and can to pay for it should consider this solution.

FSM Labs
http://www.fsmlabs.com/

## 7.3 Enterprise Linux Consulting

A number of companies offer specialized development kits, real-time Linux tools, training, and distributions which incorporate some of the above patches plus more custom improvements to the Linux kernel. Customers who are looking for enterprise support and tools should research these companies and contact their sales departments for more information. The Linux development kit provided by Diamond Systems is based on widely used Linux technology which can easily be adapted to the different real-time approaches taken by these vendors such as real-time kernels or APIs.

MontaVista Software, HardHat Linux, tools and consulting
http://www.mvista.com/

FSM Labs, RTLinux, tools and consulting
http://www.fsmlabs.com/

RedHat, tools and consulting
http://www.redhat.com/

LynuxWorks, BlueCat Linux, tools and consulting
http://www.lynuxworks.com/

Lineo, Embedix, tools and consulting
http://www.lineo.com/

## 8. SOFTWARE LICENSE INFORMATION

This development kit contains software which is licensed under the General Public License (GPL) which is a Free Software license. For more information on the GPL visit www.gnu.org. Diamond Systems includes the source code to BusyBox and TinyLogin on this CD-ROM. The latest sources to these packages are also available from their respective home pages at www.busybox.net and tinylogin.busybox.net respectively.

Source code to the rest of the software covered by the GPL in this development kit is available on CD-ROM from Diamond Systems for a reasonable fee. Get in touch with Diamond Systems support if you would like a copy of this source code.

Read the gpl.txt file included in this directory for more information on the terms of this license. If you have any questions about how this license affects your own product get in touch with Diamond Systems or visit the license section of the GNU website, www.gnu.org.