SERVER-IN-A-BOX

Manual

For EMAC Embedded Linux Distribution v3.1

REV. 3.4

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL
2390 EMAC Way, Carbondale, IL 62902
World Wide Web: http://www.emacinc.com
(618) 529-4525 FAX: (618) 457-0110

Table of Contents

Disclaimer

EMAC Inc. does not assume any liability arising out of the application or use of any of its products or designs. Products designed or distributed by EMAC Inc. are not intended for, or authorized to be used in, applications such as life support systems or for any other use in which the failure of the product could potentially result in personal injury, death or property damage.

If EMAC Inc. products are used in any of the aforementioned unintended or unauthorized applications, Purchaser shall indemnify and hold EMAC Inc. and its employees and officers harmless against all claims, costs, damages, expenses, and attorney fees that may directly or indirectly arise out of any claim of personal injury, death or property damage associated with such unintended or unauthorized use, even if it is alleged that EMAC Inc. was negligent in the design or manufacture of the product.

EMAC Inc. reserves the right to make changes to any products or documentation with the intent to improve overall quality, without further notification.

1 Introduction

The EMAC Server-In-A-Box (SIB) is designed to act as a general-purpose embedded Linux server. The SIB runs the Linux operating system kernel (a UNIX work-alike), and as such retains all of the networking and communications capabilities one would expect of such an operating system. The SIB v3.1 utilizes the Linux 2.6.14 kernel and GNU standard C libraries version 2.3.2 (glibc6 v2.3.2). EMAC Linux is based on the Debian GNU/Linux distribution.

Right out of the box, the SIB is configured for Ethernet, serial IP, serial terminal, and raw serial connections. It includes a web server, Secure Shell (SSH) server, TELNET server, FTP server, and periodic command scheduler. Furthermore, the SIB can be set up as an NFS server, SMB (MS Windows file sharing) server, mail relay, firewall, and many other configurations through additional EMAC add-on packages or by installing existing Debian GNU/Linux packages on the SIB.

This manual is designed to help you configure and work with the Linux system, and understand what to do and why. The *Features* section of this manual explains what the SIB can do, how it does it, and how each family of utilities can be set up and configured. This section will help the user to gain an understanding of the basics of the Linux operating system and environment. The *Configuration* section of this manual will explain the details of setting up the SIB using the menu-based EMAC SIB configuration utility. It is hoped that this format will first help the user understand what to configure and why, before explaining exactly how. Users with a thorough knowledge of Linux should at least skim the *Features* section, as this section explains important differences between the EMAC SIB and other (full-sized desk-top) Linux distributions.

2 Features

This section describes the features of the SIB as well as the basics of the EMAC Linux operating system. Read this section first for a background on how the Linux operating system functions and what is available before reading about how to use and configure the EMAC Linux operating system in section 3.

2.1 Users and Security

The Linux security model of separate users and password authentication is designed after the UNIX multiple user standard. For this reason, all users must create an account with a user name and password to be able to log into the machine. Each user is also a member of one or more groups. The basis for this design is security; all users have limited access to the file system. Each user (and each group) can have different permissions and privileges. These privileges take the form of "file permissions." Each file in a UNIX system has two ownerships. Every file is owned by a single user and group. Each file also has three sets of three permissions: read, write, and execute permissions may be specified separately on each file for the file's user owner, the file's group owner, and all other users on the system that do are not the owner or a member of the group. The following bit field expresses these permissions:

```
      User              Group             Others
read-write-execute-read-write-execute-read-write-execute
```

When describing permissions, this bit field is expressed as an octal (base 8) number. For example, permissions 754 mean that the user owner of the file may read, write, or execute the file (7). The group may read or execute, but not write (5). Other users on the system may read the file, but not write it or execute it (4). Permissions are also denoted by an "r" for read, a "w" for write, and an "x" for execute.

Nearly every important file on a UNIX system is owned by a single user and group, `root` (`root` user, `root` group). The root account is essential to the security of any UNIX system. Having all of the important executable and configurations files on the system owned by root means that only the root user may alter the way the system works. Other users have no permission to execute or change system files, and therefore, they cannot modify or damage the system.

As such, it is critical that the root password be kept secret, and changed as often as is practical. A clumsy or malicious individual with the SIB's root password could render the entire system inoperative in a matter of seconds. Every SIB ships from the factory with the same root password–this means that anyone who has ever purchased an SIB knows your root password! EMAC recommends you change the SIB's root password as soon as you've finished reading the manual.

Aside from the root account, which is present on every system, the SIB has one additional account configured. The username is `www`. This user does not have privileges to change the system configuration. Instead, this account is designed to allow a semi-privileged user to upload HTML files to the SIB. From there, the SIB's web server may serve them. This account may

also be used when transferring new executables to the SIB. While it is much more difficult for an individual with the www password to damage the SIB, it is still possible. EMAC therefore recommends you change www's password right after you change the root password. The default password for the www account is `emac_inc`, and can be changed using the `password` utility.

2.2 Command-line Tools

Certain standard command-line utilities are provided on nearly every UNIX / Linux system. The EMAC SIB includes as many of these utilities as is practical in the limited space available. The functionality of the majority of the basic commands on the system is provided by the binary `/bin/busybox`. Busybox provides a reduced size over the full versions of the commands. Although most of the functionality of the tools is preserved, you may find that some of these condensed command-line tools function slightly differently from the versions that you might find in a full-sized Linux distribution. While there are no manual pages on the SIB, you can find documentation for `busybox` and all of its tools in the file `busybox.html`. This can be accessed remotely from a web browser by entering the address `x.x.x.x/busybox.html`, where `x.x.x.x` is the IP address of the SIB.

The SIB's main shell is GNU `Bash`, with full scripting capability. Most of the standard command line tools are provided by busybox. In some cases, we felt that the Busybox implementation of a particular function was inadequate (tar, for example). In these cases, regular versions of these programs from full-sized distributions are used. Descriptions of some of the most common command line utilities follow. Keep in mind when running these programs that Linux is case-sensitive – capitalization matters. In all cases, the program name must be typed in all lowercase letters.

The SIB includes two simple text editors, `vi` and `nano`. If you are familiar with these programs from other Linux / UNIX distributions, the functionality remains fundamentally the same. You will find, however, that some of the functions of the larger versions of these editors are not available, especially with `vi`. To invoke these editors, type `vi <filename>` or `nano <filename>` at the shell prompt. If you are unfamiliar with their operation, it is best to obtain a commands list explaining their operation as a reference. Although these editors allow for simple editing of files on the SIB, More extensive editing of files could be done more easily on a remote machine, passing the files between machines using a file transfer utility such as SSH, FTP, or a serial terminal program. See the links in section 4 for more information about `vi` and `nano`.

The `ls` utility works much like the dir command in DOS. With no parameters, `ls` will list the files in the current directory. Directory name may also be specified on the command line, in which case `ls` will list the contents of this directory instead. For more detail, the "`-l`" switch (dash-ell, no quotes) may be specified between `ls` and the directory name. This will display information about files including ownership, permissions, size, and modification times.

The `cd` command changes the current directory, exactly as in DOS. As Linux has no concept of "drives," all files are kept in a single unified file system. Forward slashes (`/`) are used to separate subdirectory names.

Each user has a "home directory." This directory is where a user will begin after logging in. For the root user, this directory is `/root/`. For other users, this directory will usually be `/home/<username>/`. A user's home directory may be abbreviated by a tilde character (`~`) when using the command line interface. For example, `cd ~` will bring you to your home directory. The command `cd ~foo` will take you to user foo's home directory (assuming that you have permission to read that directory). Similarly, `~foo/bar` will execute a program called `bar` located in foo's home directory. Note that a tilde abbreviation should never be prefixed with a '`/`'.

The `chown`, `chgrp`, and `chmod` commands modify file ownerships and permissions. The `chown` utility will change the owner of a file to one specified on the command line and `chgrp` will change the group owner of a file in the same manner. The `chmod` utility changes the permissions of a file. Typically, the new permissions are specified numerically as described previously, but the `chmod` utility also supports alternative syntax. See the `chown` documentation in the busybox documentation (html) for more information. Note that you must have write permission on a file to use these utilities on it. The following are a few examples of these utilities:

`chown www /home/www/file` (changes the owner of `/home/www/file` to www)
`chown www:www /home/www/file` (changes the owner and group of `/home/www/file` to www)
`chgrp www /home/www/file` (changes the group of `/home/www/file` to www)
`chmod 754 /home/www/file` (changes the permissions of `/home/www/file` to 754).

The `cat` utility will display the entire contents of a file to the screen. This should only be done with text files, as "catting" binary files can cause the shell to enter odd states in which it prints unreadable characters. There is no way to pause the output from `cat`. The `more` utility fixes this shortcoming by displaying a specified file one page at a time. Simply press a key to view the next page of text. The `more` utility is the standard way to view text files on the SIB.

To move a file in the SIB filesystem, use the `mv` command. Its syntax is identical to the DOS `move` command. The `cp` command is likewise similar to the DOS `copy` utility. Linux does not have a rename function, but `mv` can be used to change a file's name. The `rm` command will delete a file. To delete a directory, the "`-r`" switch must be specified before the directory name to delete. To prevent `rm` from prompting for confirmation, use the "`-f`" switch. CAUTION: there is no undelete utility. Files deleted from the SIB are permanently gone.

2.3 Loadable Modules and Device Drivers

In Linux, device drivers can either be compiled into the kernel, or compiled separately as a loadable module. The SIB supports and utilizes both methods. Drivers that have been compiled into the kernel are always active, and do not require any input from the user or the system. Drivers such as TCP/IP, SLIP, ramdisk, and serial are compiled into the kernel.

Modular drivers, on the other hand, must be explicitly loaded into the kernel after the machine has booted. This is most commonly done through startup (boot) scripts, but can be done at any time. Modules are stored in the subdirectories of `/lib/modules/2.6.14.7-430-standard` (or similar depending on your kernel version), indexed by type. Drivers such as parallel port, serial port, and Ethernet have been compiled as modules, to be loaded as needed. Note that some drivers require parameters on the command line. In particular, ISA peripherals like network and sound hardware generally require that their I/O base address be specified.

There are several useful commands for manually managing module use. These include the commands `insmod` and `rmmod`, which can be used to load and unload modules respectively. Furthermore, all currently loaded modules can be listed using the `lsmod` command. If the required modules are properly installed in the system, they can be automatically loaded by using `modprobe`, rather than the `insmod` utility.

The modules to be loaded automatically at boot time should be listed in the file `/etc/modules`. A number of these are already listed there, such as Ethernet and PPP modules. Command line parameters for these modules may also be specified in this file. Due to space constraints, the SIB includes very few of the available modules. EMAC offers a number of modules/packages that can be optionally added to any build. See http://www.emacinc.com/operating_systems/modules for more information.

A caveat of using modules deals with dependency issues. Some modules depend on others, and will refuse to load until the dependencies have been satisfied. The SIB cannot automatically resolve these dependencies using `modprobe` unless they are included in the kernel modules.dep file, so a user must explicitly specify the loading of any module dependencies before loading the desired modules. All of these dependencies should be addressed when the SIB build is set up, but this issue will need to be addressed if the user adds any custom modules. Some common examples of dependencies are the NE2000 driver (`ne.o`), which requires the 8390.o module, and the SLIP and PPP modules (`slip.o` and `ppp.o`), which require the `slhc.o` module. These could be loaded in order manually or by a script using `insmod` which does not require a dependency reference to load a module.

2.4 Flash Disks and RAM Disks

The EMAC SIB uses both flash disks and RAM disks during operation, with the operating system itself being ran from a flash disk. Be careful that you do not insert or eject the flash disk from the SIB when the unit is powered up. This can cause damage to the flash card and corrupt the operating system.

The SIB typically runs its operating system from flash media. However, the SIB does use RAM disks for some tasks. Flash media can only support a limited number of write operations (although it can be read an unlimited number of times). As such, a RAM disk is used for the directory tree which is most often written to: the `/var` directories.

The `/var` directories contain files which are intended to be changed often. Lock files, temporary files, and log files are all kept in `/var`. Keeping the `/var` directories in a RAM disk saves a great deal of wear on the flash. However, it also means that log files are lost any time the SIB loses power. If you wish to keep log files on a permanent basis, you should use the

periodic commend scheduler CRON (described in section 2.6) to copy the logs to flash, or to send them to a remote computer via FTP or SCP (SSH Secure Copy).

As with most other operation systems, a Linux system should generally not be powered off without first shutting down the operating system. This allows Linux to flush its buffers and finish any pending writes to disk. If a Linux system is powered down without warning, data may be lost. In the worst case, a power down could occur while a write is taking place, causing corruption of the filesystem. This is particularly dangerous when using flash, as a flash device must be erased before it can be written, greatly increasing the time in which the media is vulnerable to corruption. The SIB can be properly shutdown by using the command `shutdown –h now`. To reboot the device, type `reboot`, or `shutdown –r now` at the shell prompt.

To protect against possible file corruption due to loss of power, EMAC has built in options to easily mount the filesystem "read only." If no writes can be performed, no corruption can occur. Mounting read only is done in the boot scripts. If the file `/etc/READONLY` exists, the SIB will attempt to mount its root filesystem read only. This option can be changed through the menu configuration utility described in section 3.

It is also possible to change the filesystem from read only to read/write and back from the command line. The command `mount –o remount,rw /` will remount the filesystem in read/write mode. Using the same command but replacing "`rw`" with "`ro`" will remount the flash read only. It is also possible to change the way the filesystem is mounted for the current session using the system configuration menu utility.

Note that when the root filesystem is mounted read only, the `/dev` directory (device nodes) is mounted in a second RAM disk. This allows the system to write log files and for users to log in even when the filesystem is read only. If you use the commands above to remount the flash read/write, changes made to `/dev` will be lost when the system loses power. To make persistent changes to `/dev`, you must first boot the device into read/write mode. This behavior can be changed using the system configuration utility.

2.5 Login Terminals

There are a number of different ways to access and login to the SIB. These include the console, serial terminals, dial-in terminals, SSH, FTP, and TELNET. For connection via a network, SSH is preferred over TELNET and FTP because all data is exchanged through an authenticated and encrypted tunnel between devices.

2.5.1 Console and Serial Terminal

The first way to access the SIB is via the console. If the unit is equipped with video, the console is accessed in the traditional manner, by connecting a monitor and keyboard to the SIB. In this instance, you may also optionally configure the SIB with a serial console by editing `/etc/lilo.conf.devname`, where devname is the boot device used, and uncommenting the references to the serial console. Then re-run LILO to update your changes. You will also need to enable login via a serial port by editing the file `/etc/inittab` and uncommenting getty for the serial terminal you wish to use.

If the unit ships without video it will be preconfigured to use the console, the console is accessed by connecting a null modem cable to `COM1` / `ttyS0`, the first serial port of the SIB with the other end connected to a serial port on a desktop machine. From here, you can give the LILO bootloader boot commands, and login to the SIB. During the boot process, messages will be displayed to the screen detailing the startup process. The serial terminal settings are: 9600 baud, 8 data bits, 1 stop bit, no parity, no flow control, and vt100 terminal emulation.

When a "dumb terminal" is plugged into this serial port, a login prompt appears, allowing a user to login into the SIB. Alternately, a second PC can be plugged into `COM1` / `ttyS0` using a NULL modem cable, and a terminal program such as Minicom or HyperTerminal may be used. It is important that when using a second PC, that a "NULL modem" or "crossover " cable be used, as a standard serial cable will not work.

2.5.2 TELNET Server

Warning: using the TELNET with the SIB will expose all data "in the clear" to everyone in the network you are using. This includes your root password and any data you transmit between the SIB and your computer. If you are on an insecure connection on which you do not trust all parties on the network, consider using the SSH Server described below. **TELNET should not be used if the SIB is connected directly to the Internet outside of any firewall or NAT router.**

A TELNET server is available on all SIBs. For this to work, the SIB's Ethernet port must be connected to the network, and its networking must be configured properly. In particular, you must set the SIB's IP address and routing data to reflect your LAN. If you are using DHCP, you must know what IP address your DHCP server assigns the SIB. Ask the administrator of your DHCP server to provide you with this information. You can test the network setup of the SIB by using the ping program found on most network operating systems. If the SIB responds on the expected IP address, you can then use the TELNET client included with the operating system on another machine to TELNET to the SIB and login.

2.5.3 Dial-in Serial Terminal

Another login method is available on SIBs with the modem option installed. These SIBs can be set up to answer the phone, then provide a serial terminal on the modem line. This configuration is also kept in the `/etc/inittab` file and controlled through the system configuration menu utility. With the exception of answering the phone line, this terminal type behaves exactly as a serial terminal.

2.5.4 SSH Server

SSH, Secure Shell, is a replacement for TELNET with encryption. SSH uses standard SSL encryption which is the same method that is used by web browsers to connect to secure web sites on the Internet. If you intend to use SSH instead of TELNET to remotely login to the SIB, it is recommended that you disable the TELNET daemon by commenting out the telnetd line in `/etc/inetd.conf`.

To establish an SSH connection with the SIB, you will need an SSH client of some type available on the machine that you are using to connect to the SIB. For Linux, the OpenSSH client is a standard package in most distributions. For a graphical connection, you may use a program such as Konqueror to establish an SFTP connection. To do this, type `sftp://x.x.x.x`, where `x.x.x.x` is the IP address or hostname of the SIB, in the address bar. You will then be allowed to login and edit files, browse the filesystem, and upload and download files. You may also use the FISH protocol for this connection, which is similar to SFTP (`fish://x.x.x.x`).

For Windows, a program called Putty will allow you to establish a connection to the SIB through the SSH server. A graphical connection can be established through a program such as WinSCP, which will work in conjunction with Putty. See the links in section 4 for more information about these programs. Note that Internet Explorer will not support the SFTP protocol, and will default to unencrypted FTP without notification if you attempt to establish an SFTP connection.

To increase security, it is wise to generate a key pair to be used during the SSH connection. The steps using the OpenSSH client are given below; putty provides a similar method described in the putty documentation. Perform the following steps to generate a key pair and connect to the remote machine:

1. From the shell on the machine that you are attempting to connect to the SIB with, execute the command `ssh-keygen -t dsa`, which will create the files `~/.ssh/id_dsa` and `~/.ssh/id_dsa.pub` ('~' represents the users home directory). You may use any password (or no password) during the key creation; it does not need to match the password of any users. The `.pub` file contains the public key that needs to be copied to the SIB.

2. (Securely) Copy the file to the SIB using a command such as `scp ~/.ssh/id_dsa.pub user@x.x.x.x:/home/user/`, where `user` is your username on the SIB, `/home/user` is your home directory on the SIB, and `x.x.x.x` is the IP address or host name of the SIB.

3. Login to the SIB (under the account that you are creating the key for) either directly (via the console) or remotely, and add the key to the authorized keys file.

    a. If the directory `~/.ssh` and the file `~/.ssh/authorized_keys` do not exist, create them now.
    b. Concatenate the key to the authorized keys file by executing the following command:

    `cat ~/id_dsa.pub >> ~/.ssh/authorized_keys`.

4. Attempt to connect via an SSH connection (`ssh user@x.x.x.x`). The password used to login will be the one used in the creation of the key pair.

5. Delete the `id_dsa.pub` file in the user's home directory on the SIB.

2.5.5 Access Issues and Devices

To increase security, the root user is only allowed to login via certain "devices." Every terminal type uses a different type of device. The root user must be explicitly allowed to login on each device. However, since UNIX/Linux allows for multiple users, these are multiple devices of each type.

The console uses devices `tty0` through `tty4` if the SIB is equipped with video. To switch between consoles from a direct console connection to the SIB, press `<alt>+F(1,2,3,4,5)` (i.e., pressing `<alt>F2` will switch to the second console). Console 5 is reserved for the X Windows Server on SIB's with the X Windows package installed. Otherwise, it will use a serial terminal. Serial terminals use devices `ttyS0` – `ttyS3` (these correspond to `COM1` – `COM4`). Since modems are generally treated as serial devices, dial-in terminals also use a `ttySx` device, depending on which serial port the modem occupies (usually `ttyS2`). TELNET uses devices `ttyp0` and so on; there are as many `ttyp*` devices as the number of users that may be logged into TELNET at once.

The root user may only login on devices explicitly listed in the file `/etc/securetty`. Should changes need to be made, this file must be edited by hand. By default, several TELNET devices are listed in this file. This aids remote configuration and troubleshooting, but it is also often a security risk. If you do not intend to configure or manage the SIB via TELNET, it is recommended that the TELNET devices (`ttyp*`) be deleted from this file. IT IS IMPORTANT THAT THE CONSOLE AND SERIAL DEVICES NOT BE REMOVED FROM THIS FILE. If that happened, root would be unable to log in, and configuration and management of the SIB would be impossible.

When a user logs in, certin scripts are automatically run. These scripts will set the user's path, prompt, and other environmental variables. The scripts `/etc/profile`, `~/.profile`, and `~/.bashrc` are all executed, in order. These scripts are by default executed by the `Bash` shell. The scripts in your home directory may be changed should you wish to add to the path, set a command alias, or run a program every time you log in.

This is the case with the root user. Every time root logs in, the system configuration menu is started by the `/root/.profile` script. Note, however, that only the `~/.bashrc` script is executed if a shell is not the user's login shell. Thus, the configuration system is not started again when the root user requests a second shell (bash prompt).

2.6 Periodic Command Scheduler

The SIB comes with the standard UNIX periodic command scheduler, CRON ("Vixie" implementation). This is a program which starts at boot time and runs in the background, executing specified commands at specified times. CRON is configured by editing its configuration file `/etc/crontab`, which may be done through the main system configuration utility.

The crontab file consists of zero or more entries consisting of a set time to execute a specific command. Since CRON executes on its own (with no user intervention), crontab may also include statements which set and export the `PATH` shell variable (the default crontab file contains such statements, and should not be modified except by an experienced user). When editing the crontab file, times may be specified from the minute up to the month. The specific format for a crontab entry is:

```
<minute> <hour> <day-of-month> <month> <day-of-week> <user> <command>
```

All times may be specified by a number, and the hour field uses a 24-hour clock. The day of the week may be specified numerically, or by the first three letters of its name. When specified numerically, the day of the week may be 0-7, with both 0 and 7 representing Sunday. Ranges and lists of times are supported. Thus, placing "5-10" in the hour field will cause the command to be executed every hour from 5 AM to 10 AM. The equivalent (comma-separated) list would be "5, 6, 7, 8, 9, 10". Note that lists and ranges are not supported when using weekday names. Divisors are also supported. For example, "5-10/2" in the hour column would execute the given command once every other hour between 5 and 10 AM.

Finally, "any" or "all" may be specified by placing a star ("*") in the field. For example, the line

```
42 0 * 1, 5 2-4 root /bin/echo "CRON command is executing" > /dev/console
```

will print the message "CRON command is executing" to the SIB's console at 12:42 AM every Tuesday through Thursday during January and May. The field "user" contains the user that the command will be executed as. In most cases, this will be

root. (Note that root has special privileges; care should be taken that essential system files are not modified by crontab entries.)

Crontab has several pre-defined entries designed to perform standard system "housekeeping" tasks. These entries are designed to run every file (usually shell scripts) in a particular /etc subdirectory at regular intervals. These subdirectories are `cron.daily/`, `cron.weekly/`, and `cron.monthly/`. Simply put, everything in `/etc/cron.daily/` will be executed once per day (frequently late at night so as not to slow down user operations). Everything in `/etc/cron.weekly/` will be executed once per week, and so on. This can be a good mechanism to limit the lengths of log files, and basic scripts to manage log files are already in the CRON directories. User files placed into these subdirectories will be executed along with the log file management scripts, providing a simple alternative to making actual crontab entries.

2.7 Developing Your Own Applications

The SIB has great potential and flexibility for running custom applications due to its fully functioning Linux operating system. This will allow you to run any application that would run on a full-sized Linux operating system, assuming that all of the necessary packages and libraries are available.

Due to its size, the SIB is not a development environment in and of itself. It does not include a compiler, standard headers, help files, or advanced editor. This requires the use of a machine with development capabilities and utilities in order to design and implement custom applications for the SIB.

2.7.1 The EMAC SIB-SDK

One of the common problems encountered when developing applications for the SIB is a result of library dependencies and incompatibilities. If the development environment does not successfully emulate the system on the SIB by using the same libraries, library versions, and kernel, it can be difficult to develop applications. In the past, the common solution to this was to maintain a static development environment using one of the modern Linux distributions that matched the environment on the SIB with the same kernel and standard C library versions. However, EMAC has developed a Software Development Kit (SDK) for the SIB, which allows users to guarantee compatibility of their applications.

The EMAC SIB-SDK is obtainable through EMAC for free, and contains everything needed to develop applications that will run on the SIB without having to maintain a static environment on your development machine. The SDK also provides cross-compilers, example projects, and methods for interfacing with the SIB and debugging applications. This also provides a method to develop under Microsoft Windows.

Complete documentation explaining how to install and use the EMAC Eclipse SDK is available in the EMAC document titled "Eclipse SIB-SDK Manual." See the links in section 4 for more information on where to obtain this document.

2.7.2 Languages and Libraries

Many languages and compilers are available for Linux, including C/C++, Fortran, Java, BASIC, assembly, and Perl. Not all languages are supported by default. For example, the default SIB configuration does not include interpreters for Java, Perl, Python, or any other interpreted language (shell script excepted). Compiled languages such as C, Fortran, and assembly do not require special interpreters to execute. However, a particular program may need certain "shared objects" added to the system before they will execute.

To save disk space, GNU/Linux systems keep code used by multiple applications in "shared object" libraries. The libraries work much like `.dll` files in MS Windows. The libraries are usually found in the `/lib`, `/usr/lib`, and `/usr/local/lib` directories, and can be identified by their `.so.<version>` extensions. The `/lib` directory also contains the dynamic linker, `ld-linux.so.2`.

Because flash space is limited, the SIB only contains the most commonly used libraries, such as the C libraries, network name resolution, and the ncurses library. Many programs will need other libraries to run. If your application will not run, use the command `ldd <filename>` to display a list of the libraries your application needs. If any of the libraries are listed as "not found," then you will need to either copy that library to the SIB, or specify static linking when you compile your program.

It is also important to note that many “libraries” are actually dynamic links to files with similar names. Depending on the compiler, an application may search for a library by several different names. If you copy a library to the SIB and your application still does not find it, you may need to create a link to the library with a different name. Shared objects are also cached, along with their locations. If your application continues to insist that a library does not exist even though it does, you may need to use a program called `ldconfig`. Copy this binary to the SIB from your development workstation, and execute it once, then try your application again – odds are that it will now function properly.

If your program works properly on your development workstation but reports “segmentation faults” when run on the SIB, there are several possibilities. The first is that your program was linked with a newer version of the standard C libraries, and the SIB’s libraries are not forward compatible. This behavior was seen, for example, when programs linked against glibc 2.1.2 were run using glibc 2.0.7. To reduce this problem, the SIB was created using the newest libraries available at the time. However technology is sure to progress. Development under the SIB-SDK will also eliminate this problem.
The second major possibility is that your program binary was corrupted in transfer. In particular, the DOS-based FTP client included with MS Windows ‘95 and ‘98 default to ASCII mode, meaning that the FTP client may modify the file it is transferring. This FTP client must explicitly be set to transfer binary files. Other FTP clients may have similar behavior.

2.7.3 Startup Scripts and Init

It can be desirable to have an application start at boot time. There are two ways to accomplish this: via setup scripts, or via `init`. Init is that “mother of all processes.” Its task is to start login terminals on specified devices, and usually keep them running by re-starting them when they exit. Init is configured by the `/etc/inittab` file. Init is designed to associate tasks with devices. If your application will use a single serial port or a virtual console terminal, then init may be an appropriate method to run your application each time the SIB boots.

If your application is not associated with a single device, then it should be started via boot scripts. These scripts perform much the same tasks as the DOS `autoexec.bat` script, although on a much grander scale. SIB boot scripts are stored in the `/etc/init.d` directory, however they are not executed from that location. When init starts, it executes a script called `rcS`. This script in turn starts several other scripts. Scripts in `/etc/rcS.d` are executed in numerical then alphabetical order. Only scripts that start with the letter ‘`S`’ are executed. A two-digit number after the ‘`S`’ is used to control order of execution. Note that these scripts in `/etc/rcS.d` are actually links to the actual scripts in `/etc/init.d`.

Once the scripts in `/etc/rcS.d` have been executed, another directory of scripts is executed. Which directory depends on the “run level” of the system. The run level is a concept used by UNIX to make it easy to control the starting and stopping of different services at boot time. The default run level is set in `/etc/inittab`. The SIB’s default run level is 2. When run level 2 is entered, scripts in `/etc/rc2.d` are executed. Similarly, when run level 5 is entered, the scripts in `/etc/rc5.d` will be executed.

When the system leaves one run level, it first attempts to “kill” services that are no longer desired by running scripts in the appropriate directory that begin with ‘`K`’. For example, when the system leaves run level 2, all scripts in `/etc/rc2.d` beginning with ‘`K`’ are executed with the command line option ‘`stop`’. The start scripts (scripts that begins with ‘`S`’) are then executed in the directory command line option ‘`start`’ for the new run level.

To start your application at boot time, create a shell script which starts your program and place it in the `/etc/init.d` directory. Be certain to set execute permissions on your script. Then, in the directory associated with the desired run level, create a link to your script. Make sure the link name begins with ‘`S`’, and use a number that falls after the rest of the scripts (‘`S99`' is usually safe). Your script will then run after all other system configuration has taken place, and will start your application.

2.8 Networking

As a UNIX work-alike, Linux is built with extensive networking capability, making it an ideal choice for a server. The SIB benefits from this Linux feature, inheriting the entire range of networking options. As a result, reliable network connections can be made easily and offer a wide range of configuration options to suit the user's needs.

2.8.1 Network Addressing

The SIB communicates with other computers using the standard suite of Internet Protocols (IP). Computers using IP talk to one another by means of an IP address. Each machine on the Internet has a unique IP address, assigned by their network

administrator. An IP address usually consists of four numbers (0-255), separated by dots. Examples are `192.168.1.50` and `10.0.2.1`.

The SIB has several ways in which it can connect to a network. The SIB can use IP over one or more serial ports (SLIP devices,) modems (PPP devices,) or Ethernet devices. These devices are referred to as "interfaces". A particular interface is described by a combination of letters, which describe the type of interface, and a number. Interfaces of a particular type are numbered from 0, in the order that they were activated. For example, the first Ethernet interface activated is identified as "`eth0`". Similarly, the first SLIP device activated is "`sl0`," and the first PPP device is "`ppp0`." Note that `eth0`, `sl0`, and `ppp0` may all exist on the same machine at the same time.

An SIB is not restricted to having a single IP address. In fact, every network interface may have a different IP address. For example, the `eth0` interface may have address `10.0.2.41` (the factory default). IP packets arriving at that interface addressed to `192.168.0.5` would be ignored. However, at the same time, the `ppp0` interface may be assigned IP address `192.168.0.5`, and accept packets so addressed.

Different IP addresses on the same SIB bring us to the concept of routing. The SIB's kernel holds an internal routing table, which tells the network subsystem where to find remote computers. Specifically, the routing table holds zero or more entries stating which computers or "networks" may be found on a particular interface. For example, a single routing entry may specify that packets addressed to the computer with IP address `124.172.18.9` should be sent out over interface `sl0`.

A "network" is a group of computers with similar IP addresses. A network of computers may have one or more sections of their IP addresses in common. For example, computers with IP addresses `10.0.2.4` and `10.0.2.5` are on the same network. The "network mask" (netmask) determines the number of sections that the network machines have in common. When the IP addresses and network mask are converted to binary, the network mask contains 1's in all locations where the IP addresses match. Zeros in the network mask may not appear to the left of 1's. For example, `10.0.2.5` and `10.0.2.78` are in a network with network mask `255.255.255.0` (this is called a "class C" network). Machines with addresses `127.6.83.9` and `127.4.14.62` would be on a network with network mask `255.0.0.0` (a "class A" network, very uncommon).

Each network also has a "broadcast address". The broadcast address is an IP address consisting of all of the sections of the IP address that the machines have in common and 0's in the rest. For example, in a network with machines numbered `192.168.*.*` and network mask `255.255.0.0`, the broadcast address is `192.168.0.0`. All machines on a network will receive a packet to the broadcast address.

It is common for an IP packet to be addressed to a machine that cannot be located in the SIB kernel's routing table. These packets are dealt with by means of a "default gateway." An SIB may have zero or one default gateway(s). The default gateway must be locatable by the kernel's routing table. Whenever a packet is encountered which is addressed to an unknown machine, it is sent to the default gateway. This gateway is almost always a switch, bridge, or router between one or more networks.

Each interface has an IP address, network mask, and broadcast address associated with it. This information may be set or viewed via the "`ifconfig`" command. With no parameters, `ifconfig` will display information for every active interface. Alternately, a particular interface may be specified on the command line, whose information will be displayed whether the interface is active or not (though the device must exist; the driver must be loaded). Interface configuration should be done through the main system menu configuration utility (described in section 3).

The kernel routing table may be changed or viewed using the "`route`" command. With no parameters, the `route` command will display the entire kernel routing table using hostnames instead of IP addresses where possible (hostnames are described in section 2.8.2). Each routing entry consists of a host or network address, a network mask, and an interface. The default gateway is an exception; this entry does not include an interface. The kernel must be able to locate the default gateway using the rest of the information in the routing table. Routing information should be changed through the main menu configuration utility.

If your SIB does not use nameservers (described in section 2.8.2), it is recommended that you use the "`-n`" command line switch with `route`. If you do not, the program will hang while it attempts to resolve IP addresses into hostnames, which will ultimately fail.

Some networks want to assign IP addresses and other interface information using a protocol named DHCP. The SIB includes a DHCP client, which is capable of automatically configuring the `eth0` interface. You must have a DHCP server configured on your network to use DHCP. Configuring interfaces to use DHCP should be done through the main system configuration menu.

2.8.2 Name Resolution

Internet servers are frequently referred to by a "hostname", rather than by IP address. The hostname takes the form of two or more strings of characters separated by dots, such as `www.emacinc.com`. The sections of the hostname generally have no correspondence to the sections of the IP address.

Internet-wide hostnames are registered with one of a few companies, usually for a fee. A registrant must also provide the IP address that a hostname is associated with. Hostname / IP address pairs are stored on DNS machines (Domain Name Service). DNS servers may run on machines with other services, such as mail and web server. Your company network and/or ISP should be able to provide you with a list of name servers.

A hostname does not need to be Internet-wide; every UNIX box has and uses a hostname, at least internally. The factory default hostname for the SIB is "`SIBx31`" (no quotes.) The hostname of the SIB is set at boot time, using the `hostname` utility. Without arguments, this utility will also display the system's current hostname. The name set at boot time is the name contained in the `/etc/hostname` file. The contents of this file may be edited manually, or through the master configuration utility. Note that simply editing this file does not change the hostname of a running system; the hostname utility must be used, or the system may be rebooted. The full system hostname is often displayed (as in the background on the main menu configuration utility) as a concatenation of the hostname stored in `/etc/hostname` and the current domain name (i.e. `SIBx31.emacinc.com`).

The SIB has two ways to resolve hostnames into IP addresses. The simplest is locally, via the `/etc/hosts` file. This file contains a local listing of hostname / IP address pairs which are accessible only to the SIB. This file always includes one entry, the `localhost` / `127.0.0.1` pair. Generally, this file should also include a second entry relating the SIB's set hostname with the IP address of interface `eth0`. This entry is important, as some network applications will refuse to run if they cannot resolve the hostname of the SIB into the IP address of the interface they wish to use. The `/etc/hosts` file should be configured through the main system configuration menu.

The second way the SIB can resolve hostnames into IP addresses is via DNS servers. Valid DNS servers should be listed in the `/etc/resolv.conf`. Two to three DNS nameservers are common, but more are allowed. Note that if an application on the SIB attempts to resolve a hostname to an IP address via DNS servers, the application may stop for several minutes while the SIB attempts to contact the DNS server, especially if the DNS is inaccessible and the operation times out.
By default, the `/etc/hosts` file is searched before a DNS is accessed to resolve a name. This behavior may be altered by the `/etc/host.conf` file, which must be edited by hand. However, it is recommended that only an experienced Linux user do so.

2.8.3 Access Control

Many common Internet services (HTTP, FTP, TELNET) on the SIB are managed through a "superserver" called `inetd`. This superserver reads all incoming IP traffic, and then passes the data to the appropriate service. Since all IP traffic passes through this single program, this allows for unified access control; the same configuration files can control access to many different Internet services.

Control begins with the `/etc/services` file. This file contains a master list of all TCP and UDP ports that `inetd` will listen on, and the appropriate service to pass data on that port to. If a port is not listed in this file, then `inetd` will ignore traffic on it (though other stand-alone network servers may listen on that port.) In general, this file should not be edited unless security is a major concern, and then only by those already knowledgeable about Linux.

The next line of defense are the files `/etc/hosts.allow` and `/etc/hosts.deny`. As their names suggest, these files allow a system administrator to explicitly list servers, services, users, networks, and domains which may or may not use the `inetd` services (a "domain" is the hostname equivalent of a network).

The `hosts.allow` file takes the greatest precedence. If an entity is granted access by the `hosts.allow` file, then access will be granted to that entity no matter what is listed in `hosts.deny`. The `hosts.deny` file lists entities that should be

denied access to the SIB. If an entity appears in neither file, the default action is to allow access. By default, the SIB ships with a relatively permissive access policy. `Hosts.allow` and `hosts.deny` may be edited by hand should you wish to alter the access policy. EMAC recommends that only an experienced Linux user do this. The format for these files is beyond the scope of this document.

The FTP server is configured by the files `/etc/ftpaccess`, `/etc/ftpusers`, `/etc/ftphosts`, and `/etc/ftpgroups`. Not all of these files are present on the SIB, though they may be added if desired. The file `ftpaccess` is the main FTPD configuration file while the others are used for fine-grained access control. The format of these files is beyond the scope of this document. Configuration of the FTP server is accomplished by editing these files by hand. EMAC recommends that only an experienced user do this.

2.8.4 PPP Dial-in and Dial-out

The `pppd` program works as both PPP server and client on the SIB. The `pppd` program may be invoked in one of two ways, both of which are simplified by scripts. The PPP protocol was designed to use dynamic IP addresses. This means that the PPP client (usually the machine doing the dialing) gets its IP address and interface configuration from the PPP server (usually the machine being dialed) before any IP packets are exchanged.

To use the SIB as a PPP client, you must first create a dial-out script, and then execute it. Dial-out scripts should be created and executed using the main system configuration menu. These scripts are stored in the `/etc/ppp/` directory. These scripts can also be executed by typing the name of the script on the command line, just like any other program. PPP dial-out scripts may also be invoked by the CRON program, so that the SIB can connect to the Internet via phone at specified times and upload collected data to a central server, for example.

PPP dial-out scripts will dial a phone number, enter a username and password, and then optionally enter a command to start `ppp` on a remote machine. If further interaction with a remote server is required, an experienced Linux user may edit the `/etc/ppp/ppp-on-dialer` script.

The script which invokes `pppd` as a server is `/usr/bin/ppp`. This script will start the PPP server on any login terminal, including the console and telnet sessions. However, it is generally only useful when started on a dial-in terminal. To use the PPP server in the normal sense (as you would dial your ISP from home), you must first set the SIB's modem to answer the phone as a dial-in terminal. Once the dial-in terminal connection has been established, you must log in as a non-root user (usually www). After logging in, a PPP session may be started by simply typing "`ppp`" (no quotes) on the command line. Dial scripts on the client machine usually automate the process of logging in and starting PPP. Once PPP has been started, unreadable character strings will begin to appear in the dial-in terminal which is PPP data. At this point you should instruct the client machine to begin a PPP session.

When acting as a PPP server, the SIB needs two IP addresses assigned to a PPP interface before it is started – a local address which the SIB will use for the PPP interface, and a remote address to assign to the calling, client machine. For simplicity and logical reasons, these IP addresses are associated with serial devices rather than with network interfaces. These IP addresses are stored as `/etc/ppp/options.ttyS*`. If using an SIB with the modem package, the modem is on ttyS2 by default, causing IP addresses will be stored as `options.ttyS2`. Alternatively, an external modem may be used, in which case the file would correspond to the device that the modem is connected to. The format for these addresses is `<local IP>:<remote IP>`. An IP address of `0.0.0.0` means that PPP should try to get the address from the remote host, rather than assigning an IP address to the remote host.

Since these IP addresses are assigned to serial devices and the program works as both PPP server and client, `pppd` will use the addresses whether it is acting as a server or a client. That is, if `options.ttyS*` specifies IP addresses, then `pppd` will attempt to use those addresses even if it is dialing out. This can lead to problems if the remote computer is not expecting to have its IP address assigned to it. While this can be difficult to manage if a modem will be used for both PPP dial-in and dial-out, it also allows for a very useful feature: dial-on-demand.

Dial-on-demand is a feature whereby a PPP interface may be configured and activated without dialing the phone. The interface will sit idle until the SIB's network subsystem is given a packet that must be transmitted to the remote host on the PPP interface that is using dial-on-demand. The SIB will then dial the number, make the connection, and transmit the data. After a certain amount of time without PPP traffic, the SIB will hang up the phone and again wait for outgoing PPP traffic.

To use PPP dial-on-demand, two things must be satisfied. First, a dial-out script must be created. Second, the modem device must have at least a remote IP address assigned to it. If the SIB does not know the remote address on the PPP link, it cannot know when to activate the link by dialing the modem. The local IP address may be specified, or it may be retrieved from the remote host after dialing. To use dial-on-demand, the "demand" option in the global PPP options file (`/etc/ppp/options`) or the interface specific PPP options file (`/etc/ppp/options.ttyS*`) must be set. You may use the main menu configuration utility to configure this. However, the phone will not dial until IP traffic addressed to the remote PPP IP address is encountered.

2.8.5 HTTP / Web Server

The SIB's web server is lightweight, single-tasking HTTP server named "Boa" (`/usr/sbin/boa`). In testing, Boa has been shown to be quite fast, able to serve hundreds of requests per second on a 486-class machine. Since it is single tasking, Boa also uses much less memory than some more standard HTTP servers (such as Apache). Boa also takes comparatively little flash disk space. The trade-off is that Boa does not have many of Apache's advanced features, such as ultra-fine-grained access control, or SSL encryption. However, Boa has proven sufficient for most SIB tasks.

Boa is started at boot time by default. The script `/etc/init.d/netstd_init` starts both Boa (http) and the FTP server (`ftpd`). If this is not desired, it may be changed in two ways. First, the link `S03netstd_init` may be deleted from the `/etc/rc2.d` directory. This will prevent the `netstd_init` script from being executed when run level 2 (the default) is entered. Second, you may edit the `/etc/init.d/netstd_init` script by hand. Placing a '#' (no quotes) in front of the line which starts that http server will prevent the server from being started.

Boa's configuration is read from the `/etc/boa/boa.conf` file. This file contains directives for the TCP port, user and group to run as, logging, and other configurations. An HTTP server usually runs on port 80; it is not recommended that this be changed. Once the server starts, it runs as user "`www-data`" and group "`www-data`." This is done for security reasons: the user www-data and group www-data have no special permissions anywhere on the system. Therefore the HTTP server cannot read files that a non-privileged user should not, nor can it change file (via CGI, perhaps) that it should not.

To save RAM disk space, no logging is done. If you are having problems with the web server, logging may be enabled for both errors and HTTP accesses. The configuration file also allows you to set certain paths, directories, and file types. File types (MIME types) may be associated with file extensions both in the HTTP configuration file, and in the `/etc/mime.types` file. The paths tell the HTTP server where to find CGI (Common Gateway Interface) and other files. In particular, Boa uses a CGI `/usr/lib/cgi-bin/boa_indexer`. This CGI generates directory listings on the fly when no `index.html` file is available. The `boa.conf` file is also somewhat self-explanatory; see the comments in the file for more information.

One important setting in the `boa.conf` file is the "document root." The document root is the directory that Boa considers the top-level directory; no higher levels can be accessed. For example, the default server root is `/home/www`. When a browser is given the URL `http://10.0.2.41/` (assuming that the SIB is still at the default address `10.0.2.41`,) the contents of the `/home/www` directory will be displayed.

Note that Boa is a stand-alone server, which does not use the `inetd`. Therefore, `inetd` access control may not work when using Boa. See the links in section 4 for more information regarding Boa.

2.8.6 FTP Server

The SIB's FTP server is the Linux-standard WU FTPD. This server is started by default at boot by the same scripts that begin the HTTP server, and it may be disabled in the same manner. WU-FTPD (`/usr/sbin/in.ftpd`) is run through `inetd`, thus all `inetd` access control methods will work as previously described. The file `/etc/ftpaccess` is the master `ftpd` configuration file. The files `/etc/ftphosts`, `/etc/ftpusers`, and `/etc/ftpgroups` all configure access control for the FTP server. The syntax of these files remains beyond the scope of this document. See the links in section 4 for more detailed information. The SIB's FTP server uses the default ports 21 (for commands) and 20 (for data).

2.8.7 TELNET Server

The SIB's TELNET server (`/usr/sbin/in.telnetd`) is also the Linux standard. The TELNET server is run through `inetd`, thus all access control methods previously described will work. Further access control is accomplished by the same methods as limiting terminal access to any user – the username / password pair, as well as the `/etc/securetty` file. See

the manual pages on your development workstation for more details. The SIB's TELNET server runs on the default TELNET port 23.

2.8.8 SSH Server

The SIB also runs an SSH server by default, to provide encrypted connections and tunnels over TCP/IP connections. The SSH server is the standard OpenSSH version 3.8.1. Access control for the SSH server is not done through `inetd`, so other methods must be used to control this, such as the `hosts.allow` and `hosts.deny` files (described in section 2.8.3). Also, the file `/etc/ssh/sshd_config` can be used to control access to the SSH server and configure its operation. See the links in section 4 for more information concerning OpenSSH.

2.9 Time Zones

Due to its size and space constraints, the base SIB does not support locales and time zones in the same manner as full-sized Linux distributions. It does not include the database of time zones that are found on other Linux distributions. Although the SIB can be configured to use local time and time zones, there is no distinction between the system clock and hardware clock; the two are always synchronized when the system is shutdown and when the system is booted.

The environmental variable `TZ` stores the current time zone, which is used by the system when determining and printing the time. This variable is set in the file `/etc/default/timezone`, and is accessed during shutdown and boot up. Because the SIB does not store time zone information for available time zones in a database, it cannot determine the correct local time solely from the time zone name. If you will be setting the time manually (through the menu system or by using the date command), the time zone may be specified simply using the abbreviation for the local time zone. If you will be setting the clock using an automatic update service through the rdate command, however, it is necessary to specify an offset for the sytem to use. In order to do this, it is necessary to specify the offset from Universal Coordinated Time (UTC) in the time zone, and create a POSIX.1 `TZ` variable from this information.

There are two parts to the POSIX.1 specification for `TZ` variables. The first form is the simplest, and can be used when there is no Daylight Savings Time or "summer time" in the current time zone. It is specified using only a time zone identifier and an offset. The identifier is at least three letters long, and can not contain a leading colon, digits, commas, or plus or minus signs. Examples of this are CST for Central Standard Time or EST for Eastern Standard Time. Generally, these abbreviations will be all capital letters such as CST and EST. The system does not use this identifier to determine the time, so you may use any identifier that fits the POSIX.1 specification so that it may be parsed by the system.

The offset specified is the amount of time that must be added to the local time to get UTC. This value will be positive if the time zone is to the west of the Prime Meridian and negative if it is to the east of the Prime Meridian. It is constructed using the form `[+|-]hh[:mm[:ss]]` (terms in `[]` are optional). For example, EST is 5 hours behind UTC, so it could be specified using an offset of `+5`. To specify a value of 10 hours, 23 minutes, and 34 seconds, the offset would be `+10:23:34`. Putting the identifier and offset together, you will have a specification of something like: `EST+5` (with no Daylight Savings Time information).

The `TZ` variable may also contain information regarding Daylight Savings Time. This is added onto the end of the standard time zone information and offset already constructed, and is of the form `dst[offset],start[/time],end[/time]`. `dst` is the identifier string for the Daylight Savings Time zone. The offset describes how far the standard time (specified by the identifier and offset) is changed when Daylight Savings Time is in effect (if ommitted, this defaults to 1 hour). The `start[/time]` and `end[/time]` sections specify when Daylight Savings Time is applicable.

Start and end specify the day of the year, and can be specified by one of three methods: `J`*`n`*, which specifies the day of the year from 0-365, never counting February 29; *`n`*, which specifies the day of the year, counting February 29 in leap years; and `M`*`m.w.d`* which specifies the month (`m`) (in the range 1-12), the week of the month (`w`) (1-5), and the day of the week (`d`) (0-6). The time field specifies at what (local) time Daylight Savings Time takes effect. If not given, it defaults to 2:00:00 (2:00 AM).

The GNU C library gives the following example of a specification for EST in the US:

```
EST+5EDT,M4.1.0/2,M10.5.0/2
```

Note that the most secure method is to set the clock to use UTC. Also, in order for time zone changes to take effect, the SIB will need to be restarted in order to set the `TZ` variable system wide, and synchronize the hardware and software clocks.

3 Configuration

The EMAC SIB provides a complete menu-based configuration utility that allows for configuration or displaying information about serial ports, network devices, network routing, hostname resolution, CRON scheduling, default system configuration, user account maintenance, and build information. This menu system is activated automatically anytime the root user logs in. Also, root may execute the menu system from the command line by typing the command `menu`, which is a link to the file `/etc/sibconfig/SIB-Config`, the main configuration menu script.

3.1 Getting Started

The menu based configuration system can be reached through the console, a TCP connection via SSH or TELNET, or via a serial terminal. A connection to the SIB via one of these options must be established before it can be configured. Connecting through the console is possible if the SIB has a video interface (SIBs can be purchased "headless" with no video). This is what would be thought of as the standard way of connecting to a computer. Simply plug a VGA monitor and keyboard into the appropriate connectors on the SIB and power the SIB on. The SIB should finish its boot process and present you with its login prompt.

To log in via a serial terminal, a "dumb terminal" may be attached to the SIB'S primary serial port (`COM1` or `ttyS0`). A standard PC running a terminal emulator such as HyperTerminal will also work provided a null-modem cable is used. The terminal settings are: 9600 bps, 8 data bits, 1 stop bit, no parity, no flow control, and vt100 terminal emulation. Begin by connecting the terminal to the SIB, powering on the SIB, and waiting for the SIB to finish its boot process. If necessary, press the enter key a few times to "wake up" the terminal. You should then be presented with a login prompt.

There are several options available to establish a connection via TCP. To establish a direct connection with the SIB, connect the Ethernet port on the SIB to the Ethernet port on a PC using a point-to-point (crossover) Ethernet cable. Alternatively, the SIB and a PC could be connected to an Ethernet hub or switch with no other devices attached.

By default, the SIB is set to attempt a DHCP configuration of its Ethernet connection. If your network has a DHCP server, simply plug the SIB into the network via its Ethernet port. The SIB will then attempt to lease an IP address. Find out which IP address the SIB has been leased, and connect to the device using SSH or TELNET.

If your network does not have a DHCP server, the SIB will fall back to its default settings: IP address `10.0.2.41`, network mask `255.255.255.0`, broadcast `10.0.2.255`, and default gateway `10.0.2.1` on interface `eth0`. (For future reference, these defaults are stored in `/etc/default/network/eth0defaults`.)

If DHCP is not available, you will need to configure your PC to communicate directly with the SIB for the initial configuration. Under Windows, you will need to change your default gateway to `10.0.2.41`; the SIB's factory set address. Under GNU/Linux, simply add a routing entry for `10.0.2.41` on the appropriate interface using the route command. You may then connect to the SIB at address `10.0.2.41` via SSH or TELNET. See section 2.5 for more information on this.

Once you are connected to the SIB, log in as `root`. A password may have been provided with the SIB. Otherwise, use the password `emac_inc`. As soon as the password is accepted, the main menu configuration utility will be displayed on the screen.

3.2 Main Configuration Menu

The main menu is used to navigate all of the secondary menus and single configuration options. The eleven options on this menu provide extensive and efficient access to settings and configuration for the SIB as a whole. Figure 1 shows a screen shot of the main menu. (Note that the color scheme may differ from that of Figure 1 depending on the connection type and environment.)

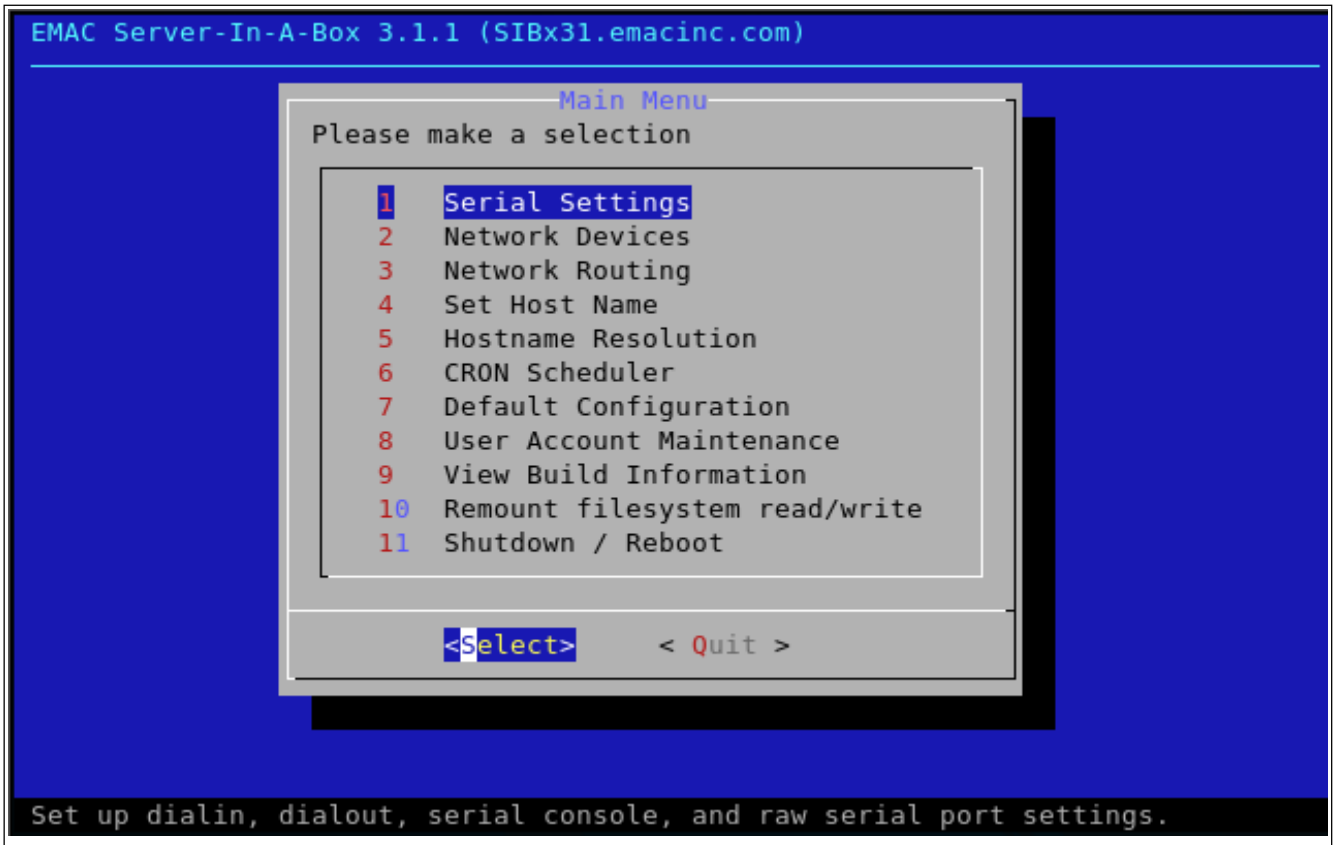

Figure 1: Main Menu Configuration System

Operation of the menu is a simple task. To navigate the menu, use the arrows on the keyboard. Typing a particular number will also highlight the respective entry. Pressing the left or right arrow will move the cursor between *Select* and *Quit*. To select an option, highlight the correct line, check that *Select* is also highlighted, and press enter. Pressing enter when *Quit* is highlighted will exit the menu, regardless of what option is selected. To restart the menu system after quitting, type `menu` at the shell prompt. The caption at the bottom of the screen displays a description of the current option. All subsequent menus operate in a similar manner. (Note that the caption is not present on all sub-menus.)

The following is a quick description of the menu options:

1. The *Serial Settings* option allows the user to configure raw serial settings, serial text terminals, dial-in terminals, and PPP dial-in and dial-out connections on the available serial ports.
2. The *Network Devices* option allows the user to configure Ethernet settings, delete a saved network device configuration, or reset a device to use DHCP.
3. The *Network Routing* menu option provides methods to set up, delete, and view network routes, including the default gateway.
4. The *Set Host Name* option simply allows the user to change the host name of the SIB.
5. *Hostname Resolution* changes and displays many settings including the domain name, nameservers, search domains, and static hosts. These settings allow the SIB to resolve IP addresses from host names and visa versa.
6. The *CRON Scheduler* option brings up a new window with the CRON configuration file (discussed later) using the default editor so that CRON tasks may be configured.
7. *Default Configuration* allows the user to change settings such as the time and date, timezone, root filesystem properties (read/write or read only) on boot, edit kernel modules, and change the default editor.
8. *User Account Maintenance* provides options to add and delete users and groups, add and delete users to and from existing groups, and change the root password.
9. *View Build Information* simply displays the build reference information such as kernel version, Libc version, and distribution information.

10. The *Remount Filesystem read/write* option allows the user to override the default settings of mounting the filesystem as read/write or read only for the current session. If the filesystem were mounted read/write when the menu was started, this option would read *Remount Filesystem read only* instead.
11. *Shutdown / Reboot* allows the user to shutdown or reboot the SIB.

Each menu is explained in detail in the subsequent sections.

3.2.1 Serial Settings

The *Serial Settings* option on the main menu will bring up the *Serial Configuration Menu*. The options for this menu are shown in Figure 2. Selecting *Back* on this or any other menu will return you to the previous menu, in this case the main menu. (*Back* will return you to the most logical window depending on what is being configured. This is generally the previous menu.)

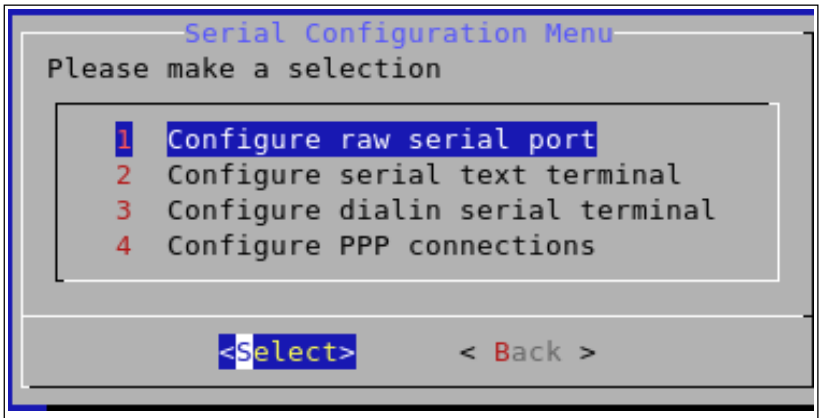

Figure 2: Serial Configuration Menu

There are four options on the *Serial Configuration Menu*:

Configure raw serial port

This menu option allows users to set and view parameters of the serial ports such as I/O port, UART, IRQ, and baud rate. Selecting this option will bring up a second menu with options to *View current properties*, *View saved settings*, or *Configure raw serial ports*. The serial port configuration is done through a utility called `setserial`, which interfaces with the Linux kernel to provide information about the serial ports. Any manual configuration is saved in the file `/etc/serial.conf`, which is used by `setserial` to override the system defaults at boot time.

The *View current properties* and *View saved settings* options differ in that viewing the current properties of the serial ports simply reports the information that the system has regarding its serial ports, while viewing the saved settings of the serial ports will allow you to see any custom configurations that are being used by the system from the `/etc/serial.conf` file. If none of the settings under *View current properties* are wrong, configuration should be unnecessary.

The *Configure raw serial ports* option also has a sub-menu, with options to *Configure raw serial device* or *Delete raw serial device*. Selecting *Delete raw serial device* will bring up a menu prompting the user for the entry to delete from `/etc/serial.conf`. Select a line to delete it, or return to the raw serial device configuration menu by selecting *Back*. Upon selecting *Ok*, you will be prompted to confirm that the correct entry is being deleted. After confirming this, a window showing the saved settings of the serial devices will be brought up. Check that the proper line has been deleted.

Selecting *Configure raw serial device* will bring up a menu prompting you for the device that you would like to configure. Select a device and press return, or select *Back* to exit this menu. A new menu will be brought up to allow you to create a custom entry for the `/etc/serial.conf` file. If the device that you are looking for is not recognized by the system, select *Other* to enter the device number manually. All of the information for this device must be known prior to configuring it.

This new window allows the user to set the port, UART, IRQ, baud, and flags for the selected serial device. The existing values for the device are used as the default values. To change the values for the device, highlight the setting that you would like to change. Next, use the arrows to highlight *Edit* at the bottom of the box and press enter. Use the keyboard to enter the

new value, and press enter when you are finished. You may edit any other settings also. When you are finished making changes, make sure that the information is correct, highlight *Ok*, and press enter. The flag settings determine properties of the device configuration, and generally should not be changed. (Note that entering incorrect settings can lock up your machine!) To exit the menu without making any changes to the system, select *Back* and press enter.

After you have committed your changes, you will be prompted to confirm that the correct settings are being written to `/etc/serial.conf`. After confirming that the settings are correct, a window showing the current settings of the serial ports will be brought up. Check that the changes that you have made were accepted by the system and that the device has been configured properly.

Configure serial text terminal

This menu option will allow you to set up serial text terminals on any of the serial devices on the SIB. After selecting this item, you will be given choices to *Add serial text terminal*, *Delete serial text terminal*, or *View current settings*. Serial text terminals are configured in the `/etc/inittab` file.

To add a text terminal, select *Add text terminal* from the menu and select the device that you would like to configure from the list. You will then be prompted to enter the baud rate and terminal emulation setting for the port. This menu operates in the same manner as the one described for setting raw serial ports. To edit an entry, select the field to change, move the cursor to *Edit*, and press enter. Select *Ok* and press enter to commit the changes.

Once you have selected *Ok*, you will be prompted to confirm that the correct settings will be written to the `/etc/inittab` file. After confirming that the settings are correct, the configuration will be saved.

Selecting *Delete text terminal* will allow you to select the serial text terminal configuration to delete, if any exist. Upon selecting an item to delete, you will be prompted to confirm that the correct configuration is being deleted. Select *Yes* to delete the configuration.

Selecting *View current settings* will display all configured serial text terminals that exist on the system.

Configure dialin serial terminal

The *Configure dialin serial terminal* menu option will allow you to set up a terminal server that is modem-aware on the specified serial port. A serial device is specified as with previous menu items. When a serial port is configured this way and a modem is attached to it (or if the port is an internal modem), the SIB will answer the phone when it rings and attempt to make a data connection with the caller. If a data connection is made, the SIB will then prompt the caller for a login and password (in plain text), and the caller may log in as if they were using a standard terminal or the console.

To configure a serial dialin terminal, select *Add dialin terminal* from the *Dialin Terminal Configuration Menu*, and select the port to add the dialin terminal on. If the serial device currently has a conflicting configuration (such as serial text terminal), a message will be displayed asking you to delete the existing configuration or abort the current operation. After deleting any existing configuration, you will need to specify the line speed of the serial port. Enter the desired line speed and press enter to add the new configuration. To cancel the configuration, press the tab key and use the arrow keys to move the cursor from *Ok* to *Cancel*.

The rest of the menu options (*Delete dialin terminal*, and *View current setup*) operate in the same manner as described in the *Configure serial text terminal* section above.

Configure PPP connections

Configure PPP connections allow the user to configure dialin and dialout PPP connections. After selecting this menu option, a new menu will be brought up with options to *Configure dialin PPP connections*, *Configure dialout PPP connections*, *View active global PPP options*, and *Configure global PPP options*.

The *Configure dialin PPP connections* options sets up a PPP server on a serial port so that the server can be dialed from another computer and PPP connections can be established. Upon selecting this menu option, a menu will be displayed with options to *Add dialin PPP*, *Delete dialin PPP*, and *View current settings*.

To configure a dialin PPP connection, select *Add dialin PPP* and choose the correct serial port to use for the dialin PPP server. This will allow you to set the remote and local IP addresses for the connection. Use the *Edit* option to edit an entry and the *Ok* button to commit changes. If the calling client is to assign either or both IP addresses, then those addresses should be entered as `0.0.0.0`.

After entering the IP addresses and selecting *Ok*, you will be prompted for confirmation that the correct settings are being written. Following this confirmation, if the server has not yet been configured as a dialin server (a prerequisite for a dialin PPP server), you will be notified that this must be configured. The configuration for the dialin PPP server will still be written regardless of the existence of a dialin terminal server configuration.

To start a PPP link, dial the modem on the selected serial port and log in as the `www` user. When presented with a command prompt, simply type `ppp`. This will begin the PPP connection. Note that most PPP-aware software attempts to automate this login process. Check the documentation for your client software for more information.
The *Delete dialin PPP* and *View current setup* options operate exactly as in the previous serial configuration options.
Selecting *Configure dialout PPP connections* provides the user with options to *Add dialout PPP*, *Delete dialout PPP*, *Change boot-time dialing*, *Manually dial connections*, and *View current settings*.

To add a dialout PPP connection, select *Add dialout PPP* from the menu and enter a name for the connection (the name should not contain any spaces). This will create a custom configuration file for the selected serial port. After entering a name for the connection, you will be prompted for all of the information for the particular connection. It is important that all of the information entered is correct, so contact your ISP if you are unsure of some of the settings. Once you have entered all of the settings on this menu, you will be prompted to enter the password to use for the connection.
The *Delete dialout PPP* option will allow you to select the configuration to delete from existing configurations.
If you would like connections to be dialed automatically when the system boots, you can configure this through the *Change boot-time dialing* menu. Use the space bar to toggle the check marks in front of the connections that should be dialed automatically when the system boots.

In order to dial a connection, select the appropriate connection to dial from the *Manually dial connections* menu.
The *View current settings* menu option will display a list of the currently configured dialout PPP connections. Select one of these connections to view the specific settings for that connection.

Moving back to the *PPP Configuration Menu*, two more options remain. The *View active global PPP options* menu item will display a list of the PPP options from the file `/etc/ppp/options` that have been enabled. To edit this file manually, select the *Configure global PPP options* menu option.

3.2.2 Network Devices

The *Network devices* menu option on the main configuration menu will allow you to change the settings of the network devices on the system, such as the IP addresses, network masks, and other settings. The kernel assigns interface names and numbers at boot time in the order that they are configured. For Ethernet devices, this will be `eth0`, `eth1` and so on. The *Network Configuration Menu* has three options:

Manually configure network device

Use this option to statically configure a network device to use the same settings every time that the device is started by the system. The default configuration for network devices is DHCP.

After selecting this option, you will be required to identify the network device that you would like to configure. After entering the device, you will be required to input the desired IP address, network mask, and broadcast address for the device. If the device has already been configured and is active, the current values of the device will be used as the default values in the menu field. Select *Edit* to edit a field and *Ok* to save the configuration, as in the menus for serial device configuration. Once you have pressed *Ok*, a window will appear that will remind you to add a default gateway for the device. Press *Ok*. The next window will prompt you to reset the network devices now, or continue without restarting network devices. If you are connected through a network connection, it is possible that resetting your network devices will cause you to lose your connection and be forced to reconnect. If the settings that you have changed are incorrect, it may be possible to lock out network access to the SIB. It is safe to restart the devices if you are sure that the settings that you have written are correct, and you have not changed the connections for the device that you are connected on.

After resetting the network interfaces or rebooting the SIB, the changes that you have made will take effect on the network device.

Reset network device to DHCP

Reset network device to DHCP will allow you to overwrite static configuration settings that exist for a particular network device or create a new configuration for the network device so that the system will attempt to use DHCP to configure the settings of the interface.

After selecting this menu option, you will be prompted to enter the name of the network device to reset. Make sure that you enter the correct device name to prevent from overwriting existing settings for other interfaces. Once you have entered the device name, a message will be displayed noting which device the system will reset. You will then be prompted to restart all network devices or continue without restarting the network devices. See the description of this option under the *Manually configure network device* section.

Remove network interface

The *Remove network interface* option is the final item on the *Network Devices* menu. This will allow you to delete any custom configuration, both static and DHCP, for the network devices on the system. This is necessary when the system is attempting to bring up interfaces that either no longer exist or should not be initiated under the current environment.

Upon selecting this option, you will be prompted for the name of the device to delete. You will then be prompted for a confirmation to delete the configuration file for the device. Select *Ok* to delete the device configuration file. If you choose to delete the interface, you will once again be prompted to restart the network interfaces or restart the interfaces later. See the section on *Manually configure network device* for more information on the implications of restarting the network devices.

3.2.3 Network Routing

The *Network Routing Configuration Menu* provides options to add a default gateway, add an IP route, delete an IP route, view the IP routing table, and view custom IP routes. Routing entries are stored in a custom script, which is automatically run every time the system boots. Most of the routing functions are not necessary for DHCP addressing configurations.

Add a default gateway

This menu option allows the user to manually add a default gateway entry into the routing table. This requires that the default gateway be in the same subnet as the SIB and that only one gateway should be added per network device.
After selecting this option, the user is required to input the IP address or DNS address of the default gateway. If a value has already been set for the default gateway, that value will be used as the initial value for the input menu. Enter the correct value and press enter. The SIB will then set the default gateway to the address entered in the menu, and add an entry to the custom routing script `/etc/sibconfig/network/route.sh`.

An information box will be displayed indicating the new value for the default gateway as well as the route command that was added to the script.

Add an IP route

This menu option allows the user to add an IP route to the routing table for an IP host or network of hosts. After selecting this option, you will be allowed to choose between adding a route for a host or net. Choose the appropriate value and press enter. You will then be prompted for the IP address, network mask, and interface to use to contact the host or net. Enter the correct values and select *Ok*. Navigate this menu using the *Edit* button and arrow keys as in all of the previous similar menus.

Once you have completed the settings for the host or net, you will be prompted for confirmation to add the route to the permanent routing script. Check that all of the settings are correct and select *Yes* to add the route.

Delete an IP route

This option is provided to delete entries from the custom routing script so that the routes are not executed every time the system restarts. After deleting the lines from the script, this menu will also execute commands to delete those routes from the current kernel routing table.

After selecting this option, a menu listing the custom route commands will be displayed. Below this listing, you may enter the address of the routing entry to delete. This entry will be used by the configuration system to search for a matching route to delete within the list of routes. For this reason, please ensure that this entry is a string that is unique to the entry that you would like to delete to prevent from deleting multiple routes at once.

After entering an address to delete, the system will prompt the user for confirmation that the correct route was located and will be deleted. Select *Yes* to delete the selected route.

View IP routing table

This menu option displays the current contents of the kernel routing table on the SIB from the output of the "`route`" command. Check that all routes are correct and press *Ok* to return to the routing menu.

View custom IP routes
This menu option simply displays the custom routing entries that are saved in the custom routing script `/etc/sibconfig/network/route.sh`. Entries in this file are executed every time that the system boots.

3.2.4 Set Host Name

The *Set Host Name* menu allows the user to change the current host name of the SIB. This host name must be a valid network identifier to be recognized by the network that the SIB is running in. By default, the host name will be set to SIBx3* (* is the current version number) when delivered from EMAC.

To set the host name, select this menu option and enter the desired host name in the new window that is brought up. The default value for this field will be the current host name of the SIB. Press *Ok* to set the host name to the newly entered value.

3.2.5 Hostname Resolution

The *Hostname Resolution* menu allows users to change the domain name of the SIB as well as add and delete nameservers, search domains, and static hosts. The menu also provides functions to view the resolver information, static hosts information, and save the resolver information as default.

Change domain name

Once selected, the *Change domain name* option will start a new menu with the ability to enter a new domain name to use for the SIB. The default initial value for this field is the current value of the domain name as reported by the system.
Enter a new domain name if necessary and select *Ok* to commit your changes. After changing the settings, a message dialog will be displayed detailing the new domain name for the system.

Add nameserver

The *Add nameserver* menu item will allow you to add a new nameserver to the current list of nameservers used by the system. After selecting this option, enter the IP address of the nameserver to add and press enter. This will add a new nameserver entry to `/etc/resolv.conf`, the system-wide resolver configuration file.

Delete nameserver

To delete a nameserver entry from `/etc/resolv.conf`, select the *Delete nameserver* option. This will initiate a new menu listing all of the nameserver entries currently available to the system. Use the cursor to select the item that you would like to delete, highlight *Ok*, and press enter to delete the item. A new menu will then confirm that the nameserver entry has been deleted.

Add search domain

The *Add search domain* menu option operates in the exact same way as the *Add namesever* option. Enter a search domain (i.e. `mydomain.com`) to add it to the `/etc/resolv.conf` file.

Delete search domain

The *Delete search domain* menu option operates in the same manner as the *Delete nameserver* menu option. Select one of the entries from the list in order to delete it.

Add static host

Add static host allows you to add an entry to the `/etc/hosts` file so that the SIB explicitly associates a particular IP address with a corresponding host name. The first screen on this menu will prompt you for the IP address for the host to add. Enter the IP address and select *Ok*.
Next, you will be prompted for the host name of the host, followed by any aliases (alternate names for local use on the SIB) separated by spaces. Enter the host name and any other aliases and select *Ok* to commit the changes.
The menu system will create a new entry for the host and add it to the `/etc/hosts` file. A new window displaying the entry created will be brought up before returning to the host name menu.

Delete static host

This menu option operates in the same way that the *Delete nameserver* item operates. Select an entry from the menu that is displayed to delete it.

View resolver information

The *View resolver information* menu item simply displays the current resolver information used by the system from the file `/etc/resolv.conf`. All changes made to resolver information (domain name, nameservers, and search domains) will be registered in this view.
Check that all of the information is correct, and press enter to return to the hostname resolution menu.

View static host information

The *View static host information* option allows the user to view the contents of the `/etc/hosts` file containing the information that the system uses to identify static hosts.
Check that all information is correct, and press enter to return to the hostname resolution menu.

Save resolver information as default

Use this menu option to copy the current contents of the resolver information in `/etc/resolv.conf` to the default values in `/etc/default/resolv.conf`. This setting is only necessary when attempting backup the current information during testing or for static IP address configurations.
After selecting this option, a message will be displayed explaining the purpose of this option and allowing the user to save the current settings as default or to continue without saving the current settings.

3.2.6 CRON Scheduler

This menu option allows the user to directly edit the configuration file for CRON (`/etc/crontab`), the periodic command scheduler. The default editor as set in the *Default Configuration* menu will be used to edit the file. More information about how to use the editors is covered in section 2.2, with information on setting the default editor in section 3.2.7. You will find more information on CRON and its operation in section 2.6. Upon exiting the editor, control will return to the main menu.

3.2.7 Default Configuration

The *Default Configuration* menu provides options for user preferences and settings such as setting the time, date, and time zone, changing the read/write properties of the filesystem, editing kernel modules, and changing the default editor.

Configure time/date

This menu option allows the user to easily set the time and date from a convenient user interface. Upon selecting this option, a window will be brought up announcing the current system time and date. Select *Yes* to edit the system time and date, or *No* to return to the *Default Configuration* menu.

After selecting *Yes*, a calendar will be displayed with the current date according to the system clock highlighted. Use the arrows on the keyboard to navigate the calendar until the correct date has been highlighted. Press tab to be able to move the cursor between *Ok* and *Cancel*. Pressing tab multiple times will change the field that is being edited between day, month, and year. Select *Ok* and press enter to select the date.

The next window will display a clock showing the current system time in hours, minutes, and seconds. To modify the time, use the arrows to highlight the field that you would like to edit (the tab key may also be used for this), and use the up and down arrows to change the values. Move the cursor to *Ok* and press enter to set the system clock to the given time.

After setting the clock, a new window will be displayed to show the new system time and date. Press *Ok* to return to the *Default Configuration* menu.

Change time zone

The *Change time zone* menu item will allow the user to change the current time zone of the system. In order to specify Universal Coordinated Time (UTC), enter UTC as the time zone. The time zone must be specified as a valid `POSIX.1 TZ` variable, described in depth in section 2.9.

When this menu item is selected, the current time zone setting will be displayed with a message prompting the user to change the time zone or continue with the current time zone. Select *Yes* to change the time zone. This will allow you to enter the desired time zone into the menu and have it saved as default by the SIB. It is necessary to reboot the SIB after changing the time zone to allow the new time zone to be recognized by the system, and to synchronize the hardware and software clocks. Note that the most secure method is to set the clock to UTC.

Change root filesystem to read/write

This menu option toggles the default setting of the root filesystem so that the SIB is booted in read/write mode rather than read only mode. If the current setting is to boot read/write, then this menu item will read *Change root filesystem to read only*. Selecting this option creates or deletes the file `/etc/READONLY`, which the system uses to determine how the filesystem is mounted at boot time. Note that booting the system in read only mode causes the `/dev` directory to be mounted as a ramdisk, which forces any changes made to device nodes to be temporary for this session only. When the system is rebooted, these changes will be lost. Booting the system in read/write mode, however, will allow changes to the device nodes to be preserved through reboots.

It is advantageous to mount the filesystem in read only mode whenever changes are not being made, as this prevents damages to the filesystem in the case of a power outage or similar event. System configuration must be done in read/write mode, however.

Simply select this option to toggle the properties of the filesystem. This will display a message reflecting the new setting as well as the implications of mounting the system in this way.

Edit kernel modules

The *Edit kernel modules* menu option allows the user to edit the kernel modules that are loaded at boot time by directly editing the file `/etc/modules`. The default editor set in the *Default Configuration* menu will be used to edit the file. Refer to section 2.2 for more information on how to use these editors.

Exiting the editor will return control back to the *Default Configuration* menu.

Change default editor

This option changes the editor used to modify files directly from the menu system. The default setting for this is `/bin/vi` when shipped from EMAC. The other editor available on the system is `/bin/nano` for the base EMAC Linux build. Enter the absolute path (`/bin/vi` or `/bin/nano`) into the menu and press enter to save the setting.

3.2.8 User Account Maintenance

The *User Account Maintenance* menu allows the user to add and delete users and groups, add and delete users from groups, and change the root password.

Add user

Use the *Add user* menu item to add a new user to the system. First, a new window will be brought up that prompts for the user name to add. Enter a new user name and press *Ok*. (Note that this should be a valid user name and should include no spaces.)

Next, you will be prompted to enter a password for the user. Enter a password and select *Ok*. (Note that you will not be able to see the password as it is typed.) The system will then prompt for verification of the password. If the two passwords entered match, a new menu will be used to record all of the information for the user, including home directory, shell, default group, and additional groups to add the user to. Complete the information and press enter to create the user. The system will confirm that the user has been added.

Delete user

This menu simply prompts for the user name of a user account to be deleted from the system. If the user exists, they will be deleted from the system, but their home directory must be manually removed. The system will confirm that the user has been deleted.

Add group

This menu prompts for a group name to be added to the system as a user group. If the group does not already exist, it is added. The system will confirm that the group has been added.

Delete group

This menu prompts for a group name to be deleted and deletes the group if it exists. Before deleting the group from the system, the user must confirm that the correct group is being deleted. The system will confirm that the group has been deleted.

Add users to group

The *Add users to group* menu option allows existing users to be added to existing groups on the system. The system prompts for a group name followed by a user name and adds the user to the group, if they both exist. The system will confirm that the add was successful before returning to the *Default Configuration* menu.

Delete users from group

The *Delete users from group* item allows the user to remove a user from a specific group. The user is first prompted for the group name, followed by the user to delete from that group. If both the user and the group exist on the system and the user is a member of the group, the user will be removed from the group. The configuration system will display a confirmation that the deletion was successful.

Change root password

This menu option will allow the root password to be changed on the system. This should be done as soon as possible after the SIB is setup and accessible. Choose a secure password for the root password, and keep it secret. If a malicious (or reckless) user were to have access to the root account, the system could be rendered unusable. If the root password is lost or forgotten, contact EMAC support for information on how to reset the password.
After selecting this menu option, you will be prompted for the new root password twice. If the passwords are identical, the root password will be changed. The system will confirm that the password change was successful and return to the *Default Configuration* menu.

3.2.9 View Build Information

The *View Build Information* menu option provides an easy way to view information about the custom build and operating system. Items included on this menu option include:

- Build reference – the EMAC part number including information about the custom modules and packages that are included.
- Distribution – the EMAC Linux distribution type and version number.
- Kernel – the Linux Kernel version number with EMAC naming conventions.
- Libc version – the version of libc (glibc), the standard C libraries that are included on the build.

3.2.10 Remount Filesystem read / write

Remount Filesystem read/write changes how the filesystem is mounted after the menu system is exited for the current session only. The root filesystem is mounted read/write while the menu system is running, regardless of the properties that it was mounted with before the menu system was executed. This menu option will allow you to override this behavior by toggling the way the system is mounted. If the system was mounted read/write when the menu configuration system was started, this option would read "Remount Filesystem read only", and would cause the filesystem to be remounted read only upon exiting the menu system. The filesystem will be mounted with the system defaults (read only or read/write) as set in the *Default Configuration* menu when the SIB is rebooted.

3.2.11 Shutdown / Reboot

The *Shutdown / Reboot* option will allow the SIB to be shutdown or rebooted directly from the menu system. The user is given the choice to shutdown or reboot. This will immediately execute the shutdown or reboot command, and will exit the menu system.

4 Links

- EMAC SIB Manual: ftp://ftp.emacinc.com/PCSBC/EMAC_Linux/SIB/EMAC_Linux_3
- EMAC Open Tools: ftp://ftp.emacinc.com/PCSBC/Development_Kits/EMAC_Open_Tools/
- EMAC Eclipse SIB-SDK: ftp://ftp.emacinc.com/PCSBC/EMAC_Linux/SIB/EMAC_Linux_3/Eclipse_SIB-SDK
- Putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/
- WinSCP: http://winscp.net/eng/index.php
- Nano: http://www.nano-editor.org/
- Vi: http://thomer.com/vi/vi.html
- OpenSSH: http://www.openssh.com/
- Boa: http://www.boa.org/
- WU FTPD: http://www.wu-ftpd.org/

If you have any questions regarding your SIB, contact EMAC support (support@emacinc.com) with the subject heading "SIB Support."