TC-6110 Linux User's Manual

First Edition, September 2013

www.moxa.com/product

TC-6110 Linux User's Manual

Copyright Notice

Trademarks

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.

Moxa provides this document as is, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Technical Support Contact Information

www.moxa.com/support

Moxa Americas
Toll-free: 1-888-669-2872
Tel: +1-714-528-6777
Fax: +1-714-528-6778

Moxa China (Shanghai office)
Toll-free: 800-820-5036
Tel: +86-21-5258-9955
Fax: +86-21-5258-5505

Moxa Europe
Tel: +49-89-3 70 03 99-0
Fax: +49-89-3 70 03 99-99

Moxa Asia-Pacific
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

Table of Contents

1

Introduction

Thank you for purchasing the Moxa TC-6110 series of ready-to-run x86 embedded computers. This manual introduces the software configuration and management of the TC-6110-LX, which runs the Linux operating system. For hardware installation, connector interfaces, setup, and upgrading the BIOS, please refer to the **TC-6110 Hardware Manual**.

Linux is an open, scalable operating system that allows you to build a wide range of innovative, small footprint devices. Software written for desktop PCs can be easily ported to the Linux-based embedded computer with a GNU cross compiler and a minimum of source code modifications. Examples of Linux-powered devices in the workplace include enterprise servers, industrial controllers, communications hubs, point-of-sale terminals, and display devices that include HMIs, advertisement appliances, and interactive panels.

The following topics are covered in this chapter:

- **Overview**
- **Software Specifications**
- **Software Components**

Overview

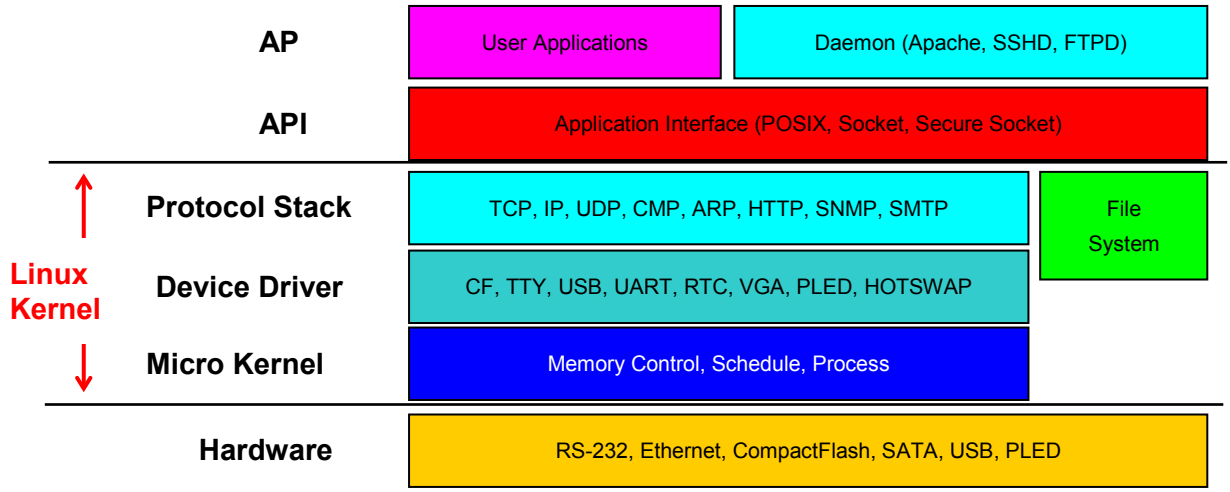
TC-6110 train computers are designed specifically for car-borne train applications like network video recorders, passenger information systems, condition monitoring, and train-to-ground communications. The computers come with two gigabit LAN ports, one RS-232 serial port, three USB 2.0 ports, and two TC-SATA-T storage modules, giving customers a versatile solution applicable to a variety of on-board train computing needs.

Designed for high reliability in the demanding conditions experienced aboard trains, TC-6110 computers come with M12 connectors on both the gigabit LAN ports and dual power inputs, as well as an additional M12 USB port. The TC-6110's expansion modules further allow for highly flexible, convenient integration into a variety of systems. Users can easily add storage modules for additional capacity, gigabit switch modules to expand connectivity and/or bandwidth, or mini PCIe modules for customized peripheral development.

Pre-installed with Linux, the TC-6110 series provides programmers with a friendly environment for developing sophisticated, bug-free application software at a lower cost.

Software Specifications

The Linux operating system pre-installed on the TC-6110 embedded computers is the **Debian 7.1 "Wheezy"** distribution. The Debian project involves a worldwide group of volunteers who endeavor to produce an operating system distribution composed entirely of free software. The Debian GNU/Linux distribution closely follows the standard Linux architecture, making it easy to use programs that meet the POSIX standard. Program porting can be conveniently achieved using the GNU Tool Chain provided by Moxa. In addition to the standard POSIX APIs, device drivers for Moxa UART and other special peripherals are also included. A map of the software architecture is shown below:



This diagram is only an example. Different models or build revisions of the Linux operating system may deviate from the graphic as shown.

ATTENTION

Refer to http://www.debian.org/ and http://www.gnu.org/ for information and documentation related to Debian GNU/Linux and the design and implementation of free software.

Software Components

The TC-6110-LX has been pre-installed with the Debian Wheezy 7.1 Linux distribution. For an exhaustive list of the software packages and applications included with this system, see **Appendix A: Software Components**.

2

Basic Platform Configuration

In this chapter, we explain how to configure a TC-6110-LX computer. There are two ways to do this:

1) Connecting to the TC-6110-LX computer directly, with keyboard/monitor for input/output, or
2) Connecting remotely, over a network, using an SSH console from another Windows or Linux machine.

This chapter describes basic Linux operating system configurations. Advanced network management and configuration instructions will be described in chapter 3, **Managing Communications**.

The following topics are covered in this chapter:

- **Account Management**
- **Starting from a VGA Console**
- **Connecting from an SSH Console**
  - Windows Users
  - Linux Users
- **Adjusting the System Time**
  - Setting the Time Manually
  - NTP Client
  - Updating the Time Automatically
- **Enabling and Disabling Daemons**
- **Managing Service with insserv**
- **Cron—Daemon for Executing Scheduled Commands**
- **Inserting a USB Storage Device into the Computer**

Default User Account and Password

To provide stronger system security, Moxa has disabled the root account. When shipped, the TC-6110-LX only provides a single user account: **moxa** (in lowercase). The default login and password are moxa/moxa.

Login: moxa
Password: moxa

For improved security, the TC-6110-LX computer will force the administrator to change the system password immediately after the first login. For the strongest security, we suggest administrators create a new user account and disable the user moxa.

```
Debian GNU/Linux 7 Moxa tty1

Moxa Login: moxa
Password:
You are required to change your password immediately (root enforced)
Changing password for moxa.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Last login: Thu Sep 26 15:28:25 CST 2013 from 192.168.27. 118 on pts/1
Linux Moxa 3.2.0-4-686-pae @ SMP Debian 3.2.46-1 i686

    ####       ####    ######    ####### ######     ##
     ###      ####   ###    ###    ####   ####     ###
      ###     ###   ###     ###    ###    ##      ###
      ###    ####   ##       ##    ###   #        ####
      ####    # ##  ###       ###    ### ##       ## ##
     ## ##    # ##  ###        ##     ####        #  ##
     ## ###  ## ##  ##         ##     ####        #  ###
     ##  ##  #  ##  ##         ##      ###       #######
     ##  ##  #  ##  ###        ###     #####      #    ##
     ##   ###   ##  ###        ###    ##  ###     #    ###
     ##   ###   ##   ##        ##     ##   ###    ##     ##
     ##   ###   ##    ##      ##      #     ###    #     ##
   ######  #  ######   ########   ####### ###########  ######

For further information check:
http://www.moxa.com/

moxa@Moxa:~#
```

ATTENTION

For the strongest system security, administrators should create a new user account and then delete the default user account, **moxa**. When creating new user accounts that will be associated with device administration, the account must be added to the `/etc/sudoers` file, to enable use of the **sudo** command.

After changing the password, administrators will need to use the **sudo** command for any commands or configurations that require root privileges. If a new user account has been created for use as an administrative account, then the user name must be added to the `/etc/sudoers` file.

As an example of how to use the **sudo** command, when reconfiguring the IP address of the LAN 1 port a network interface on the fly, a user would need to type the following:

```
moxa@MOXA:~# sudo ifconfig eth0 192.168.100.100
```

After issuing this command, you will be prompted for the password of the user account currently used to log in to the system. This will allow the command to be executed with root privileges.

```
moxa@Moxa:~# sudo ifconfig eth0 192.168.100.100
[sudo] password for moxa:
moxa@MOXA:~# sudo ifconfig eth0
[sudo] password for moxa:
eth0      Link encap:Ethernet  HWaddr 00:90:e8:00:df:fe
          inet addr:192.168.100.100  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:41 Base address:0xe000
```

When making extensive changes at the root level it is sometimes preferable to log in with full root access. However, root has been disabled as a user on this computer. Thus, to gain root privileges as a default, use the **sudo –i** command to log in as root.

```
moxa@Moxa:~# sudo –I
[sudo] password for moxa:
root@moxa@Moxa:~# sudo –i
```

Logging in to the Linux Console

Connect a display monitor to the TC-6110-LX VGA connector, and then power up the computer. It takes approximately 30 to 60 seconds for the system to boot up from a cold start. Once the system is ready, a login screen will appear on your monitor. To log in, type the user name and password as requested. As stated above, the default values are both **moxa**.

Login: moxa
Password: moxa

```
Moxa login: moxa
Password:
Last login: Thu Sep 15 22:46:00 CST 2013 on tty1
Linux Moxa 3.2.0-4-686-pae #1 SMP Debian 3.2.46-1 i686
The programs included with the Debian GNU/Liux system are free software;
The exact distribution terms for each program are described in the
Individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
Permitted by applicable law.
moxa@Moxa:~#
```

ATTENTION

It is possible to access the console from multiple (virtual) terminals on the same machine, at the same time. To switch between virtual terminals (called TTY processes, in Linux), press the **Ctrl** button and then indicate the console you wish to switch to using buttons **F1** through **F7**. Any processes started while you are in one terminal will continue running once you switch to another. Keep in mind that terminals which are left open—particularly terminals which have root access—are security risks, so make sure you have logged out of all terminals before leaving a running system.

Connecting from an SSH Console

To offer users a strongly secure remote login console, the TC-6110-LX comes with secure shell remote access (SSH) already enabled. SSH is a much more secure alternative to the now deprecated Telnet. Default IP addresses and netmasks for the network interfaces are:

	Default IP Address	Netmask
LAN 1	192.168.3.127	255.255.255.0
LAN 2	192.168.4.127	255.255.255.0

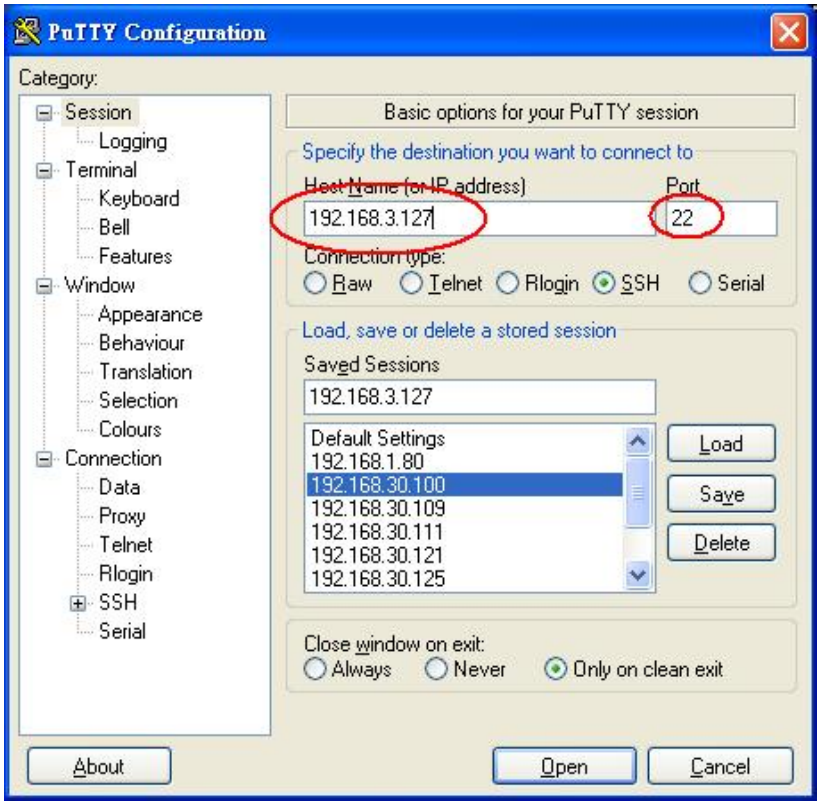
Before using the SSH client, you are advised to change the IP address of your development workstation so that the network ports are on the same subnet as the IP address for the LAN port you will connect to. For example, if you are going to connect to LAN1 on the TC-6110-LX, you should set your PC's IP address to 192.168.3.126, and the netmask to 255.255.255.0. If connecting to LAN2, you would use the same netmask, but set your PC's IP address to 192.168.4.126.

After a connection has been established, type the login name and password as requested to log on to the computer. The default values are both **moxa**.

Login: moxa
Password: moxa

Windows Users

The most popular SSH client for the Windows platform is the freely available **PuTTY** program. To download PuTTY, visit `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`. After installing PuTTY, Windows users will be able to access the TC-6110-LX using SSH. The following screen shows the setup of a sample PuTTY session.



Linux Users

Linux platforms may simply call **ssh** from the console. The following command logs a user in over LAN1:

```
moxa@MOXA:~#ssh 192.168.3.127
```

```
user@remoteDebian-moxa@Moxa:~# ssh 192.168.3.127
The authenticity of host '192.168.3.127 (192.168.3.127)' can't be established.
RSA key fingerprint is 8b:ee:ff:84:41:25:fc:cd:2a:f2:92:8f:cb:1f:6b:2f.
```

When asked if you want to continue connecting over SSH, answer yes by typing **Y**, **y**, or **yes**.

```
Are you sure you want to continue connection (yes/no)? yes_
```

Setting the System Clock and the RTC

The TC-6110-LX uses two clocks to keep time; one is the system time, and the other is the time provided by the RTC (Real Time Clock) built into the TC-6110's hardware. The system clock is set using the **date** command, and the RTC is set using the **hwclock** command.

NTP Client

It is not usually necessary to set the clocks manually, though it is necessary to configure them when first setting up the system. The TC-6110-LX comes with a built-in Network Time Protocol (NTP) client that can access remote NTP servers to synchronize your system clock to worldwide reference clocks. To resynchronize the system time to a remote reference clock, use the **ntpdate** command:

```
moxa@MOXA:~#sudo ntpdate time.stdtime.gov.tw
```

Next, the RTC may be set by using the **hwclock** command:

```
moxa@MOXA:~#sudo hwclock –w
```

```
moxa@MOXA:~# date ; sudo hwclock
Wed Dec 16 16:36:12 CST 2009
Wed 16 Dec 2009 03:38:13 AM CST -0.016751 seconds
moxa@MOXA:~#
moxa@MOXA:~# sudo ntpdate time.stdtime.gov.tw
 16 Dec 03:49:48 ntpdate[2510]: step time server 220.130.158.52 offset
155905087.9
84256 sec
moxa@MOXA:~#
moxa@MOXA:~# sudo hwclock -w
moxa@MOXA:~# date ; sudo hwclock
Wed Dec 16 03:51:07 CST 2009
Wed 16 Dec 2009 03:51:07 AM CST    -0.016771 seconds
```

ATTENTION

Before using the NTP client utility, check your IP address and network settings (gateway and DNS) to make sure an Internet connection is available. You may visit the Network Time Protocol project's home page `http://www.ntp.org` (Oct, 2013) for more information about NTP.

Using a Shell Script for Automatic Updates

As the RTC gets older, it may start to run slow and fail to accurately track time. This section provides one example of how a shell script may be used to repeatedly synchronize the RTC to the system clock by using the Liinux initialization table (**inittab**). Because the system clock will be automatically synched using NTP, the two clocks will reliably keep time. Other methods are also available, for instance using **cron** (shown below, in the section **Cron for Executing Scheduled Commands**) or using the **at** command. Below, we show you how to write a simple shell script for keeping the two clocks synchronized, and show you how to set the system to continuously run the script in the background, across re-boots.

Sample shell script for scheduled clock synchronizations

You may save this shell script using any file name, but it should be saved in the /etc/init.d directory. For example, **/etc/init.d/fixtime.sh**.

```
#!/bin/sh
ntpdate time.stdtime.gov.tw
# You can use the time server's ip address or domain
# name directly. If you use domain name, you must
# enable the domain client on the system by updating
# /etc/resolv.conf file.
hwclock -w
sleep 100
# Updates every 100 seconds. The min. time is 100 seconds.
# Change 100 to a larger number to update the RTC less often.
```

How to run a shell script automatically across re-boots

Copy the example above shell script fixtime.sh to the directory /etc/init.d, and then set its access permissions to **755**.

moxa@MOXA:~# chmod 755 fixtime.sh

Next, use open the initialization table (**inittab**) for editing in your preferred editor (we use VI as an example):

moxa@MOXA:~# vi /etc/inittab

Add the following line to the bottom of the file:

ntp : 2345 : respawn : /etc/init.d/fixtime.sh

Use the command #init q to re-initialize the kernel.

moxa@MOXA:~# init q

NOTE In *nix environments, when inserting a single line at the end of a configuration file it is possible to use a single line command. This allows administrators to save time without opening the config file in an edtor. To insert a single ine to the end of a file, use the **echo** command with **input redirects**:

moxa@MOXA:~# echo "ntp : 2345 : respawn : /etc/init.d/fixtime" >> /etc/inittab

Keep in mind, however, that care must be taken to use a **double caret** (>>). Use of a single caret (>) indicates overwriting the entire file with the single line, and will erase the current configuration.

Setting a Time Manually

System Time

When called with unquoted arguments, the date command will reset the system clock. The time and date must be entered in the format of Month-Date-Hour-Minute-Year.

```
moxa@MOXA:~# date [MMDDhhmmYYYY]
```

Month, date, hour, and minute are all entered in a two digit code, with the year entered using the full four digits, as shown below:

MM: Month
DD: Date
hhmm: Hour and Minute
YYYY: Year

```
moxa@MOXA:~# date
            Tue Aug 20 11:28:05 CST 2013
moxa@MOXA:~# sudo hwclock
[sudo] password for moxa:
Tue 20 Aug 2013 11:28:47 AM CST  -0.422555 seconds
moxa@MOXA:~# date 121616352009
Wed Dec 16 16:35:00 CST 2009
```

```
moxa@MOXA:~# sudo hwclock -w
moxa@MOXA:~# date ; sudo hwclock
Wed Dec 16 16:36:12 CST 2009
Wed 16 Dec 2009 03:38:13 AM CST -0.016751 seconds
```

Setting the RTC

After setting the system time, use **hwclock** to write the current system time to the RTC, as follows:

```
Moxa~# hwclock -w
```

Enabling and Disabling Daemons

To run a custom daemon (i.e., an automated background process called by the system), you should create an inititalization script; this process was briefly described above, in the section **Using a Shell Script for Automatic Updates**. While some people use rc.local to enable daemons, this practice is frowned upon and can lead to cases where services or background processes that require a clean exit are broken at shut-down, and will fail to start again at the next reboot. For this reason, best practices dictate that users who wish to set up a scripted process to run in the background should use inittab and an init script to guarantee the process will be cleanly managed, and any errors cleanly handled by the system.

After scripting a background process, for security's sake and convenience of administration, the script should be saved in `/usr/sbin` and then then linked to from `/etc/rc.local`. Then, an initialization script (init script) should be created, saved into `/etc/init.d`, and logged into `/etc/inittab` (for more on this, see above, **How to run a shell script automatically across re-boots**). **A stripped down sample initialization script** is given below, in appendix C, **Sample Scripts**; you may use either the sample script below, or you may use the one provided with the standard Debian distribution, which may be found at `/etc/init.d/skeleton`. (the skeleton script provides a great deal more commentary and features). The complete process is described below.

1. Copy your custom script to `/usr/sbin` using the convention of `rc<scriptname>`:
   ```
   moxa@Moxa:~# cp <full-path-to-your-program> /usr/sbin/rc<scriptname>
   ```
2. Give this script executable permissions:
   ```
   moxa@Moxa:~# chmod 755 /usr/sbin/rc<scriptname>
   ```
3. Make a copy of the sample file at `/etc/init.d/skeleton` (or the one provided in Appendix C of this manual) and edit it to create an initialization script for your program.
   ```
   moxa@Moxa:~# cp /etc/init.d/skeleton /etc/init.d/<scriptname>
   ```
4. Give the initialization script executable permissions.
   ```
   moxa@Moxa:~# chmod 755 /etc/init.d/<scriptname>
   ```
5. Now activate your script so that it can be run when the system boots; use a low startup priority (below we use 97 for starts, 03 for shutdown).
   ```
   moxa@Moxa:~# update-rc.d <scriptname> default 97 03
   ```

ATTENTION

For more details about which systems in a Linux environment should be used for automated scripting and processes, you may refer to the webpage http://bencane.com/2011/12/30/when-its-ok-and-not-ok-to-use-rc-local/ (Nov, 2013). For more information about creating custom Linux scripts, refer to http://www.linux.com/learn/tutorials/442412-managing-linux-daemons-with-init-scripts

Managing Services with insserv

Linux services can be started or stopped using of the scripts in `/etc/init.d/`. If you want to start up some service, you can use **insserv** to add or remove the service to a specific run level. This tutorial shows you how to add or remove a service from a specified run level.

WARNING

Insserve is a low level tool, and when improperly called can result in an unbootable system. For this reason, current Debian best practices recommends against the use of insserv. If you do not feel comfortable using such a low-level tool, use **update-rc.d** instead. A good summary of how to use update-rc.d is available here: http://www.debuntu.org/how-to-managing-services-with-update-rc-d/ (Nov. 2013)

The example will use a services start-stop utility named **tcps2,** which we shall add to `/etc/init.d/`.

```
# !/bin/sh

### BEGIN INIT INFO
# Provides:          tcps2
# Required-Start:
# Required-Stop:
# Default-Start:      2 3 4 5
# Default-Stop:       0 1 6
# Short-Description:   tcps2
### END INIT INFO

. /lib/lsb/init-functions

export PATH="${PATH:+$PATH:}/usr/sbin:/sbin"

case "$1" in
  start)
       start-stop-daemon --start --quiet --oknodo --pidfile /var/run/tcps2.pid
--exec /usr/sbin/tcps2
       ;;
  stop)
       start-stop-daemon --stop --quiet --oknodo --pidfile /var/run/tcps2.pid
       ;;
esac

exit 0
```

After creating the script, you will now add it as a scheduled service using **insserv**.

To add tcps2 as a service that will start at boot time and run at every runlevel, use the following syntax:

```
moxa@MOXA:~#sudo insserv -v -d tcps2
```

Check to see that the script has been added to each run level:

```
moxa@MOXA:~#ls -l /etc/rc?.d/*tcps*
lrwxrwxrwx 1 root root 15 Jul  6 09:40 /etc/rc2.d/S18tcps2 -> ../init.d/tcps2
lrwxrwxrwx 1 root root 15 Jul  6 09:40 /etc/rc3.d/S18tcps2 -> ../init.d/tcps2
lrwxrwxrwx 1 root root 15 Jul  6 09:40 /etc/rc4.d/S18tcps2 -> ../init.d/tcps2
lrwxrwxrwx 1 root root 15 Jul  6 09:40 /etc/rc5.d/S18tcps2 -> ../init.d/tcps2
```

To remove a service from inittab, use this command:

```
moxa@MOXA:~#insserv -r tcps2
```

Check to make sure the script has been removed.

```
moxa@MOXA:~#ls -l /etc/rc?.d/*tcps*
ls: cannot access /etc/rc?.d/*tcps*: No such file or directory
```

Cron for Executing Scheduled Commands

The **cron** daemon reads `/etc/crontab` to retrieve scripts and other commands to be run at regularly scheduled times.

Cron wakes up every minute and checks each command listed in the crontab file to see if it should be run at that time. Whenever cron executes a command, a report is automatically mailed to the owner of the **crontab** (or to the user named in the MAILTO environment variable in the **crontab**, if such a user exists).

Modify the file `/etc/crontab` to schedule an application. **Crontab** entries follow the format below:

mm	h	dom	mon	dow	user	command
minute	hour	day of month	month	day of week	user	Command to be run
0-59	0-23	1-31	1-12	0-6 (0 is Sunday)		

For example, to synchronize the RTC at 8 AM every day, use the following cron entry:

```
#minute     hour    dom     date    month   dow     user    command
  00          8       *       *       *       *     root    hwclock -w
```

Every column in a crontab entry must be marked with a character. The asterisk indicates "every possible unit," so that setting an asterisk in the day-of-week column will configure cron to run the command on every day of the week. If you wish to run a command "every X minutes" or "every X hours", then use the format `*/X`.

So, using the example above, the hwclock command will be run under root ownership at the 0 minute (i.e. – top of the hour) of 8 AM on every day of the month, for every date, during every month, and on every day of the week. The following example shows another way of using cron to update the system time and RTC.

1. Write a shell script named `fixtime.sh` and save it to the `/home` directory.

```
#!/bin/sh
ntpdate time.stdtime.gov.tw
hwclock -w
exit 0
```

2. Reset the access permissions for `fixtime.sh`**.**

```
moxa@MOXA:~# chmod 755 fixtime.sh
```

3. Modify the **/etc/crontab** file to run `fixtime.sh` every 10 minutes (i.e.: ***/10**) by adding this line:

```
*/10    *   *   *   *   root    /home/fixtime.sh
```

Mounting a USB Storage Device

Since mounting USB storage devices manually can be difficult, a Debian package named **usbmount** is used to mount USB devices automatically. **Usbmount** relies on **udev** to mount USB storage devices automatically under the device nodes `/media/usb0`**,** `/media/usb1`, and so forth. Use the mount command (with no arguments) to verify if the USB device has been successfully mounted.

```
moxa@MOXA:~# mount
...
/dev/sdd1 on /media/usb0 type vfat
(rw,nodev,noexec,noatime,nodiratime,sync,fmask=0022,dmask=0022,codepage=cp437,io
charset=utf8,shortname=mixed,errors=remount-ro)
```

Disconnecting a USB Storage Device

Remember to type the command `moxa@Moxa:~# sync` before you disconnect a USB block storage device. If you do not issue the command, you may lose data.

ATTENTION

Remember to exit the directory you are working in before disconnecting the USB storage device. If you do not first exit the directory, the automated unmount process will fail, possibly corrupting system processes. If you do fail to exit the directory and still disconnect the USB storage device, you may attempt to unmount the device manually by running the following command in the console:

root@moxa@Moxa:~# umount -f /media/usb0

Checking Versions for your Kernel and OS

The program **uname** prints the name, version, and other details about the operating system running on the computer. Use the -**a** option to generate a response similar to the one shown below:

```
root@moxa@Moxa:~# uname -a
Linux Moxa 3.2.0-4-686-pae #1 SMP Debian 3.2.46-1 i686 GNU/Linux
```

If you would like to check Moxa's firmware revision, type:

```
moxa@MOXA:~# kversion –a
```

Using APT to Install and Remove Software

APT is the Debian tool used to install and remove packages. Before installing a package, you need to configure the apt source file, `/etc/apt/sources.list`.

1. Open the **sources.list** file in.the vi editor.

```
moxa@MOXA:~# sudo vi /etc/apt/sources.list
```

2. The configuration file should look something like this:

```
deb http://ftp.us.debian.org/debian/ wheezy main
deb-src http://ftp.us.debian.org/debian/ wheezy main

deb http://security.debian.org/ wheezy/updates main
deb-src http://security.debian.org/ wheezy/updates main
# wheezy-updates, previously known as 'volatile'
deb http://ftp.us.debian.org/debian/ wheezy-updates main
deb-src http://ftp.us.debian.org/debian/ wheezy-updates main
```

3. Update the source list after you configure it.

```
moxa@MOXA:~# sudo apt-get update
```

4. Once you indicate which package you want to install (**openswan**, for example), type:

```
moxa@MOXA:~# sudo apt-get install openswan
```

5. You may use one of the following commands to remove a package.
   a. For a simple package removal:

```
moxa@MOXA:~# sudo apt-get remove openswan
```

   b. For a complete package removal that removes all related configuration files, including those in individual user directories:

```
moxa@MOXA:~# sudo apt-get remove openswan --purge
```

ATTENTION

The APT cache space `/var/cache/apt` is located in tmpfs. If you need to install a huge package, link `/var/cache/apt` to USB mass storage or mount it to an NFS space to generate more free space. If you worry about the available space during a package installation, use the command `moxa@Moxa:~# df –h` to check how much free space is available for tmpfs.

Cleaning Out the Package Cache

APT and the Debian system software utilities generally keep software packages stored on the system even after installation. If the disk partition on which `/var` is located runs out of space, you can free up space by using the **clean** tag with **apt-get**.

```
moxa@MOXA:~# sudo apt-get clean
```

This will delete all software installation packages currently stored in the cache; deleting these packages will not affect the functioning of the system, and will not affect the computer's current configuration.

Determining Available Drive Space

To have the system return the amount of available drive space remaining, use the **df** command with the **–h** tag. The system will return the amount of drive space broken down by file system, as shown in the example below..

```
moxa@MOXA:~# df –h
Filesystem                  Size  Used Avail Use% Mounted on
rootfs                      7.4G  857M  6.4G  12% /
udev                         10M     0   10M   0% /dev
tmpfs                       202M  288K  202M   1% /run
/dev/disk/by-label/TC6110_MOXA  7.4G  857M  6.4G  12% /
tmpfs                       5.0M     0  5.0M   0% /run/lock
tmpfs                       403M  4.0K  403M   1% /run/shm
/dev/sdd1                    7.5G  553M  7.0G   8% /media/usb0
```

3

Managing Communications

The TC-6110-LX ready-to-run embedded computer is a network-centric platform designed to serve as a front-end for data acquisition and industrial control applications. This chapter describes how to configure the various communication functions supported by the Linux operating system.

The following topics are covered in this chapter:

Configuring Network Interfaces

Configuring a Persistent Network Interface Naming Order

Debian Linux systems use the **udev** daemon to detect and enable new network interfaces and to manage the device files that are created for them. Udev must be configured with rules that enforce a ***persistent interface naming order.*** A persistent network interface naming order allows devices to be consistently named with the same device node every time the machine is rebooted. This is important because settings are configured with reference to a device name (e.g, eth1) associated with a particular device (e.g., your Broadcom gigabit Ethernet card). If every time the system is rebooted the system randomly rearranges the naming of your cards—for instance, assigning your gigabit Ethernet card to eth2 and your 10/100 Ethernet card to eth1—then there will be no way to maintain a consistent configuration across restarts.

The rule for setting up network interfaces with a persistent naming order is found here:

`/lib/udev/rules.d/75-persistent-net-generator.rules`

and it looks like this:

```
# PCI device 0x10ec:/sys/devices/pci0000:00/0000:00:1c.1/0000:02:00.0 (r8169)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:90:e8:00:de:a9", ATTR{dev_id}=="0x0", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth1"

#PCI device 0x10ec:/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0 (r8169)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:90:e8:00:de:a8", ATTR{dev_id}=="0x0", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth0"
```

The above example indicates that the system has detected two Ethernet interfaces, and assigned them the names eth0 (which is associated with the MAC address `00:90:e8:00:de:a8)` and eth1 (associated with the MAC address `00:90:e8:00:de:a9)`.

ATTENTION

When replacing or connecting a network interface, the system may fail to remove the old record from `/etc/udev/rules.d/70-persistent-net.rules`. This could cause network interfaces to be detected abnormally. To avoid this problem, simply delete the **70-persistent-net.rules** file and reboot the system.

ATTENTION

It may also be necessary to configure a persistent naming order for other system peripherals (e.g., storage drives); to find out more, you may start with the Writing Udev Rules tutorial, found at Ractivated.Net:
http://www.reactivated.net/writing_udev_rules.html
Symantec also offers an effective tutorial, **Setting Persistent SCSI Device Names On Linux Using UDEV**, found here:
http://www.symantec.com/business/support/index?page=content&id=TECH71007
To get an idea of what Udev can do for you, check out this Linux For You article from 2012, **Some Nifty udev Rules and Examples**:
http://www.linuxforu.com/2012/06/some-nifty-udev-rules-and-examples/

Ethernet Interface Configuration

The TC-6110-LX computer has two 10/100/1000 Ethernet ports named LAN1 and LAN2. The default IP addresses and netmasks of these network interfaces are:

	Default IP Address	Netmask
LAN1	192.168.3.127	255.255.255.0
LAN2	192.168.4.127	255.255.255.0

These network settings can be modified by changing the **interfaces** (`/etc/networking/interfaces`) configuration file, or they can be adjusted temporarily with the **ifconfig** command.

The file used for configuring network interfaces is the **networking interfaces configuration** file, located in the `/etc/network` directory. The `/etc/network/interfaces` file is where you will configure Ethernet LAN ports for either static or dynamic (DHCP) IP addressing. To edit this file directly, open the network configuration file with your preferred editor (below, we use VI):

```
moxa@MOXA:~#/etc/network# sudo vi interfaces
```

Static IP Address

The default static IP addresses can be modified. Below, we show the default configuration; changing these values will change the addressing and broadcast parameters used by the associated interface.

```
### The loopback network interface
auto lo
iface lo inet loopback
### The primary network interface
auto eth0
iface eth0 inet static
       address 192.168.3.127
       netmask 255.255.255.0
       broadcast 192.168.3.255
auto eth1
iface eth1 inet static
       address 192.168.4.127
       netmask 255.255.255.0
       broadcast 192.168.4.255
```

Dynamic IP Address using DHCP

To configure one or both LAN ports to receive an IP address through dynamic assignment, replace **static** with **dhcp** and then comment out the rest of the lines. The eth0 interface is shown below, as an example.

```
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet dhcp
#       address 192.168.3.127
#       netmask 255.255.255.0
#       broadcast 192.168.3.255
```

After modifying the boot settings of the LAN interface, issue the following command to immediately activate the new LAN settings:

```
moxa@MOXA:~# sudo service networking restart
```

Adjusting IP Addresses with ifconfig

IP settings can be adjusted during run-time, but the new settings will not be saved to the flash ROM without modifying the file `/etc/network/interfaces`. For example, the following command changes the IP address of **LAN1** to **192.168.1.1**.

```
moxa@MOXA:~# sudo ifconfig eth0 192.168.1.1
```

Point-to-Point Over Ethernet (PPPoE) Configuration

The Easy Way: pppoeconf

The easiest way to set up a PPPoE connection is to install the Debian package, pppoeconf. This is a script that automates the PPPoE configuration process; it may be used on any connection that is directly linking to an ADSL or other PPPoE modem.

Use apt-get or Aptitude to install ppoeconf:

```
moxa@Moxa: ~# apt-get pppoeconf
```

After installing pppoeconf, call it from the command line:

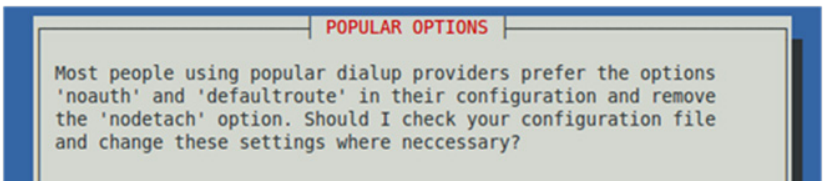
```
moxa@MOXA:~# pppoeconf
```

Next, a dialog will appear telling you pppoeconf is locating your "access concentrator." If your DSL or ADSL modem is connected to an active LAN interface, pppoeconf will find it.

If there are no available concentrators, pppoeconf will tell you, and exit; if this happens, check to see you're your modems are connected properly.

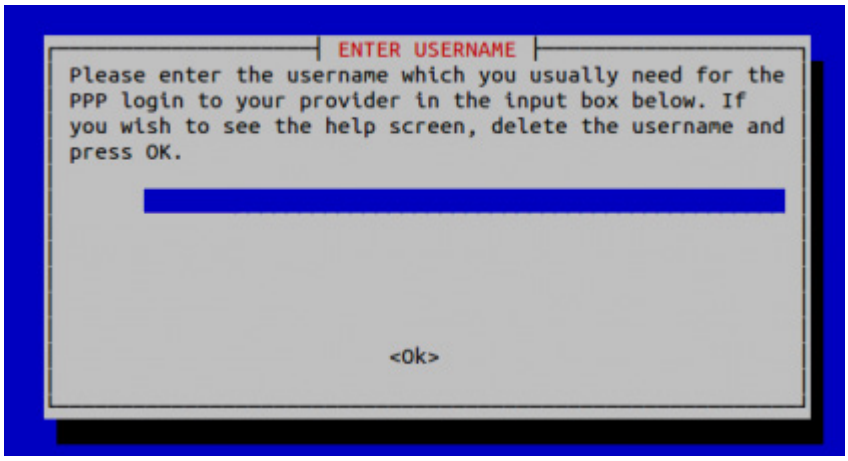
If pppoeconf successfully discovers a concentrator on an available interfaces, it will return this screen:

Answer yes. You will then see this screen:

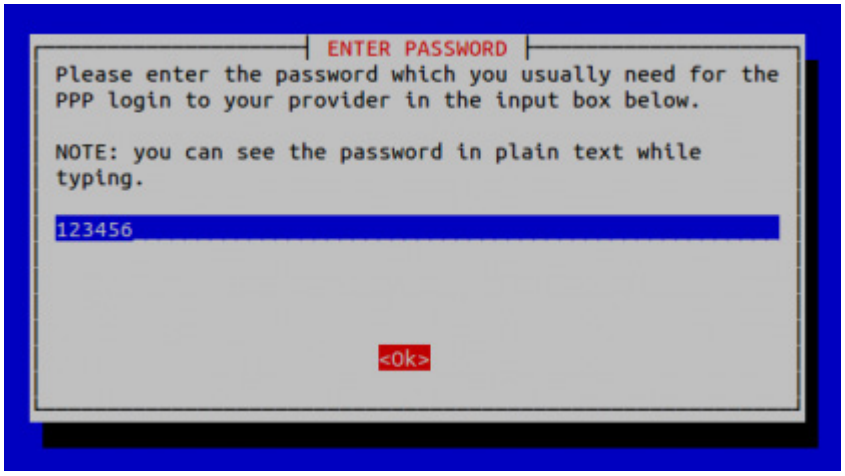


Noauth indicates that the peer does not need to authenticate itself. Nodetach indicates that the connection will not detach from the controlling terminal. Without this option, if a serial device other than the terminal on the standard input is specified, pppd will fork to become a background process.

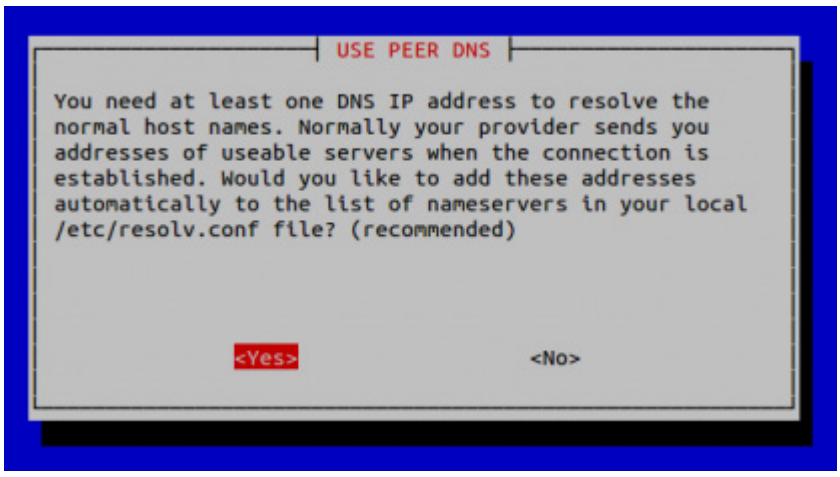
After choosing whether or not to use noauth and nodetach, the pppoeconf will next ask you for your username and password.



Next, enter your password:



Finally, you will need to choose whether or not your PPPoE provider will supply you with DNS server addresses. These addresses are necessary for DNS resolution (se below, in the section **Configuring the DNS Resolver**). It is preferable to click **Yes**, here; however, if your PPPoE provider does not supply these addresses automatically (or if you do not connect to a PPPoE provider directly), click **No** and remember that you will need to enter the DNS server addresses into `/etc/resolve.conf` by hand.



The Difficult Way (Manually)

You may wish or need to connect to your PPPoE provider by manually configuring a connection. Here is how.Use the following procedure to configure PPPoE:

1. Connect the TC-6110-LX's LAN port to an ADSL modem (you may use a cable, HUB, or switch).
2. Log in to the TC-6110-LX as the root user.

3. Edit the file **/etc/ppp/pap-secrets** and add the following entry in the place indicated below:

"username@YourProvider.net" * "password" *

```
# ATTENTION: The definitions here can allow users to login without a
# password if you don't use the login option of pppd! The mgetty Debian
# package already provides this option; make sure you don't change that.

# INBOUND connections

# Every regular user can use PPP and has to use passwords from /etc/passwd
*      hostname       ""      *
 "username@YourProvider.net"    *      "password"      *

# UserIDs that cannot use PPP at all. Check your /etc/passwd and add any
# other accounts that should not be able to use pppd!
guest   hostname       "*"     -
master  hostname       "*"     -
root    hostname       "*"     -
support hostname       "*"     -
stats   hostname       "*"     -

# OUTBOUND connections
```

username@YourProvider.net is the username obtained from the ISP to log in to the ISP account. **password** is the corresponding password for the account.

4. Edit the file **/etc/ppp/options** and add **plugin rp-pppoe** in the indicated place:

```
# Wait for up n milliseconds after the connect script finishes for a valid
# PPP packet from the peer.  At the end of this time, or when a valid PPP
# packet is received from the peer, pppd will commence negotiation by
# sending its first LCP packet.  The default value is 1000 (1 second).
# This wait period only applies if the connect or pty option is used.
#connect-delay <n>

# Load the pppoe plugin
plugin rp-pppoe.so

# ---<End of File>---
```

5. If you connecting over LAN1, use the template below to create a file **/etc/ppp/options.eth0**. LAN2 should be named **/etc/ppp/options.eth1**. All interfaces follow this convention.

```
name username@YourProvider.net
mtu 1492
mru 1492
defaultroute
noipdefault
~
~
 "/etc/ppp/options.eth0" 5 lines, 67 characters
```

Type your username (the one you set in the **/etc/ppp/pap-secrets** and **/etc/ppp/chap-secrets** files) after the **name** option. You may add other options as needed.

6. Set up DNS.
   If you are using DNS servers supplied by your ISP, edit the file **/etc/resolv.conf** by adding the following lines of code:
   nameserver ip_addr_of_first_dns_server
   nameserver ip_addr_of_second_dns_server
   For example:
   nameserver 168.95.1.1
   nameserver 139.175.10.20

```
moxa@MOXA:~# cat /etc/resolv.conf
#
# resolv.conf  This file is the resolver configuration file
# See resolver(5).
#
nameserver 168.95.1.1
nameserver 139.175.10.20
#/etc#
```

Now, you should be able to ise the following command to establish a **pppoe** connection:

```
moxa@Moxa:~# pppd eth0
```

If you want to disconnect the connection, you may use the kill command to kill the **pppd** process.

```
moxa@Moxa:~# kill -9 pppd
```

Notes:

1. If the ADSL modem is connected to the **LAN1** port, the connection will be named **eth0**. If the ADSL modem is connected to **LAN2**, it should be named **eth1**, etc.
2. Type **moxa@Moxa: ~# ifconfig ppp0** to check if the connection is OK. If the connection is OK, you should see the IP address of ppp0. You may use the **ping** command to test the IP address.

```
ppp0 Link encap Point-to-Point Protocol
        inet addr 192.76.32.3   P-t-P 129.67.1.165 Mask 255.255.255.0
        UP POINTOPOINT RUNNING  MTU 1500  Metric 1
        RX packets 33 errors 0 dropped 0 overrun 0
        TX packets 42 errors 0 dropped 0 overrun 0
```

Configuring a Point-to-Point Connection

PPP (Point to Point Protocol) is used to run IP (Internet Protocol) and other network protocols over a serial link. PPP can be used for direct serial connections (using a null-modem cable) over a Telnet link, and links established using a modem over a telephone line.

Modem/PPP access is almost identical to connecting directly to a network through the TC-6110-LX Ethernet port. Since PPP is a peer-to-peer system, the TC-6110-LX can also use PPP to link two networks (or a local network to the Internet) to create a Wide Area Network (WAN).

ATTENTION

The following links will give you more information about setting up PPP:

http://tldp.org/HOWTO/PPP-HOWTO/index.html
http://axion.physics.ubc.ca/ppp-linux.html

The following is an AT command used to connect to a PPP server by modem. Use this command for old ppp servers that prompt for a login name (replace **username** with the correct name) and password (replace **password** with the correct password). Note that **debug crtscts** and **defaultroute 192.XXX.XX.XXX** are optional.

```
moxa@Moxa:~# pppd connect 'chat -v "" ATDT5551212 CONNECT ""' login: username \
             password: password' /dev/ttyM0 115200   \
             debug crtscts modem defaultroute 192.1.1.17
```

If the PPP server does not prompt for the username and password, the command should be entered as follows (replace "username" with the correct username and replace "password" with the correct password):

```
moxa@Moxa:~# pppd connect 'chat -v "" ATDT5551212 CONNECT ""' user username
             password password /dev/ttyM0 115200 crtscts modem
```

The pppd options are described below:

connect 'chat etc...' This option gives the command to contact the PPP server. The **chat** program is used to dial a remote computer. The entire command is enclosed in single quotes because pppd expects a one-word argument for the **connect** option. The options for **chat** are given below:

-v verbose mode; log what we do to syslog

" " Double quotes—don't wait for a prompt, but instead do ... (note that you must include a space after the second quotation mark)

ATDT5551212 Dial the modem, and then ...

CONNECT Wait for an answer.

" " Send a return (null text followed by the usual return)

ogin: username word: password
Log in with username and password.

Note: Refer to the chat man page, chat.8, for more information about the **chat** utility.

/dev/ Specify the callout serial port.

115200 The baud rate.

debug Log status in syslog.

crtscts Use hardware flow control between the computer and modem (at baudrate of 115200 this is a must).

modem Indicates that this is a modem device; pppd will hang up the phone before and after making the call.

defaultroute Once the PPP link is established, make it the default route; if you have a PPP link to the Internet, this is probably what you want.

192.1.1.17 This is a degenerate case of a general option of the form x.x.x.x:y.y.y.y. Here x.x.x.x is the local IP address and y.y.y.y is the IP address of the remote end of the PPP connection. If this option is not specified, or if just one side is specified, then x.x.x.x defaults to the IP address associated with the local machine's hostname (located in /**etc/hosts**), and y.y.y.y is determined by the remote machine.

Connecting to a PPP Server over a Hardwired Link

If a username and password are not required, use the following command (note that **noipdefault** is optional):

```
moxa@Moxa:~# pppd connect 'chat –v" " " " " ' noipdefault /dev/ttyM0 19200 crtscts
```

If a username and password are required, use the following command (note that **noipdefault** is optional, and the username and password are both "root"):

```
moxa@Moxa:~# pppd connect 'chat –v" " " " ' user root password root \
             noipdefault /dev/ttyM0 19200 crtscts
```

Checking the Connection

Once you have set up a PPP connection, there are some steps you can take to test the connection. First, type:

```
moxa@Moxa:~# ifconfig
```

After executing the command, you should be able to see all of the available network interfaces.

ppp0 should be one of the network interfaces. You should recognize the first IP address as the IP address of the computer, and the **P-t-P address** is the address of the server. The output should be similar to this:

```
lo      Link encap Local Loopback
        inet addr 127.0.0.1   Bcast 127.255.255.255 Mask 255.0.0.0
        UP LOOPBACK RUNNING   MTU 2000   Metric 1
        RX packets 0 errors 0 dropped 0 overrun 0

ppp0    Link encap Point-to-Point Protocol
        inet addr 192.76.32.3   P-t-P 129.67.1.165 Mask 255.255.255.0
        UP POINTOPOINT RUNNING   MTU 1500   Metric 1
        RX packets 33 errors 0 dropped 0 overrun 0
        TX packets 42 errors 0 dropped 0 overrun 0
```

Now, type:

```
moxa@Moxa:~# ping XXX.XX.XXX.XXX
```

where **XXX.XX.XXX.XXX** is the address of your name server. The output should be similar to the following:

```
moxa@MOXA:~# sudo ping 129.67.1.165
PING 129.67.1.165 (129.67.1.165): 56 data bytes
64 bytes from 129.67.1.165: icmp_seq=0 ttl=225 time=268 ms
64 bytes from 129.67.1.165: icmp_seq=1 ttl=225 time=247 ms
64 bytes from 129.67.1.165: icmp_seq=2 ttl=225 time=266 ms
^C
--- 129.67.1.165 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 247/260/268 ms
moxa@MOXA:~#
```

Try typing:

```
moxa@Moxa:~# netstat -nr
```

You should see three routes similar to the following:

```
Kernel routing table
Destination   Gateway   Genmask         Flags   Metric  Ref Use
iface
129.67.1.165  0.0.0.0   255.255.255.255 UH      0       0   6
ppp0
127.0.0.0     0.0.0.0   255.0.0.0       U       0       0   0 lo
0.0.0.0       129.67.1.165  0.0.0.0     UG      0       0   6298
Ppp0
```

If your output looks similar but does not have the "destination 0.0.0.0" line (which refers to the default route used for connections), you may have run pppd without the **defaultroute** option. At this point, you can try using Telnet, ftp, or finger, bearing in mind that you will have to use numeric IP addresses unless you have configured **/etc/resolv.conf** correctly.

Setting up a Machine for Incoming PPP Connections

Method 1: pppd dial-in with pppd commands

This first example applies to using a modem, and requiring authorization with a username and password.

#pppd /dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2 login auth

You should also add the following line to the file **/etc/ppp/pap-secrets**:

*** * "" ***

The first star (*) lets everyone login. The second star (*) lets every host connect. The pair of double quotation marks ("") indicates that the file **/etc/passwd** can be used to check the password. The last star (*) is to let any IP connect.

The following example does not check the username and password:

moxa@Moxa:~# pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2

Method 2: pppd dial-in with pppd script

Configure a dial-in script **/etc/ppp/peer/dialin**

```
# You usually need this if there is no PAP authentication
noauth
#auth
#login

# The chat script (be sure to edit that file, too!)
init "/usr/sbin/chat -v -f /etc/ppp/ppp-ttyM0.chat"

# Set up routing to go through this PPP link
defaultroute

# Default modem (you better replace this with /dev/ttySx!)
/dev/ttyM0

# Speed
115200

# Keep modem up even if connection fails
persist
crtscts
modem
192.168.16.1:192.168.16.2
debug
-detach
```

Configure the chat script **/etc/ppp/ppp-ttyM0.chat**

```
SAY    'Auto Answer ON\n'
''     ATS0=1
```

Start the **pppd** dial-in service.

```
moxa@MOXA:~# sudo pppd call dialin
```

ATTENTION

If you would like to have auto dial-in service, you can launch the dial-in service in **/etc/inittab** with the respawn command:

```
moxa@MOXA:~# sudo echo "p0:2345:respawn:pppd call dialin" >> /etc/inittab
```

Telnet/FTP/TFTP Server

For security reasons, the TC-6110-LX only supports SSH and SFTP. The Telenet, FTP, and TFTP are installed, but have been disabled. Moxa strongly recommends against the use of Telnet or FTP, both of which are considered deprectated, today. However, if you wish to use one of these services, you may follow the directions below to enable or disable these services.

Enabling a Telnet, FTP, or TFTP Server

The following example shows the default content of the file `/etc/inetd.conf`. For security's sake, the Telnet, FTP, and TFTP servers are disabled by default. To enable these services, add the following content to `/etc/inetd.conf`:

```
telnet  stream  tcp nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp nowait  root    /usr/sbin/tcpd  /usr/sbin/proftpd
…
tftp    dgram   udp wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd  /srv/tftp
```

Then restart the inetd service:

```
moxa@MOXA:~# sudo service openbsd-inetd restart
```

Disabling a Telnet/FTP/TFTP Server

If, after enabling one of these servers, you wish to disable it again you may do so by commenting out the relevant line inserting a hash (**#**) as the line's first character. Below, the **TFTP** server has been disabled using this method.

```
telnet  stream  tcp nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp nowait  root    /usr/sbin/tcpd  /usr/sbin/proftpd
…
#tftp   dgram   udp wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /srv/tftp
```

As with any other changes to the inet.d configuration, you must restart the inetd service for the changes to take effect.

```
moxa@MOXA:~# sudo service openbsd-inetd restart
```

DNS Utilities

Basic DNS utilities are responsible for managing a system's hostname, DNS resolver, and the name service switch. The three configuration files associated with these services are `/etc/hostname`, `/etc/resolv.conf`, and `/etc/nsswitch.conf`.

Configuring the OS Hostname

When remotely administrating large networks, it is desirable to provide each computer with a descriptive hostname. This is set by changing the **hostname** file; `/etc/hostname` is a file with a single line that contains

the hostname, which can only contain the ascii characters a through z, the numbers 0 through 9, and a hyphen. Hostnames must not include dots (periods), because the hostname is used as part of a fully qualified URL.

1. To change the hostname, use the following command:

```
moxa@MOXA:~# sudo echo ”your-preferred-hostname” > /etc/hostname
```

2. Load the new hostname:

```
moxa@MOXA:~# sudo /etc/init.d/hostname.sh start
```

3. Check the new hostname.

```
moxa@MOXA:~# hostname
your-preferred-hostname
```

Configuring the DNS Resolver

This is the file most in need of updating when configuring DNS. For example, before using the command

```
moxa@Moxa:~# ntpdate time.stdtime.gov.tw
```

to update the system time, you will need to add a DNS server address to the resolver configuration. Ask your network administrator for addresses to preferred DNS servers. Each server's address is specified by prefacing the line with **nameserver**. For example, to add a DNS server with IP address is 168.95.1.1 to `/etc/resolv.conf`, you would simply append **nameserver 168.95.1.1** to the end of the file.

```
moxa@MOXA:~#/etc# echo “nameserver 168.95.1.1” >> resolv.conf
moxa@MOXA:~#/etc# cat resolv.conf
# resolv.conf  This file is the resolver configuration file
# See resolver(5).
#
#nameserver 192.168.1.16
nameserver 140.115.1.31
nameserver 140.115.236.10
nameserver 168.95.1.1
```

Configuring the Name Service Switcher

The name service switcher configuration file is **nsswitch.conf**; this file defines in what sequence system databases will be referenced to retrieve name service information when resolving URLs to IP addresses. The file is plain ASCII text, with columns separated by spaces or tab characters. The first column specifies the database name. The remaining columns describe the order of sources to query and a limited set of actions that can be performed by lookup result; the sources will be referenced in the order they appear on the line, from right to left.

Five service specifications may be indicated for any source: **files**, **db**, **nis**, **nisplus**, or **compat**. For the hosts database, you may also specify dns; compatibility mode (**compat**) may only be used with the **passwd**, **group**, and **shadow** databases. Use of the **files** source will have the name service switcher search the /etc directory to find a file that matches the source name (e.g., /etc/hosts, /etc/passwd, /etc/group), and then that file will be used. By omitting dns or files you may effectively disable dns or the local hosts file for URL resolution.

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference’ and `info’ packages installed, try:
# `info libc “Name Service Switch”‘ for information about this file.

passwd:         compat
group:          compat
shadow:         compat
```

```
hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Apache Web Server

The Apache config directory houses four basic directories: sites-enabled, mods-enabled, sites-available, and mods-available. The **sites-enabled** directory is where active websites are enabled; this is done by creating a symlink into the sites-available directory. **Sites-available** is a repository for all sites, whether inactive or active. The **mods-available** directory houses Apache **software modules**, which allow administrators to adjust the size and features of the Apache webserver to the particular needs of the application. The **mods-enabled** directory enables modules to be loaded by, again, symlinking back to the relevant module located in the mods-available directory.

ATTENTION

There are many Apache modules that may be of use to administrators in need of customizations to their webserver, such as speeding up CGI, or building heightened security. Webserver modules and features are beyond the scope of this manual. If you wish to find a complete list and full documentation for the native modules, please refer to the Apache webserver documentation, found here:

http://httpd.apache.org/modules/

For a more completely list of available modules that includes third-party modules, you may refer to Wikipedia:

http://en.wikipedia.org/wiki/List_of_Apache_modules

Default Homepage

The Apache web server's main configuration file is **/etc/apache2/sites-enabled/000-default**, with the default homepage located at **/var/www/index.html**.

Before you modify the homepage, use a browser (such as Microsoft Internet Explore or Mozilla Firefox) from your PC to test if the Apache web server is working. Type the LAN1 IP address in the browser's address box to open the homepage. If the default address hasn't changed, then when you type http://192.168.3.127/ in the address bar of your web browser you should see Apache's default web page.

Configuring the Common Gateway Interface (CGI)

Setting Up CGI

CGI comes already enabled. The root CGI directory (where you should put CGI scripts) is /usr/lib/cgi-bin. You may change this to /var/www/cgi-bin, if you so desire.

ATTENTION

If you have more questions about setting up CGI on Apache 2.2, you may refer to this web page:

http://httpd.apache.org/docs/2.2/howto/cgi.html

Disabling CGI

Support for CGI scripting is enabled by default. To disable it, follow the steps below.

1. Open the configuration file for editing (below, we use VI):
   moxa@MOXA:~# vi /etc/apache2/sites-enabled/000-default
   Then, comment out the following lines:

```
moxa@MOXA:~#/etc# vi /etc/apache2/sites-enabled/000-default
#ScriptAlias /cgi-bin/ /usr/lib/w3m/cgi-bin/
#<Directory "/usr/lib/w3m/cgi-bin/">
#     AllowOverride None
#     Options ExecCGI -MultiViews +SymLinksIFOwnerMatch
#     #Order allow,deny
#     Order deny,allow
#     Allow from all
#</Directory>
```

2. Re-start the apache server.

```
moxa@MOXA:~# sudo service apache2 restart
```

ATTENTION

If you have CGI scripts you wish to transfer to the server, make sure you make the files executable. The command for this is the **change mode** command, **chmod**. To make a file read-only but executable, you may use the numerical combination **555**. To make a file read only but available for editing by root, use the numerical key **755**. The syntax is as follows:

```
MOXA:~#chmod 555 /usr/lib/cgi-bin/[NAME OF YOUR FILE HERE]
```

Saving Web Pages to a USB Storage Device

Some applications may have web pages that take up a lot of storage space. This section describes how to save web pages to the USB mass storage device, and then configure the Apache web server's DocumentRoot to open these pages. The files used in this example can be downloaded from the Internet.

1. Connect the USB storage device to a USB port, and check where the device is mounted:

```
moxa@Moxa:~# sudo mount
```

2. Prepare the web pages and then save the entire `/var/www` directory to the appropriate USB storage device. Normally, this should be `/media/usb0`.

```
moxa@Moxa:~# sudo cp –a /var/www/ media/usb0/
```

3. Now change the Document Root setting. Open the basic Apache config file in an editor:

```
moxa@MOXA:~# /etc# sudo vi /etc/apache2/sites-avaliable/default
```

4. To enable Apache to read your website from the USB device, you must change the **DocumentRoot** entry in the Apache configuration file so that it points to the USB storage device. Navigate to the section beginning with DocumentRoot, and change the directory that immediately follows to **/media/usb0/www**. For a standard, unsecured html page, edit `/etc/apache2/sites-available/default` as below.

```
DocumentRoot /media/usb0/www
<Directory />
      Options FollowSymLinks
      AllowOverride None
</Directory>
```

5. If you have CGI scripts, you must now also change the same file so that the CGI entries point to the files on the USB device. Change your basic Apache configuration file so that it matches the lines shown in red, below:

```
ScriptAlias /cgi-bin/ /media/usb0/www/cgi-bin/
<Directory "/media/usb0/www/cgi-bin/">
        AllowOverride None
        Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
</Directory>
```

6. For webpages that will be connecting using the secure sockets layer, you will need to edit the SSL configuration file. Open the config file using the following command:

```
moxa@MOXA:~#/etc# sudo vi /etc/apache2/sites-avaliable/default-ssl
```

7. Make the changes to your config file so that it matches the lines shown in red below:

```
<VirtualHost *:443>
...
        DocumentRoot /media/usb0/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
...
        ScriptAlias /cgi-bin/ /media/usb0/www/cgi-bin/
        <Directory "/media/usb0/www/cgi-bin/">
                AllowOverride None
                Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
...
</VirtualHost>
```

8. Use the following compound command to restart the Apache web server:

```
MOXA:~# cd /etc/init.d && apache2 restart
```

9. Start your browser and connect to the TC-6110-LX by typing the current LAN1 IP address in the browser's address box.

ATTENTION

Visit the Apache website at http://httpd.apache.org/docs/ for more information about setting up Apache servers.

If you would like to check your website for HTML compliance, click on the following link to download the web page test suite from the World Wide Web Consortium:

http://www.w3.org/MarkUp/Test/HTML401.zip

Netfilter/iptables

Netfilter is an administrative tool for setting up, maintaining, and inspecting the Linux kernel's packet filtering rule tables. Netfilter is a **stateful firewall**, which means that it filters packets by tracking connections, rather than each and every individual packet. For more information on connection tracking, see the section **Connection Tracking**, in this same chapter, below.

In Netfilter, a few fundamental **rule tables** are pre-defined, with each table containing built-in chains and user-defined chains. Tables form the highest layer of organization for Netfilter's rule sets, and **rule chains** form the middle layer, by which individual rules are ordered. Each chain is a list of rules that are applied (or not) to a packets as they traverse the chains. Each rule specifies what to do with a matching packet. A rule (such as a jump to a user-defined chain in the same table, or an order to drop a certain type of packet) is also called a **target**.

Netfilter is based around three fundamental tables: **Filter** tables, **NAT** tables, and **Mangle** tables. These tables in turn are structured around a few basic, built-in rule chains. There are five basic rule chains: PREROUTING, INPUT, FORWARDING, OUTPUT, and POSTROUTING. In addition to these five built-in chains, it is possible for users to add user-defined chains of their own devising, and insert them into the filtering and mangling procedures wherever they are needed. Thus, Netfilter may be said to have three layers: the most basic is the rules layer, the next is the chains layer (which order the rules), and the final is the table layer, which orders the rule chains.

Overview of Basic Netfilter Architecture:

(1) **IP Tables and IP Chains** Review
  (a) **The NAT Table**
  (b) **The Filter Table**
  (c) **The Mangle Table**
(2) **Understanding Basic Traffic Flows**
  (a) **Netfilter Hierarchy for Incoming Packet**s
(3) **Connection Tracking**

Building the Firewall: Writing Filter Rules

(4) **Policies: Setting Default Firewall Behavior**
(5) **Viewing and Manipulating Rulesets**
(6) **Writing Rulechains**

Setting Up NAT

ATTENTION

For more information on configuring Netfilter/iptables, you may consult the official project website.
Homepage: http://www.netfilter.org/
Documentation: http://www.netfilter.org/documentation/index.html#documentation-howto
Neftilter Extensions: http://www.netfilter.org/documentation/HOWTO//netfilter-extensions-HOWTO.html

IP Tables and IP Chains

The highest layer of organization in Netfilter is the **table layer**. This is where all of the **rule chains** are organized. Rule chains are ordered lists of packet filtering and packet mangling rules; each chain represents a basic flow of operations to be performed on a packet at that stage. Where chains are prioritized lists of rules, tables are prioritized lists of chains. Additionally, each of Netfilter's built-in tables comes with a set of built-in chains that are associated with it; these chains set the basic path packets will traverse as they are processed by Netfilter. To view and manipulate (delete, flush, and add) rule tables, rulechains, and individual rules, refer to the section below, **Manipulating Rulesets**.

The NAT Table

The NAT table is the first table that all packets will encounter; no filtering takes place in this table. The only packet alterations enforced by the NAT table are changes to the **source** and **destination** addresses; moreover, only the first packet of a new connection will traverse this table: after the first packet in a **connection** has been processed, the result will be automatically applied to all future packets in the same connection (for more information on connections, see the section **Connection Tracking**, in this same chapter, below).

When the NAT table alters the destination address (on inbound packets, in the PREROUTING chain), it is called **Destination Network Address Translation** (**DNAT)**, or **Port Forwarding**. When the NAT table alters the source address (on outbound packets, in the POSTROUTING chain), it is called **Source Network Address Translation** (**SNAT**), or **IP Masquerading**. Netfilter conventions distinguish Masquerading from SNAT in the following way:

- **Masquerading** is a form of SNAT where you let your firewall automatically detect the external interface address
- **SNAT** refers a situation where you explicitly specify what source address will be used when re-writing the outbound source address field.

The NAT table does not filter packets. Packet filtering is reserved for the **Filter Table**.

The NAT table utilizes the built-in PREROUTING, OUTPUT, and POSTROUTING rule chains.

The Filter Table

The Filter table is the only table that is responsible for filtering packets; it should never alter them in the ways that the Mangle and NAT tables do, e.g., it should not alter the information in individual packets. The only work done by the Filter table consists of executing the targets ACCEPT, DROP, QUEUE, or RETURN.

ACCEPT means the packet continues traversing the chain.
DROP quietly drops the packet, without notifying the sender.
QUEUE passes the packet to userspace, where it may be picked up by the Mangle table, or may be passed along to other userspace utilities or modules.
RETURN sends the packet back to the rule following the last rule it passed in the **previous** rule chain; that is, when a rule is forwarded from one rule chain to another, the RETURN target will send a packet back to the next in the rule chain from which it was forwarded.
In addition, there one target extension may also be used with the Filter table:
Reject will drop the packet, but send an ICMP notification to the sending machine that the packet has been dropped.

The Filter table uses the built-in INPUT, OUTPUT, and FORWARD rule chains

The Mangle Table

The Mangle table is primarily used to prioritize certain connections for quality of service optimizations; it is used for general packet header modification, such as setting the Time-to-Live (TTL) or Type-of-Service (TOS) fields, or to set an internal mark (called **nfmark**, and set with the MARK target) to identify the packet for later processing.

The Five Built-In Rule Chains

The tables handle five built-in chains:

1. All inbound packets hit the **PREROUTING** chain, with no exceptions. Any changes performed on the packets here are done before the routing decision and filtering is done. When connections are bound for machines located on the local subnet this chain will alter the destination IP address address for **destination address translation** (DNAT). By the time a packet reaches the PREROUTING chain, all checks on the IP headers have been completed, but the packet has not yet been routed.
2. The **INPUT** chain receives all **inbound** packets which are addressed to the local intranet served by this firewall. All packets which are addressed to the local intranet will be filtered here, before they continue onwards.
3. The **FORWARD** chain receives and filters all packets which are addressed to computers which are not located on the local intranet located behind the firewall, i.e., it redirects packets which are intended to be forwarded to other parts of the network which are not located on the subnet administered by the firewall, or which have arrived from sections of the network (not located behind the administered subnet) and are destined for the open Internet.

4. The **OUTPUT** chain receives all **outbound** packets which are addressed to computers outside the local intranet. All packets which are addressed to the local intranet served by the firewall will be filtered here, before they continue outwards, onto the Internet.
5. The **POSTROUTING** chain is the very last chain that is applied; all outbound packets which are leaving the local machine (or subnet) will pass through this chain. Packets which are processed by the POSTROUTING chain have already been routed, but have not been sent over the Ethernet. This is where Netfilter performs **source address translation** (SNAT), altering the source address forom the IP address that is used on the local intranet to the one which identifies the firewall on the open Internet.

User-Defined Chains

User-defined chains are used to create customized filters for a wide variety of needs; however, there are some commonly used chains which most administrators call when building a firewall. One example follows:

```
moxa@Moxa:~# iptables –N TCP && iptables –N UDP
```

This creates a user-defned chain called TCP and another called UDP, which you may use to manage protocols later on. To see how to implement these chains in the INPUT chain, see below, **Rule Examples: Applying User-Defined Chains**.

ATTENTION

To find out what rules are currently written into each table and chain, use the commands described below, in the section **Viewing and Manipulating Rulesets**.

Understanding Basic Traffic Flows

Users should recognize that these five chains may be used to build three fundamental traffic flows. Additionally, certain chains are only associated with certain tables. For more information on which tables use which chains, see the next section,

A) **Forwarded** packets will traverse this set of chains in the following order:

PREROUTING	→	**FORWARD**	→	**POSTROUTING**
(in the NAT table)		(in the Filter table)		(in the NAT table)

B) **Inbound** traffic that is destined for the local subnet will traverse this set of chains:

PREROUTING	→	**INPUT**	→	**INPUT**
(in the NAT table)		(in the Mangle table)		(in the Filter table)

A) **Outbound** traffic that is leaving the firewall will traverse this set of chains:

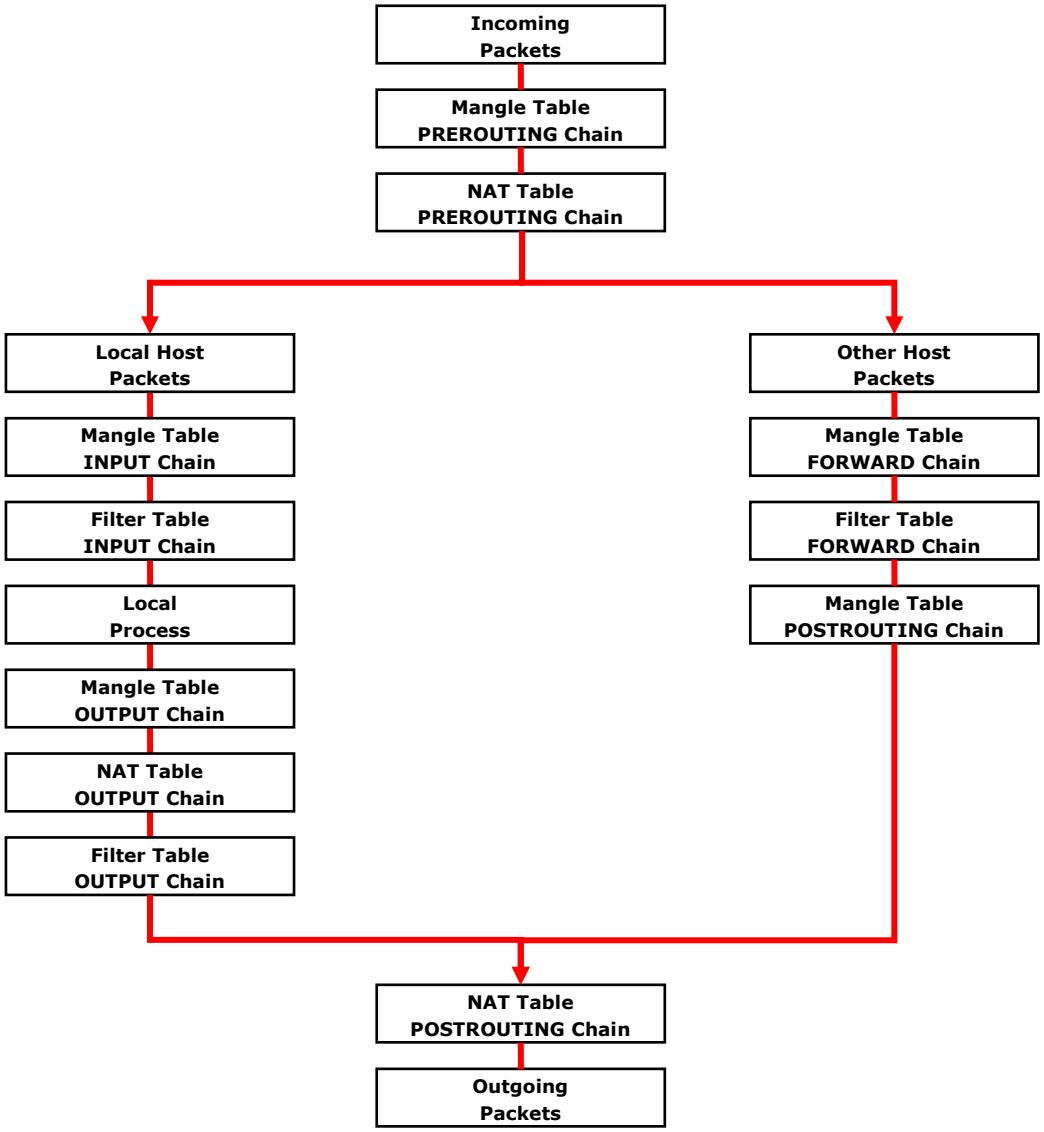
OUTPUT	→	**OUTPUT**	→	**POSTROUTING**
(in the NAT table)		(in the Mangle table)		(in the Filter table)

ATTENTION

Building complex firewalls using the Netfilter rules and interface can become overwhelming, even for experienced administrators. If you require advanced firewall capabilities, Moxa recommends using a Netfilter configuration interface. One of the easiest to learn and most powerful is the Shorewall Firewall. Shorewall is available as a standard Debian package, and may be downloaded using apt-get. Shorewall documentation is available at the Shorewall website, found at **http://www.shorewall.net**.

Netfilter Hierarchy for Incoming Packets

This figure shows how packets traverse the table hierarchy. Outbound packets originating on the local network start at the box labeled **Local Process**. Inbound packets start at the top box labeled **Incoming Packets**.



ATTENTION

Be careful when setting up iptables rules. Incorrectly configured rules can very easily break connectivity with a remote host. For simple setups requiring minimal configuration (five rules or less), Moxa recommends directly configuring iptables using the console and a standard editor. For more complicated setups, users may use Arno's iptables firewall script, or for very large, extremely complicated setups Moxa recommends the Shoreline Firewall. The following links will take you to further information about iptables setups and the various software packages mentioned above.

The netfilter/iptables Project Homepage: http://www.netfilter.org/index.html
The Official neftilter/iptables packet-filtering HOWTO:
http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.htm
Arno's iptables Firewall (click on IPTABLES FIREWALL tab at the top navigation ribbon):
http://rocky.eld.leidenuniv.nl/joomla/
The Shoreline Firewall Homepage (lots of information about netfilter/iptables, as well):
http://www.shorewall.net/Documentation_Index.html
Public iptables/neftilter Forum: http://www.linuxguruz.com/iptables/

Connection Tracking

A connection tracking system does not filter packets. The Netfilter connection tracking system monitors kernel memory structures to keep track of the state of each connection; this means that it logs the protocol types, port number pairs, and source and destination IP addresses, and associates that with various connection states and timeout values. By being able to track connection states, it is possible to build much more powerful and secure filtering rules.

There are four states that may be defined for a connection:

- NEW
  This is the state when a connection is just initiating: the firewall has only seen traffic in one direction (either inbound or outbound) and if the packet is a valid one for initating a connection (i.e., a SYN packet for a TCP request).
- ESTABLISHED
  This is used to describe a connection that has been successfully negotiated, and packet are being exchanged in both directions.
- RELATED
  At the application layer there are some protocols—like FTP passive mode, for instance—which are difficult to track. FTP passive mode uses a wide range of ports, from 1024 to 65535, rather than just one; tracking in this connections is much more difficult than simply tracking a connection across a single port (typically port 20, in FTP). The connection tracking system defines an expectation, which is a connection that is expected to happen in a set period of time, but that has a limited lifetime. Using helpers and expectations, the Netfilter connection tracking system is able to track connections according to patterns by defining **master** connections, and **related** connections.
- INVALID
  This is used to identify packets that do not follow the expected behavior of a connection. Sytems administrators can set filters to drop them.

Policies: Setting Default Firewall Behavior

Netfilter **policies** set the default behavior for its built-in tables, and policies may only be set for Netfilter's built-in tables. This means that policies set the default behavior for all packets handled by the firewall: if a packet arrives which no rule can process, Netfilter will default to the root policy set for that connection. Policies may be set for every table and chain, which means that default policies may be independently set for inbound, outbound, and forwarded packets.

The default policy for most firewalls should be an across-the-board **drop** all connections; after setting the policies to drop all connections, administrators may then add exceptions to allow connections through on a case-by-case basis. This section will only show you how to set the policies; o see how to write rules, look at the section below, **Writing Rulechains**.

WARNING

Firewall rules are ony valid for the time the computer is on. If the system is rebooted, the rules will be automatically flushed. To save a ruleset so that it loads on the next reboot, use the following command:

```
moxa@Moxa:~# /sbin/service iptables save
```

Setting Firewall Policies

```
moxa@Moxa:~# iptables [-t tables] [-P, --policy chain target] [Policy: ACCEPT, DROP,
    ETC]
```

Command Arguments:

-P, --policy: This sets a default policy the firewall will enforce on a particular chain for a particular table. Only built-in chains (i.e.: not user-defined) can have policies. Possible targets for policy enforcement are

INPUT, OUTPUT, FORWARD, PREROUTING, OUTPUT, and POSTROUTING. Possible policies that may be enforced on these chains are ACCEPT, DROP, QUEUE, and RETURN (see below for explanation).
INPUT: Targets packets coming into the TC-6110-LX over the **filter**, **mangle**, or **security** tables.
OUTPUT: Targets locally-generated packets leaving the TC-6110-LX. All tables have an output chain.
FORWARD: Targets packets routed through the machine, on the **filter**, **mangle**, or **security** tables.
PREROUTING: Targets packets for alteration before they have traversed the firewall; used on the **NAT**, **mangle**, and **raw** tables.
POSTROUTING: Targets packets as they are about to be sent out over the **NAT** and **mangle** tables.

Policy Arguments:

ACCEPT: By default, all packets are let through the chain.
DROP: Packets are dropped, with no notification or response sent back to the originating computer.
QUEUE: Passes the packet to userspace; see **NFQUEUE** in Netfilter/iptables documentation for more information about how these targets are used.
RETURN: Stop traversing this chain and resume at the next rule in the previous (calling) chain.
REJECT: Equivalent to DROP, but it returns a message to the packet's origin.
LOG: Turns on kernel logging for matching packets, printing information on all matching packets on the kernel log where it may be read using *dmesg* or *syslogd*.

Netfilter Policy Examples:

```
moxa@Moxa:~# iptables –P INPUT DROP
```

This changes the default policy so that ***all incoming packets*** on ***all chains*** are ***dropped***, with no notification. This is Moxa's recommended setting for the input interface.

```
moxa@Moxa:~# iptables –P OUTPUT ACCEPT
```

This rule acceps ***all outgoing packets*** that originate on the local network, and is acceptable for a strictly secure internal network. If you change this policy to DROP it will considerably increase the complexity of the firewall. However, you may wish to consider this for computers that will be serving as a firewall to untrusted customers. For instance, to guarantee security on a train computer that will be serving wireless connections from outside the train tolocal passengers, the default rule always be **DROP**, with only specific, secure protocols and services allowed through on a rule-by-rule basis.

To help with the construction of advanced firewalls, Moxa recommends use of the **Shoreline Firewall, mentioned above**.

```
moxa@Moxa:~# iptables –P FORWARD DROP
```

This sets the FORWARD chain in the filter table to **DROP** all packets. ***This is the recommended policy for all firewalls***, and may be safely used on devices occupying a terminal segment in the network topology, this is the appropriate rule.

```
moxa@Moxa:~# iptables –t nat –P PREROUTING ACCEPT
```

The nat tables are for address translation, not for filtering. The **PREROUTING** chain for the **NAT** should be set to **ACCEPT**, otherwise connection initialization packets will not be able to get through the firewall.

```
moxa@Moxa:~# iptables –t nat –P OUTPUT ACCEPT
```

The nat tables are for address translation, not for filtering. The **OUTPUT** chain for the NAT should be set to **ACCEPT**, otherwise connection initialization packets will not be able to get through the firewall.

```
moxa@Moxa:~# iptables -t nat –P POSTROUTING ACCEPT
```

The nat tables are for address translation, not for filtering. The **POSTROUTING** chain for the NAT should be set to **ACCEPT**, otherwise connection initialization packets will not be able to get through the firewall.

Viewing and Manipulating Rulesets

Beginning with this section you will be provided some examples of rules commonly used to manipulate, view, and configure simple firewalls for industrial enviroments. For simple setups, typically only three or four rules are needed to give a device strong protecation against unauthorized network intrusions.

List current rule chains for a target table, or for all tables

The full command for **listing** rule chains is as follows:

```
MOXA:~# iptables [-t table, or multiple, tables,…] [-L chain] [-n]
```

Command Arguments:

-t: Table to manipulate (default: 'filter'); available args are **filter**, **nat**, **mangle**, **raw**, and **security**

-L: Indicates a chain to be listed. If no chain is selected, all chains are listed.

-n: Returns the numeric output of addresses and ports: e.g. TCP and UDP ports are printed as numbers, rather than names. This also saves execution time by preventing iptables from looking up DNS requests.

WARNING

Simple commands listing iptable NAT or filter rules will autoload selected kernel modules, including the connectiong tracking (conntrack) and filter (iptable_filter) modules. On high-capacity production servers, these modules easily overload and bring the networking system down. Whenever a list command is issued, check the message buffer (**dmesg**) to see if drivers have been auto-loaded, and what they are. For more information, see http://backstage.soundcloud.com/2012/08/shoot-yourself-in-the-foot-with-iptables-and-kmod-auto-loading/.

Flush a current rule chain, or delete a user-specified chain

The full command to **flush** rule chains is as follows:

```
MOXA:~# iptables [-t table, or tables] [-FXZ]
```

Command Arguments:

-t: Table to manipulate; choices are **filter**, **nat**, **mangle**, **raw**, and **security**. Defaults to **filter**.

-F: Flush the selected chain (if no chains are specified, this flushes all the chains in the table)

-X: Delete the specified user-defined chain (chain must be empty and all references to the chain must be deleted first); if no argument is given, all non-built-in chains will be deleted

WARNING

The command `moxa@MOXA:~# iptables –F` will flush all iptables rulechains from the kernel, permanently deleting the firewall and fully exposing the computer to the open Internet.

You should save any firewall rules you configure in a file that you can use to convenient re-load them, in the event that they are flushed. Before flushing any rule chains, first make sure you have saved your configuration in an independent file that may be conviently uploaded to Netfilter. The following command will save all of the current iptables rules to `/etc/sysconfig/iptables.save`:

```
moxa@MOXA:~# /sbin/service iptables save
```

Zero-out the packet and byte counters for a rule chain

Zeroing the counters is sometimes useful when monitoring firewall activity for analysis. When used in combination with the list argument, the zero argument will give a precise measurement of the number of packets that have been processed since the last measurement, for all chains, a given chain, or even a given rule within a chain. The full command to **flush** rule chains is as follows:

```
MOXA:~# iptables –L –Z -n [chain [rulenum]]
```

Command Arguments:

-Z: Set the packet and byte counters to zero in all chains, for only a given chain, or only a rule in a chain

Delete a User-Generated Chain

This command deletes a specified user-defined chain.

```
MOXA:~# iptables –X [chain]
```

There must be no references to the chain in other chains or tables, and the chain must be empty, i.e. not contain any rules. You must delete or replace any remaining referring rules before the chain can be deleted. *If no argument is given, this will attempt to delete every user defined chain in the table.*

Writing Rulechains

In this section we show you how to write rules for a simple industrial network firewall. More complicated firewalls—such as those serving public networks, or untrusted customers—are beyond the scope of this manual. For advanced firewall needs, Moxa recommends the use of the Shoreline Firewall, **mentioned above** .

```
MOXA:~# iptables [-t table] [-AI] [INPUT, OUTPUT, FORWARD] [-io interface] /
        [-p tcp, udp, icmp, all] [-s IP/network] [--sport ports] [-d IP/network] /
        [--dport ports] –j [ACCEPT. DROP]
```

-A: **Append** one or more rules to the end of the selected chain
-I: **Insert** one or more rules in the selected chain as the given rule number
-i: Identifies an **interface** which will **received** a packet
-o: Identifies an **interface** over which a packet will be **sent**
-p: Identifies the **protocol** to be filtered
-s: Identifies a **source address** (network name, host name, network IP address, or plain IP address)
--sport: Identifies the **source port**, or the port where the packet originated
-d: Identifies the **destination address** (network name, host name, NAT or IP address)
--dport: Identifies the **destination port**, or the port where the packet will terminate
-j: Jump target. Specifies the target of the rules; i.e., how to handle matched packets.

For example, ACCEPT the packet, DROP the packet, or LOG the packet.

WARNING

For all firewalls using a strict DROP policy on incoming packets, be sure to include a rule that accepts packets on the loopback interface:

```
moxa@MOXA:~# iptables –A INPUT –i lo –j ACCEPT
```

Examples:

REQUIRED RULE for all firewalls:
Accept all packets from the loopback interface:

```
# iptables –A INPUT –i lo –j ACCEPT
```

RECOMMENDED RULE from the **sample firewall** provided in **Appendix C: Sample Scripts**:
Allow all traffic from that belongs to established connections, or new, related traffic:

```
# iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

RECOMMENDED RULE from the **sample firewall** provided in **Appendix C: Sample Scripts:**

Drops all traffic with an invalid state, e.g. “Port Unreachable” when nothing was sent to the host, invalid headers or checksums, and out-of-sequence packets:

```
# iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Basic Filter Rules show examples of how you can open commonly opened ports:

Web server / HTTP:

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Secure-sockets web server / HTTPS:

```
# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Remote SSH Connections **(*REQUIRED RULE*):**

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Incoming UDP Streams:

```
# iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

ATTENTION

ICMPv6 Neighbor Discovery packets will always be classified INVALID (if you don't know what this means, you can probably ignore it). You may accept them with this rule:

```
# iptables -A INPUT -p 41 -j ACCEPT
```

Example 1: Accept TCP packets from 192.168.0.1.

```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.1 –j ACCEPT
```

Example 2: Accept TCP packets from Class C network 192.168.1.0/24.

```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.0/24 –j ACCEPT
```

Example 3: Drop TCP packets from 192.168.1.25 (this rule is only necessary on firewalls where you have set the INPUT policy to ACCEPT; ***this is not recommended***).

```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.25 –j DROP
```

Example 4: ACCEPT all TCP packets addressed for port 21.

```
# iptables –A INPUT –i eth0 –p tcp --dport 21 –j ACCEPT
```

Example 5: Accept TCP packets from 192.168.0.24 to TC-6110-LX's port 137, 138, 139

```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.24 --dport 137:139 –j ACCEPT
```

Example 7: Log TCP packets that visit TC-6110-LX's port 25.

```
# iptables –A INPUT –i eth0 –p tcp --dport 25 –j LOG
```

ATTENTION

To use the rule in Examples 8 and 9, below, remember to first, to load the module ipt_mac:

moxa@MOXA:~# modprobe ipt_mac.

To make a module load across reboots, you may add it to the /etc/modprobe.conf file using this command:

moxa@MOXA:~# echo "ipt_mac" >> /etc/modprobe.conf

Don't forget to backup your modprobe.conf file before altering it, and take care to use the double pointer (>>)—which is **append**—rather the single pointer (>) which is **overwrite**.

Example 8: Drop all packets from MAC address 01:02:03:04:05:06.

```
# iptables –A INPUT –i eth0 –p all –m mac --mac-source 01:02:03:04:05:06 –j DROP
```

Example 9: Accept all packets from MAC address 02:03:04:05:06:07.

```
# iptables –A INPUT –i eth0 –p all –m mac --mac-source 02:03:04:05:06:07 –j ACCEPT
```

Rule Examples: Applying User-Defined Chains

Some network administrators may find it useful to define their own rule chains. Here, we show how to implement them in the INPUT chain, and use the chains defined above, in the section **User-Defined Chains.**

```
# iptables -A INPUT -p udp -m conntrack --ctstate NEW -j UDP
# iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -j TCP
```

The TCP and UDP chains are now attached to the INPUT chain; by adding in the above connection rule, once a connection is accepted by either chain, it will be handled by the RELATED/ESTABLISHED rule. You may now add rules to these chains as if you were adding rules to the INPUT chain. Using some of the INPUT rules defined above as examples:

```
# iptables -A TCP -p tcp --dport 80 -j ACCEPT
# iptables -A TCP -p tcp --dport 443 -j ACCEPT
# iptables -A TCP -p tcp --dport 22 -j ACCEPT
# iptables -A UDP -p udp --dport 53 -j ACCEPT
```

ATTENTION

Sample firewalls are provided in **Appendix C, Sample Scripts**. If you have further questions, please refer to those.

Saving the Firewall

You must your firewall so that it will reload on the next reboot; otherwise, the rules witll be flushed and the firewall permanently deleted. After configuring iptables, the following command will save the ruleset to /etc/sysconfig/iptables:

```
moxa@MOXA:~# /sbin/service iptables save
```

NAT (Network Address Translation)

The NAT (Network Address Translation) protocol translates IP addresses used on a local network into IP addresses used on a connecting network. One network is designated the inside network and the other is the outside network. Typically, the TC-6110-LX connects several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and translates the global IP address used on by packets coming in from the WAN back into local IP addresses.

IP Tables NAT Policies

IP tables policies for the NAT table should all be ACCEPT (see the section above, **Netfilter Policy Examples,** for more information):

```
# iptables -t nat -P PREROUTING ACCEPT
# iptables -t nat -P POSTROUTING ACCEPT
# iptables -t nat -P OUTPUT ACCEPT
```

Source NAT (SNAT) and Destination NAT (DNAT)

Source NAT (SNAT) is when the source address is altered on the first packet of an outbound connection. That is, it changes the originating address (which is usually a LAN address that looks like 192.168.xxx.xxx) for outbound packets so that they show the IP address with which the connection to the open internet is associated.

Destination NAT (DNAT) is when the destination address is altered on the first packet of an outbound connection. That is, it changes the originating address (which is usually a LAN address that looks like 192.168.xxx.xxx) for outbound packets so that they show the IP address with which the connection to the open internet is associated.

ATTENTION

Click on the following link for more information about NAT:
http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html

Enabling NAT Masquerading

NAT masquerading allows you to create a subnet of devices mapped to a single IP address. When used with port forwarding and static IP addressing, it can allow you to expand a single public IP address to a very large LAN.

To enable NAT in your device, first load the NAT module:

```
moxa@MOXA:~# modprobe ipt_MASQUERADE
```

ATTENTION

To make a module load across reboots, you may add it to the /etc/modprobe.conf file using this command:

moxa@MOXA:~# echo "ipt_MASQUERADE" >> /etc/modprobe.conf

Don't forget to backup your modprobe.conf file before altering it, and take care to use the double pointer (>>)—which is **append**—rather the single pointer (>) which is **overwrite**.

In the NAT table (-t nat), Append a rule (-A) after routing (POSTROUTING) for all packets going out ppp0 (-o ppp0) which says to MASQUERADE the connection (-j MASQUERADE).

```
# iptables –t nat –A POSTROUTING –o eth0 –s 555.666.777.888/24 –j MASQUERADE
```

Then turn on IP forwarding:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Using these rules and DHCP, it will now be possible to allow local devices to communicate with devices outside the subnet; however, communications will only be able to be initiated from the local network. To allow full address translation both ways, you will need to set up static IP addresses for your devices, and port forwarding rules.

Setting up a Networked File System: NFS

The Network File System (NFS) is used by client computers to mount a remote disk partition as if it were part of their local hardware. NFS is a distributed file system that allows fast, seamless sharing of files across a network. NFS allows users to develop applications for the TC-6110-LX without worrying about the amount of disk space that will be available. By default, the TC-6110-LX only natively supports the NFS client protocol. Mounting an NFS on a local machine is very simple.

The following procedures illustrate how to mount a remote NFS Server. In the example below, **192.168.3.5**—shown in step 3—is the IP address of the NFS server.

1. Scan the NFS Server's shared directory:

   moxa@MOXA:~# showmount –e HOST

   showmount: Shows the mount information of an NFS Server
   -e: Shows the NFS Server's export list.
   HOST: IP address or DNS address

2. Create a mount point on the machine which will be an NFS client:

   moxa@MOXA:~# mkdir –p /home/nfs/public

3. Mount the remote directory to a local directory:

   moxa@MOXA:~# mount -t nfs -o nolock 192.168.3.5:/home/public /home/nfs/public

   (192.168.3.5 is the example IP address of the NFS server.)

ATTENTION

To set up a mount process to mount at boot-time, copy the mount command into the `/etc/fstab` file. For more information on NFS and its configuration options, you may refer to the NFS homepage, at: **http://nfs.sourceforge.net/** (Dec. 2013).

Setting Up a VPN

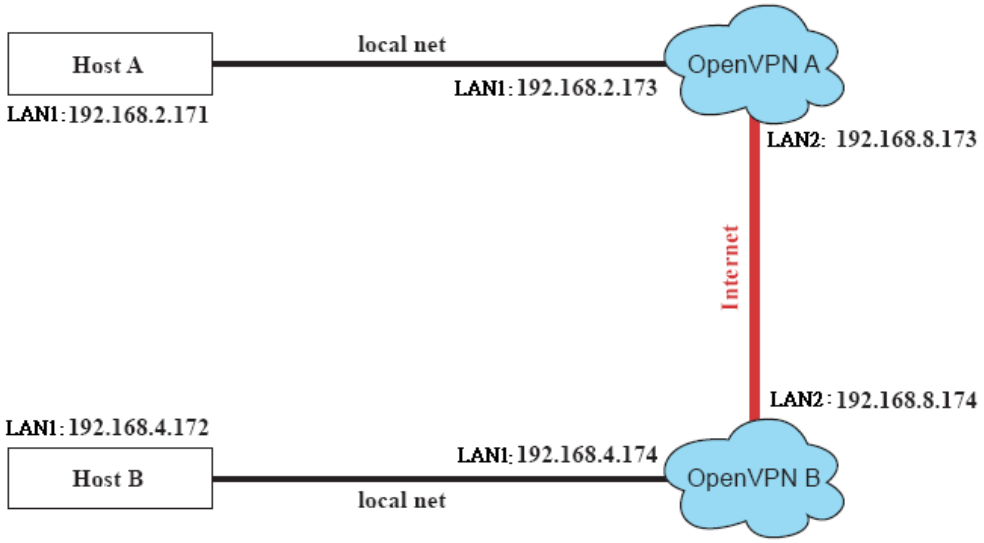
This platform uses the OpenVPN package to provide VPN capability. OpenVPN provides two basic types of tunnels for users to implement VPNS: **Routed IP Tunnels** and **Bridged Ethernet Tunnels**.

An Ethernet bridge is used to connect different Ethernet networks together. The Ethernets are bundled into one bigger, logical network that can communicate securely across the open Internet. Each Ethernet corresponds to one physical interface (or port) that is connected to the bridge.

On each OpenVPN machine, you should carry out configurations in the **/etc/openvpn** directory, where script files and key files reside. Once established, all operations will be performed in that directory.

Ethernet Bridges Linking Indepdent Subnets Over the Internet

This setup will link at two independent subnets over the Internet. It will use at least four machines, as shown in the following diagram. **OpenVPN** designates a dedicated VPN server (perhaps also a firewall), while **Host** designates a client computer located behind the VPN server.



Host A represents the machine that belongs to the subnet served by the VPN server, **OpenVPN A**, and **Host B** represents a machine that belongs to the subnet served by the VPN server, **OpenVPN B**. The two remote subnets are configured for **distinct ranges of IP addresses** on **separate subnets**. When this configuration is moved to a public network, the external interfaces of the OpenVPN machines must be configured for static IPs, or connected to another device (such as a firewall or DSL box) that uses a static address. To set up a bridged Ethernet tunnel following this basic architecture, follow the instructions below:

1. Generate a preset shared key by typing the following command:

```
moxa@MOXA:~# openvpn --genkey --secret secrouter.key
```

2. Copy the keyfile that you have just generated to the OpenVPN machines:

```
moxa@MOXA:~# scp /etc/openvpn/secrouter.key XXX.XXX.X.XXX:/etc/openvpn
```

ATTENTION

Select cipher and authentication algorithms by specifying cipher and auth. To see which algorithms and ciphers are available, type:

```
moxa@MOXA:~# openvpn --show-ciphers
moxa@MOXA:~# openvpn --show-auths
```

For testing purposes, a preshared key is provided at **/etc/openvpn/secrouter.key**. This is adequate for testing, but users must create a new key when going live or their network will be insecure..

Configuring OpenVPN A

1. Modify the remote address in the configuration file `/etc/openvpn/tap0-br.conf` by adding the IP address for the remote server (in this case, OpenVPN B).

```
# point to the peer
remote 192.168.8.174
       dev tap0
       port 1194
       secret /etc/openvpn/secrouter.key
```

```
       cipher DES-EDE3-CBC
       auth MD5
       tun-mtu 1500
       tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
#comp-lzo
```

2. Next, modify the routing table in `/etc/openvpn/tap0-br.sh script`.so that it maps the internal subnet VPN server A will be serving.

```
#-------------------------Start---------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 dev br0
#---------------------------end---------------------------
```

3. And then configure the bridge interface in `/etc/openvpn/bridge`.

```
#!/bin/bash
# Create global variables
# Define Bridge Interface
br="br0"
# Define list of TAP interfaces to be bridged,
       # for example tap="tap0 tap1 tap2".
       tap="tap0"
# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="eth1"
eth_ip="192.168.8.173"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.8.255"
#gw="192.168.8.174"
...
```

4. Start the VPN link by calling the bridge script:

```
moxa@MOXA:~# /etc/openvpn/bridge restart
```

Configuring OpenVPN B,

5. Modify the **remote address** entry in the VPN configuration file, `/etc/openvpn/tap0-br.conf`.

```
# point to the peer
remote 192.168.8.173
dev tap0
secret /etc/openvpn/secrouter.key
       cipher DES-EDE3-CBC
       auth MD5
       tun-mtu 1500
       tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
#comp-lzo
```

6. Next modify the routing table in the `/etc/openvpn/tap0-br.sh script` file.

```
#---------------------------------Start---------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
```

```
route add -net 192.168.2.0 netmask 255.255.255.0 dev br0
#--------------------------------- end ----------------------------
```

7. And then configure the bridge interface script in /etc/openvpn/bridge.

```
#!/bin/bash
# Create global variables
# Define Bridge Interface
br="br0"
# Define list of TAP interfaces to be bridged,
      # for example tap="tap0 tap1 tap2".
      tap="tap0"
# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="eth1"
eth_ip="192.168.8.174"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.8.255"
#gw="192.168.8.173"
...
```

8. Start the bridge script file to configure the bridge interface.

moxa@MOXA:~# /etc/openvpn/bridge restart

9. Start the OpenVPN peers that are on machine OpenVPN A and OpenVPN B with the following command:

moxa@MOXA:~# openvpn --config /etc/openvpn/tap0-br.conf&

If you see a line that looks like **Peer Connection Initiated with 192.168.8.173:5000**on each machine, then the connection the Ehternet bridge has been successfully established over UDP port 5000.

10. Check the routing table on each VPN server by typing the command below:

```
moxa@MOXA:~# route
Destination     Gateway Genmsk          Flags   Metric  Ref Use Iface
192.168.5.0     0.0.0.0 255.255.255.0   U       0       0   0   eth2
192.168.4.0     0.0.0.0 255.255.255.0   U       0       0   0   br0
192.168.3.0     0.0.0.0 255.255.255.0   U       0       0   0   eth0
192.168.30.0    0.0.0.0 255.255.255.0   U       0       0   0   eth3
192.168.8.0     0.0.0.0 255.255.255.0   U       0       0   0   br0
```

Interface **eth1** and device **tap0** both connect to the bridging interface, and the virtual device **tun** sits on top of **tap0**. This ensures that all traffic coming to this bridge from internal networks connected to interface eth1 write to the TAP/TUN device that the OpenVPN program monitors. Once the OpenVPN program detects traffic on the virtual device, it sends the traffic to its peer.

11. To create an indirect connection to Host B from Host A, you need to add the following routing item:

moxa@MOXA:~# route add –net 192.168.4.0 netmask 255.255.255.0 dev eth0

To create an indirect connection to Host A from Host B, you need to add the following routing item:

moxa@MOXA:~# route add –net 192.168.2.0 netmask 255.255.255.0 dev eth0

Now ping Host B from Host A by typing:

moxa@MOXA:~# ping 192.168.4.174

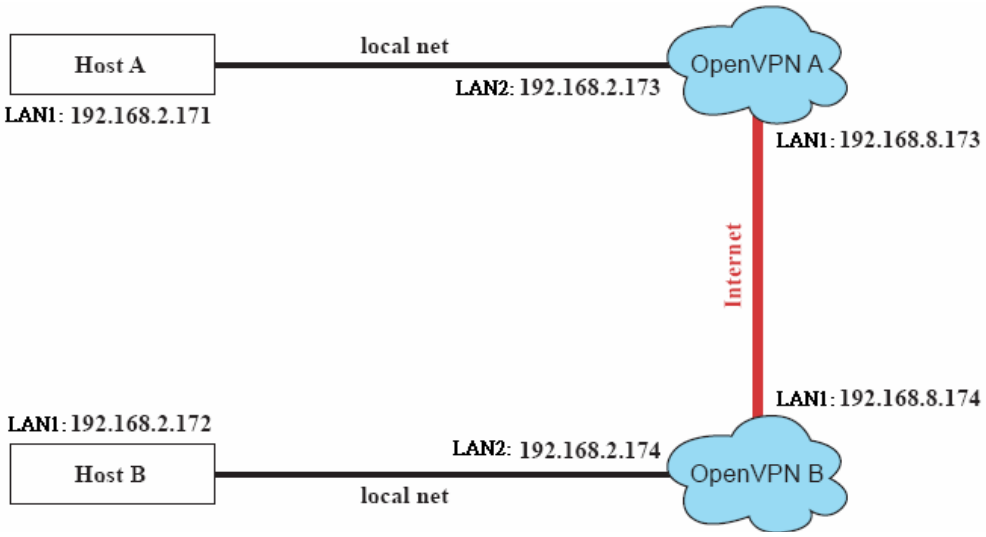
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

12. To shut down the VPN servers, use the **killall** command:

moxa@MOXA:~# killall -TERM openvpn

Ethernet Bridging for Private Networks on the Same Subnet

Like the last example, this setup will link two subnets across the oopen Ethernet; however, these two subnets will share addressing as if they were located on the same local subnet.



All of the clients on the two remote subnets are configured for a range of IP addresses that spans **the same subnet**. When this configuration is moved to a public network, the external interfaces of the OpenVPN machines must be configured for static IPs or connected to another device (such as a firewall or DSL box) that uses a static address.

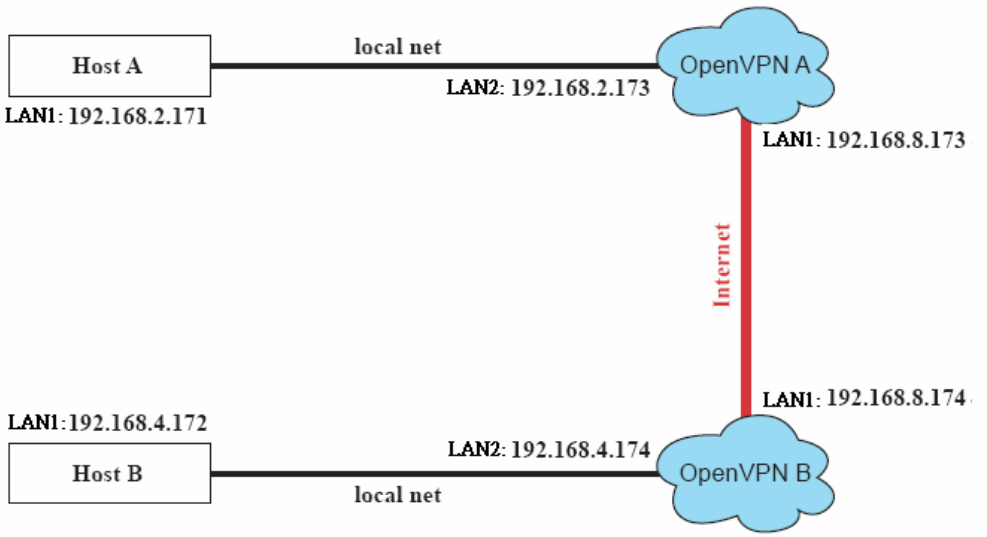
The configuration procedure for this setup is almost the same as for the previous example. The only difference is that you will need to comment out the parameter **up** in the `/etc/openvpn/tap0-br.conf` on each of the gateways, OpenVPN A and OpenVPN B.

```
# point to the peer
remote 192.168.8.174
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
#up /etc/openvpn/tap0-br.sh
#comp-lzo
```

Routed IP Tunnels

Routed IP tunnels are used to route point-to-point IP traffic without broadcasts; the advantage of routed IP tunnels is that they are slightly more efficient than bridged ethernet tunnels and easier to configure.

1. **Host A** represents the machine that belongs to the subnet served by the VPN server, **OpenVPN A**, and **Host B** represents a machine that belongs to the subnet served by the VPN server, **OpenVPN B**. The two remote subnets are configured for **distinct ranges** of private IP addresses on **separate subnets**.



2. On VPN server A (**OpenVPN A**), modify the **remote address** entry in the configuration file `/etc/openvpn/tun.conf` by adding the address of OpenVPN B. Also, you must add an **ifconfig** entry which indicates the local (1st) and remote (2nd) VPN gateway addresses, separated by a space.

```
# point to the peer
remote 192.168.8.174
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.2.173 192.168.4.174
up /etc/openvpn/tun.sh
```

3. Next, change OpenVPN A's `/etc/openvpn/tun.sh` so that the routing table matches the local subnet the VPN gateway is serving. Notice the **gw $5** appended to the end of this line: the **$5** is a variable argument that OpenVPN passes to the startup script. Its value is the second argument of **ifconfig** in the `/etc/openvpn/tun.conf` file.

```
#----------------------------Start-----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 gw $5
#-----------------------------end-----------------------------
```

4. On VPN server B (**OpenVPN B**), change the **remote address** in configuration file `/etc/openvpn/tun.conf` by adding the address of OpenVPN A. Also, you must add an **ifconfig** entry which indicates the local (1st) and remote (2nd) VPN gateway addresses, each separated by a space.

```
# point to the peer
remote 192.168.8.173
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
```

```
ifconfig 192.168.4.174 192.168.2.173
up /etc/openvpn/tun.sh
```

5. Next, change OpenVPN B's routing table in the file `/etc/openvpn/tun.sh` so that it matches the local subnet the VPN gateway is serving. Notice the **gw $5** appended to the end of this line: the **$5** is a variable argument that OpenVPN passes to the script file. Its value is the second argument of **ifconfig** in the **/etc/openvpn/tun.conf** file.

```
#--------------------------Start-----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 gw $5
#---------------------------end-----------------------------
```

6. Check the routing table after you run OpenVPN; it should show an established route running between your two VPN gateways. The command to see the routing table is:

   moxa@Moxa:~# route.

```
Destination     Gateway         Genmsk          Flags   Metric  Ref Use
    Iface
192.168.4.174   *               255.255.255.255 UH      0       0   0   tun0
192.168.4.0     192.168.4.174   255.255.255.0   UG      0       0   0   tun0
192.168.2.0     *               255.255.255.0   U       0       0   0   eth1
192.168.8.0     *               255.255.255.0   U       0       0   0   eth0
```

Setting Up Hot Swap for Block Storage

The TC-6110-LX computers come with two removable trays for additional block storage devices like hard disks or SSD drives. It also supports hot swapping capability along with user-defined programmable LEDs and a related API for convenient storage management.

File Overview

- **mxhtspd:** a daemon for monitoring hot-swap events
- **mxhtspd-setled:** a command to set up LED signals
- **/etc/mxhtspd/scripts:** scripts executed when an event occurs; the following files are included:
  - action-btn -pressed
  - action-disk-plugged
  - action-disk-unplugged
  - action-part-over-usage
- **/etc/mxhtspd/mxhtspd.conf:** configuration file for the mxhtspd daemon
- **libmxhtsp.so:** library

Hot Swap Daemon Customization

The **mxhtspd** daemon—or the Moxa hot-swap daemon—manages block-storage hot-swapping, and is pre-enabled on this device. It is launched in the background at boot-time by the `/etc/init.d/mxhtspd.sh` script, and will automatically detect drive status so long as the machine is running.

Users can configure how **mxhtspd** is called by editing up the **/etc/init.d/mxhtspd.sh**; mxhtspd accepts the following arguments:

- :**-i [INTERVAL_IN_SECONDS]:** interval in seconds to check partition usage

- **-l [FACILITY_LEVEL]:** the facility level is used to specify what type of program is logging the message. This enables rsyslog to handle messages from different facilities (e.g., FTP, UUCP, cron, etc.) in different ways.
- **-v:** run in verbose mode
- **-h:** show command help / print command arguments

The following example shows how to use the –v option to modify `/etc/init.d/mxhtspd.sh`:

```
…
start)
       echo "Starting mxhtspd daemon..."
       sleep 1
       mxhtspd -v &
…
```

The mxhtspd daemon also provides the capability to monitor partition usage. You can configure the `/etc/mxhtspd/mxhtspd.conf` file to monitor a configured mount point like `/media/disk1p1`. Note that `disk`**`N`**`p`**`M`** is the ***M***th partition of hotswap disk ***N***.

An example of **mxhtspd.conf** is shown below:

```
#mount point   usage limit(%)
/media/disk2p1 90
/media/disk1p1 90
…
```

With the setup above, once more than 90% of partition 1 on either drive 1 or drive 2 is allocated, a (user-defined) event will be triggered.

Handling an Event with mxhtspd: Moxa Hot-Swap Daemon

mxhtspd is automatically triggered whenever one of the following events occurs:

1. **A block storage device module is plugged into a drive slot**
2. **The X1 button on the spine of a drive module is pressed**
3. **A drive is removed without being properly unmounted**
4. **A monitored partition has reached or is over a specified allocation limit; or it does not exist**

These events are explained in more detail below.

1. **A block storage device is plugged into the drive slot**

   When drive ***N*** with ***M*** partitions has been plugged in, the system will automatically mount its partitions on `/media/disk`***N***`p`***M***, where ***M*** ranges from **1** to ***M***. For example, if disk 1 has two partitions, they would be mounted on **`/media/disk1p1`** and **`/media/disk1p2`**.

   The script **action-disk-plugged.sh** will be called by **mxhtspd** with the appropriate drive number. The action-disk-plugged script will run a scan on the drive, and users may append operations here if they wish.

   Use the **mount** command to return the currently mounted partitions; the code below shows (in red) what 2 mounted drives, each with 2 partitions, will look like when mounted in a standard configuration.

```
moxa@Moxa:~# mount
rootfs on / type rootfs (rw)
none on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
none on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type tmpfs (rw,relatime,size=10240k,mode=755)
/dev/hda1 on / type ext2 (ro,relatime,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,relatime,mode=755)
usbfs on /proc/bus/usb type usbfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,relatime)
```

```
devpts on /dev/pts type devpts
(rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
none on /tmp type tmpfs (rw,relatime)
/dev/hda2 on /home type ext2 (rw,relatime,errors=continue)
/dev/sda1 on /media/disk2p1 type ext3
(rw,relatime,errors=continue,data=ordered)
/dev/sda2 on /media/disk2p2 type ext3
(rw,relatime,errors=continue,data=ordered)
/dev/sdb1 on /media/disk1p1 type ext3
(rw,relatime,errors=continue,data=ordered)
/dev/sdb2 on /media/disk1p2 type ext3
(rw,relatime,errors=continue,data=ordered)
```

2. **The X1 button on the spine of a drive module is pressed**

   On the spine of each module there is an inset button labeled X1; this button may be custom-configured, but by default it signals the OS to un-mount the hard drive so that the module may be safely removed. Users should use a small screwdriver or other sharply tipped instrument to gently depress the X1 button.

   When the X1 button is pressed less than 5 seconds (i.e.: a short press), the **`/etc/mxhtspd/action-btn-pressed`** script will be executed. It will umount all partitions on that module, and when it is completed the LEDs on the module spine will blink 3 times, at 1 second intervals, to indicate that the disk has been successfully unmounted. Users may then remove the module from the slot.

3. **A drive is removed without being properly unmounted**

   The only purpose of this script is to warn of misuse through incorrect operation of the drive modules.

   When disk *n* is unplugged, the **`/etc/mxhtspd/action-disk-unplugged`** script will be triggered with argument *n*. It will check if all partitions on the module were umounted before they were unplugged and warn the user if they weren't. Users should remember: the correct procedure to remove a module is to first depress the X1 button on the spine of the module, and wait for the partitions to unmount. Only then should they remove the module.

4. **A monitored partition has reached or is over a specified allocation limit; or it does not exist**

   When /media/disk***N***p***M*** has allocated its storage space to the limit defined in **`/etc/mxhtspd/mxhtspd.conf`**, **mxhtspd** will display an error message and blink the warning LED ***N*** at 1 second intervals. In addition, the **`/etc/mxhtspd/action-part-over-usage`** script will be launched with argument **`/media/diskNpM`**.

WARNING!

Be sure to press the X1 button on the spine of the storage module before removing it. After depressing the button (for which you should use a small screwdriver, or other pointed instrument), the LEDs will blink 3 times to indicate that the disks have been successfully unmounted. You may then safely remove the drive modules from the computer.

Setting Up Hot Swap Daemon Logging

This section describes associate **mxhtspd** log messages with the **rsyslogd** daemon.

1. Check the system run level by running the command:

```
moxa@Moxa:~# sudo runlevel
N 2
```

2. If the runlevel is 2, then skip this step. If it is not, then either modify inittab (`/etc/inittab`) so that the default runlevel is 2, or add a symlink that will call **mxhtspd** to the runlevel for which the machine is currently configured. To set the inititialization table (/etc/inittab) to runlevel 2, edit two lines near the top so that they look like this:

```
# The default runlevel.
id:2:initdefault:
```

3. Enable **rsyslogd** at startup by editing the symlink located in the appropriate runlevel control directoriy (here, we are using the default runlevel 2):

```
Moxa:/etc/rc2.d# mv N10rsyslog S10rsyslog
```

4. You may set syslog to report mxhtspd as a specialized facility level, but the POSIX standard suggests you set it to be recognized as a daemon. To set the system logger (syslog) to report **mxhtspd** as a **daemon**, set the **/etc/init.d/mxhtspd.sh** script to set mxhstpd with facility level 3:

```
#Add parameter if necessary
mxhtspd -l 3&
```

ATTENTION

You may refer to the Wikipedia page for a good overview of the syslog system and facility and severity levels:: **http://en.wikipedia.org/wiki/Syslog** (Dec. 2013)

5. Edit the configuration file **/etc/rsyslogd.conf**.

```
#Uncomment below lines for mxhtspd with local 0
daemon.*                    -/var/log/mxhtspd.log
```

mxhtspd will now log messages as a daemon facility. The destination file is /var/log/mxhtspd.log. The minus (-) sign indicates to omit synching the file after every logging.

6. Restart your computer to activate the settings.

ATTENTION

When you run rsyslogd daemon to log messages at startup, take care to prevent excessive disk usage.

A Sample mxhtspd Setup

In this section we show an example that illustrates how to deploy **mxhtspd**. The program is named **log_application**, and its purpose is to collect important data and save the data to the permanent storage drive once daily. The following settings will illustrate how to identify that the drive is full, and how to start or stop the application without using an external monitor.

1. Whenever a module's X1 button (located on the spine of the module) is pressed for longer than 5 seconds, the **action_btn_pressed** script corresponding to that module is called. By adding the following lines in the appropriate **action_btn_pressed**, whenever button X1 is pressed longer than 5 seconds the log_application program will be triggered.

```
#!/bin/sh
file=`basename $0`
num=$1

#Add your commands here
/home/log_application

echo "$file: Button $num is pressed"
```

2. Whenever a module's X1 button is pressed for shorter than 5 seconds, the **action_btn_short_pressed** script corresponding to that module is called. The script below sets the action for a short press to stop the daemon so that the disk may be removed. In the **action_btn_short_pressed** file, you should kill the

log_application script before the disk is umounted. The command **pidof** returns the process i.d. of the program referenced in the argument position.

```
#!/bin/sh
file=`basename $0`
num=$1

#Add your commands here
kill -9 `pidof -x log_application`

#Internal operation
mxhtspd-remove-disk $num
```

Setting Up GPS

The National Marine Electronics Association (NMEA) has developed a specification for a communications interface that links marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment. GPS receivers communicate via the NMEA interface, and computer programs expect data to be in the NMEA format. This data includes the complete PVT (position, velocity, time) solution computed by the GPS receiver.

Retrieving GPS Data

The GPS kernel module is precompiled into the Linux kernel binary, so there is no need to install one. GPS hardware is read and managed by the **gpsd** daemon, and accessed through the port `/dev/ttyACM0`. Follow these steps for installation:

1. First check if the GPS card is transmitting raw data by issuing the following command to the device node, /dev/ttyACM0. If no data is being returned by the card, try adjusting the GPS antenna to troubleshoot the problem. If there is no way of establishing reception, contact Moxa technical support at the phone number provided in the title plate of this manual.

```
moxa@Moxa:~# cat /dev/ttyACM0
$GPGSV,1,1,04,24,28,123,37,21,09,054,31,19,52,213,,23,47,270,*74
$GPGGA,061824.0,2458.835139,N,12133.055835,E,1,05,19.7,-103.5,M,,,,*14
$GPRMC,061824.0,A,2458.835139,N,12133.055835,E,,,290710,,,A*68
$GPGSA,A,3,24,21,06,31,16,,,,,,,,25.5,19.7,18.5*29
$GPVTG,,T,,M,0.0,N,0.0,K*4E
```

2. Next, install GPSd, the GPS background interface that will communicate with the raw GPS device. First, terminate the cat process you have just initiated using either `<ctrl> + c`, or :

   moxa@Moxa:~# killall cat

   and then install the GPS daemon using the command `moxa@Moxa:~# apt-get install gpsd`:

```
moxa@Moxa:~$ sudo apt-get install gpsd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libbluetooth3 libgps20
Suggested packages:
  gpsd-clients
The following NEW packages will be installed:
  gpsd libbluetooth3 libgps20
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 425 kB of archives.
After this operation, 946 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

3. You may change the options with which **gpsd** will be called by using the Debian package installer. To reconfigure how gpsd is called, use the command:

```
moxa@Moxa:~# dpkg-reconfigure gpsd
```

If necessary, you may edit the gpsd configuration file by hand; the file is located at /etc/default/gpsd. However, Debian recommends against this, and there is the possibility that whenever the gpsd package is updated you will lose your settings. If possible, use the dpkg-reconfigure command shown above.

```
# Default settings for gpsd.
# Please do not edit this file directly - use `dpkg-reconfigure gpsd' to
# change the options.
START_DAEMON="true"
GPSD_OPTIONS=""
DEVICES="/dev/ttyACM0"
USBAUTO="true"
GPSD_SOCKET="/var/run/gpsd.sock"
```

4. After gpsd is configured, start the daemon**:**

```
moxa@Moxa:~# sudo /etc/init.d/gpsd start
```

5. You should now be able to fetch GPS data via /dev/ttyACM0 device node.

```
root@Moxa:/home/moxa# cat /dev/ttyACM0
$GPTXT,01,01,02,u-blox ag - www.u-blox.com*50
$GPTXT,01,01,02,HW  UBX-G60xx  00040007 EFFBFFFFp*25
$GPTXT,01,01,02,ROM CORE 7.03 (45969) Mar 17 2011 16:18:34*59
$GPTXT,01,01,02,ANTSUPERV=AC SD PDoS SR*20
$GPTXT,01,01,02,ANTSTATUS=OK*3B
$GPRMC,121714.00,A,2459.07353,N,12133.14090,E,0.205,,150813,,,A*7F
$GPVTG,,T,,M,0.205,N,0.380,K,A*2F
$GPGGA,121714.00,2459.07353,N,12133.14090,E,1,06,2.70,39.5,M,17.1,M,,*61
$GPGSA,A,3,10,13,04,09,02,07,,,,,,,3.88,2.70,2.79*03
$GPGSV,4,1,13,02,55,355,36,04,48,065,38,05,55,285,21,07,05,093,36*7E
$GPGSV,4,2,13,08,03,120,,09,08,125,23,10,48,025,40,12,21,258,18*7B
$GPGSV,4,3,13,13,18,043,32,17,24,150,08,25,08,296,,26,28,186,*7E
$GPGSV,4,4,13,29,00,323,*42
$GPGLL,2459.07353,N,12133.14090,E,121714.00,A,A*6F
$GPRMC,121715.00,A,2459.07338,N,12133.14081,E,0.067,,150813,,,A*75
$GPVTG,,T,,M,0.067,N,0.124,K,A*25
$GPGGA,121715.00,2459.07338,N,12133.14081,E,1,07,2.70,39.8,M,17.1,M,,*61
$GPGSA,A,3,10,13,17,04,09,02,07,,,,,,3.88,2.70,2.79*05
$GPGSV,4,1,13,02,55,355,36,04,48,065,38,05,55,285,21,07,05,093,36*7E
$GPGSV,4,2,13,08,03,120,,09,08,125,24,10,48,025,40,12,21,258,18*7C
$GPGSV,4,3,13,13,18,043,31,17,24,150,13,25,08,296,,26,28,186,*77
$GPGSV,4,4,13,29,00,323,*42
$GPGLL,2459.07338,N,12133.14081,E,121715.00,A,A*63
$GPRMC,121716.00,A,2459.07322,N,12133.14070,E,0.334,,150813,,,A*76
$GPVTG,,T,,M,0.334,N,0.619,K,A*29
```

6. You may use AWK to parse and reorder the NMEA string format:

```
awk -F, '/\$GPGGA/ {print  strftime("%Y-%m-%d "),  $2, (substr($3,0,2) +\
        (substr($3,3) / 60.0)) $4,  (substr($5,0,3) + (substr($5,4) / 60.0)) \
        $6, $10; fflush();}' /dev/ttyACM0
```

```
root@Moxa:/home/moxa# awk -F, '/\$GPGGA/ {print  strftime("%Y-%m-%d "),  $2, (substr($3,0,2
) + (substr($3,3) / 60.0)) $4,  (substr($5,0,3) + (substr($5,4) / 60.0)) $6, $10; fflush();
}' /dev/ttyACM0
2013-08-15  121804.00 24.9845N 121.552E 45.8
2013-08-15  121805.00 24.9845N 121.552E 45.8
2013-08-15  121806.00 24.9845N 121.552E 46.0
2013-08-15  121807.00 24.9845N 121.552E 46.2
2013-08-15  121808.00 24.9845N 121.552E 46.4
2013-08-15  121809.00 24.9845N 121.552E 46.7
2013-08-15  121810.00 24.9845N 121.552E 47.0
2013-08-15  121811.00 24.9845N 121.552E 47.2
2013-08-15  121812.00 24.9845N 121.552E 47.4
2013-08-15  121813.00 24.9845N 121.552E 47.6
2013-08-15  121814.00 24.9845N 121.552E 47.8
```

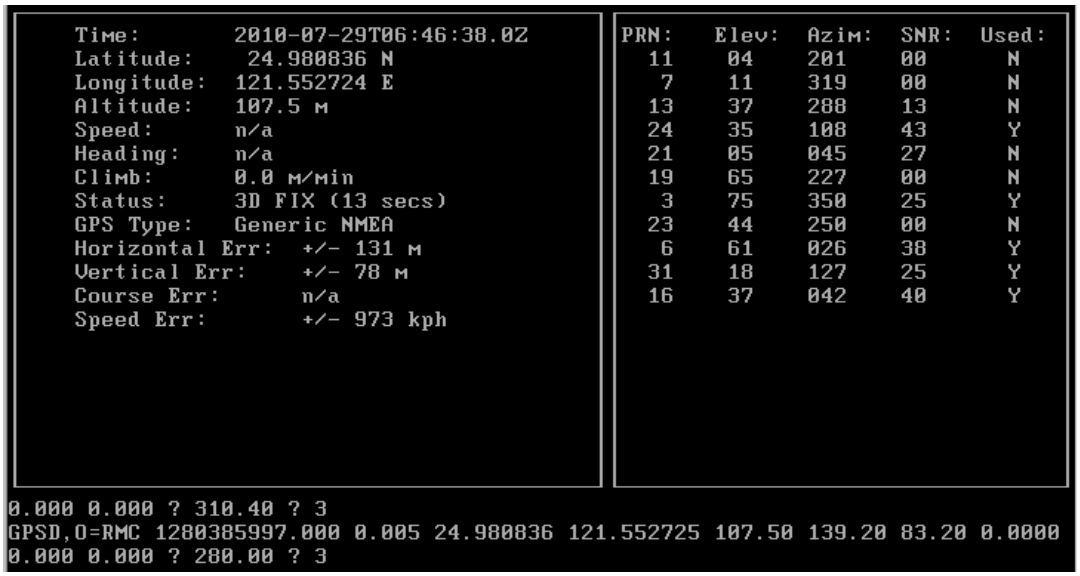
7. You may also install the GPS test clients **cgps** and **xgps**, which can beu used to query **gpsd**:

```
Moxa:~# apt-get install gpsd-clients
```

8. **Xgps** is used on the desktop, and **cgps** is used on the command line terminal, or over a serial emulator/interface. You may access either client by logging in remotely, using SSH or a virtual desktop. To get a basic report on the current GPS data, call **cgps** on the console:

```
moxa@Moxa:~# cgps
```

and you should a report that looks something like this:



NOTE Use the GNU system manual command to get more information on the GPS daemon interface and client:

```
moxa@Moxa:~# man gpsd
moxa@Moxa:~# man cgps
```

Or visit the GPS project website for more information: http://gpsd.berlios.de/

4

Moxa's Rcore Software Packages

This chapters describes Moxa's software packages that can be used on the TC-6110-LX computers.

The following topics are covered in this chapter:

- **Moxa Predictive Maintenance Diagnostic Tool**
  - Overview
  - Installing the Predictive Maintenance Diagnostic Tool
  - Moxa Rcore Predictive Maintenance Diagnostic Tool
  - The T-sensor Log
  - The Accelerometer (G-Sensor) Log
  - Removing the Moxa Predictive Maintenance Diagnostic Tool
- **Moxa SynMap Package**
  - Overview
  - Moxa SynMaP OIDs List
  - Installing Moxa Synmap
  - Using Moxa Synmap OIDs: snmpwalk & snmpget
  - Configuring the Programmable LEDs
  - Checking T-sensor
  - Checking Voltage Sensor
  - Checking G-sensor
  - Enabling the Watchdog

Moxa Predictive Maintenance Diagnostic Tool

Overview

The TC-6110 computers provide Moxa predictive maintenance diagnostic tool for you to monitor the system status of your disk drive. This package includes two tools: the G-Sensor chart tool and monitoring tool. The G-Sensor chart tool provides the display and log function for you to monitor the gravity value to predict the damage of hard disk, and the monitoring tool provide the system information(CPU frequency, disk drive temperature...etc).

Installing the Predictive Maintenance Diagnostic Tool

Follow these steps to install the G-sensor (vibration) package.

1. From the software CD that shipped with the computer or from the web, copy the **moxa-SafeGuard-TC-6110-1.0.deb** software package to your computer.
2. Install the software package using this command:

```
moxa@Moxa:~# sudo dpkg --force-all -i moxa-SafeGuard-TC-6110-1.0.deb
```

```
moxa@Moxa:~$ sudo dpkg --force-all -i moxa-SafeGuard-TC-6110-1.0.deb
Selecting previously unselected package moxa-safeguard-tc-6110.
(Reading database ... 32690 files and directories currently installed.)
Unpacking moxa-safeguard-tc-6110 (from moxa-SafeGuard-TC-6110-1.0.deb) ...
Setting up moxa-safeguard-tc-6110 (1.0) ...
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc0.d/K02mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc1.d/K02mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc2.d/S22mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc3.d/S22mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc4.d/S22mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc5.d/S22mx_safeguard
insserv: enable service ../init.d/mx_safeguard -> /etc/init.d/../rc6.d/K02mx_safeguard
insserv: creating .depend.boot
insserv: creating .depend.start
insserv: creating .depend.stop
Starting Moxa SafeGuard...
Starting Moxa SafeGuard OK
```

Moxa Rcore Predictive Maintenance Diagnostic Tool

The Moxa SafeGuard™ technology suite includes Rcore software to help administrators configure sensors for intelligent monitoring of vibration and temperature, giving system operators important data about operational conditions and possible causes of data corruption influencing the machine and its surrounding environs.

Initial Check:

1. Start Safeguard using **moxa@Moxa:~# sudo safeguard**. You should see the following display:

```
+--------------------Monitor tool-----------------------+
|                                                       |
|  ---------------------------------------------       |
|    Frequency  |    Voltage    | Temperature           |
|      (MHz)    |      (V)      |  (Celsius)            |
|  ---------------------------------------------       |
|     CPU: 1796 |     CPU: 1.12 | Disk1:  44.00         |
|  Memory:  667 |  CPU_IO: 1.04 | Disk2:  44.00         |
|               | Chipset: 1.48 |                       |
|               |  Memory: 1.48 |                       |
|  ---------------------------------------------       |
|                   G-Sensor(mG)                        |
|  ---------------------------------------------       |
|        [ Disk 1 ]     |     [ Disk 2 ]               |
|     x-axis: -0.003907  | x-axis:   0.015629          |
|     y-axis:  0.003907  | y-axis:   0.000000          |
|     z-axis: -0.007814  | z-axis:   0.003907          |
|  ---------------------------------------------       |
|                                                       |
+-------------------------------------------------------+
```

2. To exit, send a **SIGINT** using **Ctrl + C**, **killall**, or **kill**.

The T-sensor Log

The configuration file for the temperature sensor log files is `/etc/mxsensor.conf`. The temperature sensor logs may be configured for temperature scale (Celsius or Fahrenheit) and log mechanics (on/off, file path, logging interval, and max. file size).

1. The temperature sensor configuration file is the same as the accelerometer configuration file: mxsensor.conf. In the screenshot below you see the portion of the configuration file relevant to the temperature sensor. Edit this to configure how the temperature sensor will be logged.

   ```
   moxa@Moxa:~# sudo vi /etc/mx_sensor.conf
   ```

   ```
   # Celsius(°C) = 1, Fahrenheit(°F) =2.
   # default=1
   TEMPERATURE_SCALE=1

   # Temperature log status
   # turn off log: 0
   # start to log: 1
   TEMPERATURE_LOG_STATUS=0

   # Log path
   TEMPERATURE_1_LOG_PATH=/var/log/tsensor_1.log
   TEMPERATURE_2_LOG_PATH=/var/log/tsensor_2.log

   # unit: MB
   # default: 100 MB
   TEMPERATURE_LOG_SIZE=100

   # unit: second
   # default: 1s
   TEMPERATURE_LOG_INTERVAL=1
   ```

2. Save the file and exit your editor (in VI, that is done by typing **:wq**).
3. Restart Safeguard so that the changes will take effect:

   ```
   moxa@Moxa:~# sudo /etc/init.d/mx_safeguard restart
   ```

4. If the tempature log status is set to 1 (enabled), a log file will be recorded at the specified path.
5. Below is a sample temperature log.

   ```
   2013-08-27 11:45:41, 43.00
   2013-08-27 11:45:42, 44.00
   2013-08-27 11:45:43, 43.00
   2013-08-27 11:45:44, 43.00
   2013-08-27 11:45:45, 43.00
   2013-08-27 11:45:46, 44.00
   2013-08-27 11:45:47, 44.00
   2013-08-27 11:45:49, 44.00
   2013-08-27 11:45:50, 43.00
   2013-08-27 11:45:51, 44.00
   2013-08-27 11:45:52, 43.00
   2013-08-27 11:45:53, 43.00
   2013-08-27 11:45:54, 44.00
   2013-08-27 11:45:55, 43.00
   2013-08-27 11:45:56, 44.00
   2013-08-27 11:45:57, 44.00
   ```

The Accelerometer (G-Sensor) Log

The configuration file for the temperature sensor log files is `/etc/mxsensor.conf`. The temperature sensor logs may be configured for log mechanics (on/off, file path, max. file size, and logging interval).

1. The temperature sensor configuration file is the same as the accelerometer configuration file: mxsensor.conf. In the screenshot below you see the portion of the configuration file relevant to the temperature sensor. Edit this to configure how the temperature sensor will be logged.

```
moxa@Moxa:~# sudo vi /etc/mx_sensor.conf.
```

```
# G-sensor log status
# turn off log: 0
# start to log: 1
GSENSOR_LOG_STATUS=0

# Log path
GSENSOR_1_LOG_PATH=/var/log/gsensor_1.log
GSENSOR_2_LOG_PATH=/var/log/gsensor_2.log

# unit: MB
# default: 100 MB
GSENSOR_LOG_SIZE=100

# unit: second
# default: 1s
GSENSOR_LOG_INTERVAL=1
```

2. Save the file and exit your editor (in VI, that is done by typing **:wq**).
3. Restart Safeguard so that the changes will take effect:

```
moxa@Moxa:~# sudo /etc/init.d/mx_safeguard restart
```

4. If the accelerometer log status is set to 1 (enabled), a log file will be recorded at the specified path.
5. Below is a sample accelerometer log.

```
2013-08-27 11:42:44, -0.003907, 0.007814, 0.000000
2013-08-27 11:42:45, 0.000000, 0.007814, 0.007814
2013-08-27 11:42:46, 0.000000, 0.003907, 0.000000
2013-08-27 11:42:47, 0.000000, 0.003907, 0.003907
2013-08-27 11:42:48, 0.000000, 0.003907, 0.000000
2013-08-27 11:42:49, -0.003907, 0.007814, 0.015629
2013-08-27 11:42:51, -0.003907, 0.007814, 0.000000
2013-08-27 11:42:52, -0.003907, 0.000000, 0.003907
2013-08-27 11:42:53, 0.000000, 0.007814, 0.003907
2013-08-27 11:42:54, -0.003907, 0.000000, 0.007814
2013-08-27 11:42:55, -0.003907, 0.007814, 0.000000
2013-08-27 11:42:57, -0.003907, 0.007814, -0.003907
2013-08-27 11:42:58, 0.000000, 0.003907, 0.007814
2013-08-27 11:42:59, 0.000000, 0.000000, 0.000000
2013-08-27 11:43:00, -0.003907, -0.003907, -0.003907
2013-08-27 11:43:01, -0.003907, 0.007814, 0.003907
2013-08-27 11:43:02, 0.000000, 0.003907, 0.000000
2013-08-27 11:43:04, 0.000000, 0.007814, 0.000000
```

Removing the Moxa Predictive Maintenance Diagnostic Tool

Use **dpkg**, **apt**, or **aptitude** to remove Safeguard. The dpkg is:

```
moxa@Moxa:~# dpkg -r moxa-SafeGuard-TC-6110
```

```
root@Moxa:/home/moxa# dpkg -r moxa-SafeGuard-TC-6110
(Reading database ... 32706 files and directories currently installed.)
Removing moxa-safeguard-tc-6110 ...
Stopping Moxa SafeGuard ...
Stopping Moxa SafeGuard OK
```

Moxa SynMap Package

Overview

SynMap is Moxa's revolutionary software virtualization, an evolutionary advance in network device control that adapts solid, reliable SNMP into a fully portable remote procedure interface. SynMap allows engineers to automate remote processes using SNMP object identifiers (OIDs) rather than device-specific addresses, making a scripted SynMap procedure fully interoperable with any other SynMap device. This means that a script created for one SynMap device may be directly copied to another, immediately conferring the same functionality. This eliminates the need for rewriting and compiling code for newly configured devices, significantly reducing maintenance and deployment times.

SNMP is lightweight and easy-to-configure, and is already long popular with IT professionals. SNMP also enjoys comprehensive native support in high-level languages like .NET, Java, Python, or Ruby. For these reasons, the SynMap framework has re-imagined SNMP as a universal configuration and control interface for remote procedures, adapting it to not only monitor and control device internals like temperature, BIOS parameters, and local interfaces, but also to report on and automate tasks at the process layer, as well. Easily integrated into any existing Network Management System (NMS), SynMap devices are a flexible and cost-effective upgrade that returns obvious benefits to any IA network.

SynMap currently allows you to use SNMP for remote monitoring and control of a select set of computer processes, but its list of features is rapidly growing. Using SynMap's fully portable scripts, engineers will soon be able to:

- Access, monitor, control, and report on digital I/O at both the process and hardware layers
- Use OIDs to monitor, configure, and give process control over serial ports and other interfaces
- Monitor and control system attributes and process events via any NMS
- Build automated remote procedures using SynMap OIDs called by simple shell scripts, or a preferred high-level language like Python, Perl, or VBScript—all without any need for low-level C APIs, or platform-specific libraries
- Significantly simplify and reduce development times for custom utilities and automated executables
- Gain scripting and automation independence from OS-dependent libraries

All of this may be achieved using the simple, reliable, and familiar SNMP, the easily accessible standard every IT engineer knows. Discover how Moxa is expanding automation frontiers with the innovation we call SynMap.

Moxa SynMaP OIDs List

The following table shows the support OIDs on TC-6110, to review full Moxa SynMap OIDs, check the Appendix Section.

Item Name	**OID**	**File Access**	**Description**
productName	**1.3.6.1.4.1.8691.17.1.1.1**	**read-only**	Returns the product name
productDesc	**1.3.6.1.4.1.8691.17.1.1.2**	**read-only**	Returns a short device description
productVersion	**1.3.6.1.4.1.8691.17.1.1.3**	**read-only**	Returns the product version
productBuildDate	**1.3.6.1.4.1.8691.17.1.1.4**	**read-only**	Returns the product's last build date, in format YYMMDDHH.
systemCpuUsage	**.1.3.6.1.4.1.8691.17.1.2.1.1.0**	**read-only**	Returns current CPU usage (0-100 %)
systemMemUsage	**.1.3.6.1.4.1.8691.17.1.2.1.3.0**	**read-only**	Returns current memory usage (0-100 %)
systemUptime	**.1.3.6.1.4.1.8691.17.1.2.1.5.0**	**read-only**	The amount of time since the host was last initialized.
systemTotalUptime	**.1.3.6.1.4.1.8691.17.1.2.1.6.0**	**read-only**	The total amount of running

			time to date, since the beginning of time
systemMemorySize	**.1.3.6.1.4.1.8691.17.1.2.3.1.0**	**read-only**	The total amount of physical main memory
systemVolumeCount	**.1.3.6.1.4.1.8691.17.1.2.3.2.0**	**read-only**	Returns the total number of block storage volumes
systemVolumeIndex	**.1.3.6.1.4.1.8691.17.1.2.3.3.1.1**	**read-only**	Reference index for each block storage device
systemVolumeName	**.1.3.6.1.4.1.8691.17.1.2.3.3.1.2**	**read-only**	Returns a volumen name
systemVolumeLabel	**.1.3.6.1.4.1.8691.17.1.2.3.3.1.3**	**read-only**	Returns a volume label
systemVolumeSize	**.1.3.6.1.4.1.8691.17.1.2.3.3.1.4**	**read-only**	Returns the total size of a a block storage device
systemVolumeAvail	**.1.3.6.1.4.1.8691.17.1.2.3.3.1.5**	**read-only**	Returns the unallocated space of a block storage device
tempSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.1**	**read-only**	Returns a list of numbers that correspond with the temperature sensors; used by SNMP for identification, begins with 1
tempSensorsDevice	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.2**	**read-only**	Returns a list of string values identifying the temp sensors by name/location
tempSensorsValue	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.3**	**read-only**	Returns the temperature reading of a sensor in milli-C.
voltSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.1**	**read-only**	Returns a list of numbers that correspond with the voltage sensors; used by SNMP for identification, begins with 1
voltSensorsDevice	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.2**	**read-only**	Returns a list of string values identifying the voltage sensors by name/location. Possible values are **Vcore, V1.05, V1.5_S3, V1.5.** □
voltSensorsValue	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.3**	**read-only**	Returns voltage in millivolts
accelerometerIndex	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.1**	**read-only**	Returns a list of numbers that correspond with available accelerometers; used by SNMP for identification, begins with 1
accelerometerAxis	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.2**	**read-only**	Returns the name of an accelerometer axis
accelerometerValue	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.3**	**read-only**	Returns the accelerometer value in milli-Gs.
accelerometerTimestamp	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.4**	**read-only**	Returns the current timestamp at the accelerometer
ledNumber	**.1.3.6.1.4.1.8691.17.1.6.2.1.0**	**read-only**	
ledIndex	**.1.3.6.1.4.1.8691.17.1.6.2.1.0**	**read-only**	Returns a list of numbers that correspond with LEDs on the front panel; used by SNMP for identification, begins with 1
ledPort	**.1.3.6.1.4.1.8691.17.1.6.2.2.1.2**	**read-only**	
ledValue	**.1.3.6.1.4.1.8691.17.1.6.2.2.1.3**	**read-write**	Returns/Changs the status of the LED: off, or on
usbNumber	**.1.3.6.1.4.1.8691.17.1.6.4.1.1.0**	**read-only**	Returns the total number of

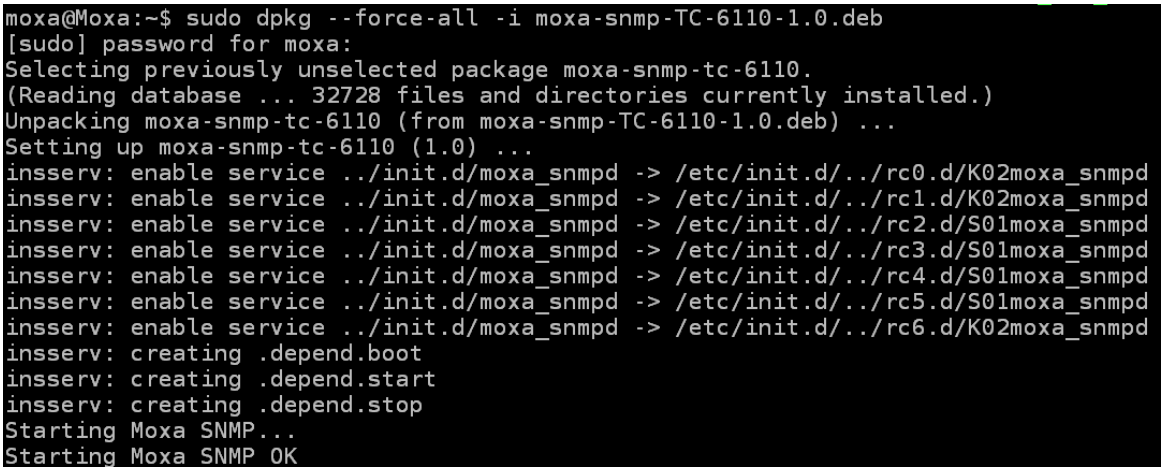
			USB ports, regardless of their current state in the usb general port table
usbDeviceIndex	**.1.3.6.1.4.1.8691.17.1.6.4.1.3.1.1**	**read-only**	The index is identical to usbPortIndex for the correspondent USB port
usbDeviceVendorID	**.1.3.6.1.4.1.8691.17.1.6.4.1.3.1.2**	**read-only**	The hexadecimal vendor string of the connected USB device as it is provided to the USB host
usbDeviceProductID	**.1.3.6.1.4.1.8691.17.1.6.4.1.3.1.3**	**read-only**	The hexadecimal product ID as it is provided to the USB host by the USB device.
usbDeviceIndex	**.1.3.6.1.4.1.8691.17.1.6.4.1.3.1.4**	**read-only**	This object returns the USB device class of the active configuration
watchdogPeriod	**.1.3.6.1.4.1.8691.17.1.6.6.2.1.0**	**read-write**	The configured watchdog period; 0 indicates the watchdog is disabled. The watchdog period is an integer between 0 and 255 that configures the watchdog expiration time in intervals of 1 second, with 255 representing an interval of 255 seconds.
watchdogStatus	**.1.3.6.1.4.1.8691.17.1.6.6.2.2.0**	**read-write**	Returns the watchdog's status.

Installing Moxa Synmap

To use Moxa Synmap, you need to install the Synmap package.

1. Copy the Synmap software package **moxa-snmp-TC-6110-1.0.deb** from the CD or the web.
2. Install the Synmap software package; because some libraries are shared, use the **`--force-all`** option.

```
moxa@Moxa:~# sudo dpkg --force-all -i moxa-snmp-TC-6110-1.0.deb.
```



Using Moxa Synmap OIDs: snmpwalk & snmpget

1. The command **snmpwalk** uses SNMP GETNEXT requests to query a network or device for the entire OID tree. An OID may be given on the command linke; this OID specifies which portion of the OID tree will be searched. All variables in the subtree below the given OID are then queried and their values presented to the user.
In this example, we walk the OID space beginning with the first OID in the Moxa MIB, **1.3.6.1.4.1.8691.17**:

moxa@Moxa:~# snmpwalk –v 2c –c public 192.168.XX.XXX 1.3.6.1.4.1.8691.17

```
root@Lock-Lin:~# snmpwalk -v 2c -c public 192.168.27.116 1.3.6.1.4.1.8691.17
iso.3.6.1.4.1.8691.17.1.1.1.0 = STRING: "TC-6110"
iso.3.6.1.4.1.8691.17.1.1.2.0 = STRING: "Moxa embedded computer"
iso.3.6.1.4.1.8691.17.1.1.3.0 = STRING: "1.0"
iso.3.6.1.4.1.8691.17.1.1.4.0 = STRING: "13081910"
iso.3.6.1.4.1.8691.17.1.2.1.1.0 = INTEGER: 2
iso.3.6.1.4.1.8691.17.1.2.1.3.0 = INTEGER: 2
iso.3.6.1.4.1.8691.17.1.2.1.5.0 = INTEGER: 270
iso.3.6.1.4.1.8691.17.1.2.1.6.0 = INTEGER: 270
iso.3.6.1.4.1.8691.17.1.2.3.1.0 = INTEGER: 2062500
iso.3.6.1.4.1.8691.17.1.2.3.2.0 = INTEGER: 1
iso.3.6.1.4.1.8691.17.1.2.3.3.1.1.1 = INTEGER: 1
iso.3.6.1.4.1.8691.17.1.2.3.3.1.2.1 = STRING: "/dev/sdd1"
iso.3.6.1.4.1.8691.17.1.2.3.3.1.3.1 = STRING: "TRANSCEND"
```

2. Use a Linux PC and type **snmpget** command to fetch a specific OID value.

```
root@Lock-Lin:~# snmpget -v 2c -c public 192.168.27.116 iso.3.6.1.4.1.8691.17.1.1.1.0
iso.3.6.1.4.1.8691.17.1.1.1.0 = STRING: "TC-6110"
```

Configuring the Programmable LEDs

The following figure shows the locations of the LED indicators on the TC-6110 computer. You can set these LED for your own application.

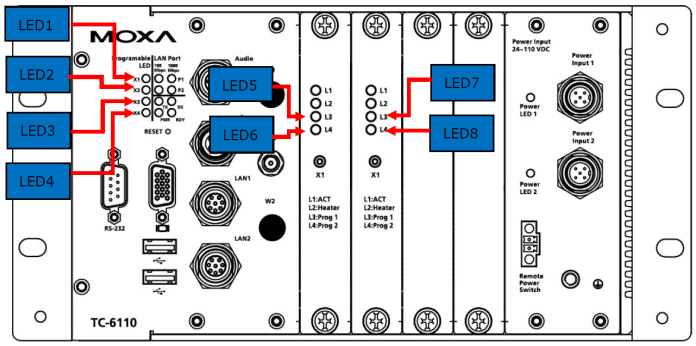
Item Name	OID	File Access	Value to set
ledNumber	**.1.3.6.1.4.1.8691.17.1.6.2.1.0**	**read-only**	**N/A**
ledIndex	**.1.3.6.1.4.1.8691.17.1.6.2.1.0**	**read-only**	Returns a list of numbers that correspond with LEDs on the front panel; used by SNMP for identification, begins with 1
ledPort	**.1.3.6.1.4.1.8691.17.1.6.2.2.1.2**	**read-only**	**N/A**
ledValue	**.1.3.6.1.4.1.8691.17.1.6.2.2.1.3**	**read-write**	**0(off), 1(on)**

1. Use a Linux PC and type **snmpset** command to turn the LED off on by re-setting the ledValue:

```
root@Lock-Lin:~# snmpset -v 2c -c private 192.168.27.116 .1.3.6.1.4.1.8691.17.1.6.2.2.1.3.1 i 1
iso.3.6.1.4.1.8691.17.1.6.2.2.1.3.1 = INTEGER: 1
```

2. Check if the LED indicator on the TC-6110 has been turned on.
3. Set value to 0 to turn off the LED, and then check if the LED indicator on the TC-6110 has been turned off.

```
root@Lock-Lin:~# snmpset -v 2c -c private 192.168.27.116 .1.3.6.1.4.1.8691.17.1.6.2.2.1.3.1 i 0
iso.3.6.1.4.1.8691.17.1.6.2.2.1.3.1 = INTEGER: 0
```



ATTENTION

If you modified PLED status via command **`echo 1111 > /dev/pled`**, the SNMP database wont be synched

Checking T-sensor

The following table shows the OIDS of the temperature sensors, read/write option and available values.

Item Name	OID	File Access
tempSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.1**	**read-only**
tempSensorsDevice	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.2**	**read-only**
tempSensorsValue	**.1.3.6.1.4.1.8691.17.1.5.1.1.1.3**	**read-only**

1. Use a Linux PC and type **snmpwalk** command to fetch all T-sensor OID value. The temperature is represented in milli-degrees, and defaults to Celsius (but can be reconfigured for Fahrenheit, if so desired).

```
root@Lock-Lin:~# snmpwalk -v 2c -c public 192.168.27.116 .1.3.6.1.4.1.8691.17.1.5.1.1.1
iso.3.6.1.4.1.8691.17.1.5.1.1.1.1.1 = INTEGER: 1
iso.3.6.1.4.1.8691.17.1.5.1.1.1.1.2 = INTEGER: 2
iso.3.6.1.4.1.8691.17.1.5.1.1.1.2.1 = STRING: "EXP1"
iso.3.6.1.4.1.8691.17.1.5.1.1.1.2.2 = STRING: "EXP2"
iso.3.6.1.4.1.8691.17.1.5.1.1.1.3.1 = Gauge32: 44000
iso.3.6.1.4.1.8691.17.1.5.1.1.1.3.2 = Gauge32: 44000
```

Checking Voltage Sensor

The following table shows the OID of temperature sensor, read/write option and available values.

Item Name	OID	File Access
voltSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.1**	**read-only**
voltSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.2**	**read-only**
voltSensorsIndex	**.1.3.6.1.4.1.8691.17.1.5.1.2.1.3**	**read-only**

1. Use a Linux PC and type **snmpwalk** command to fetch all voltage sensor OID value. The voltage is represented in millivolts.

```
root@Lock-Lin:~# snmpwalk -v 2c -c public 192.168.27.116 .1.3.6.1.4.1.8691.17.1.5.1.2.1
iso.3.6.1.4.1.8691.17.1.5.1.2.1.1.1 = INTEGER: 1
iso.3.6.1.4.1.8691.17.1.5.1.2.1.1.2 = INTEGER: 2
iso.3.6.1.4.1.8691.17.1.5.1.2.1.1.3 = INTEGER: 3
iso.3.6.1.4.1.8691.17.1.5.1.2.1.1.4 = INTEGER: 4
iso.3.6.1.4.1.8691.17.1.5.1.2.1.2.1 = STRING: "CPU"
iso.3.6.1.4.1.8691.17.1.5.1.2.1.2.2 = STRING: "CPU_IO"
iso.3.6.1.4.1.8691.17.1.5.1.2.1.2.3 = STRING: "Chipset"
iso.3.6.1.4.1.8691.17.1.5.1.2.1.2.4 = STRING: "Memory"
iso.3.6.1.4.1.8691.17.1.5.1.2.1.3.1 = Gauge32: 1120
iso.3.6.1.4.1.8691.17.1.5.1.2.1.3.2 = Gauge32: 1040
iso.3.6.1.4.1.8691.17.1.5.1.2.1.3.3 = Gauge32: 1487
iso.3.6.1.4.1.8691.17.1.5.1.2.1.3.4 = Gauge32: 1487
```

Checking G-sensor

The following table shows the OID of G-Sensor, read/write option and available values.

Item Name	OID	File Access
accelerometerIndex	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.1**	**read-only**
accelerometerAxis	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.2**	**read-only**
accelerometerValue	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.3**	**read-only**
accelerometerTimestamp	**.1.3.6.1.4.1.8691.17.1.5.1.3.1.4**	**read-only**

Use a Linux PC and type **snmpwalk** command to fetch all G-sensor OID value; vibration is returned in units of mill-Gees.

```
root@Lock-Lin:~# snmpwalk -v 2c -c public 192.168.27.116 .1.3.6.1.4.1.8691.17.1.5.1.3.1
iso.3.6.1.4.1.8691.17.1.5.1.3.1.1.1 = INTEGER: 1
iso.3.6.1.4.1.8691.17.1.5.1.3.1.1.2 = INTEGER: 2
iso.3.6.1.4.1.8691.17.1.5.1.3.1.2.1 = STRING: "x-axis, y-axis, z-axis"
iso.3.6.1.4.1.8691.17.1.5.1.3.1.2.2 = STRING: "x-axis, y-axis, z-axis"
iso.3.6.1.4.1.8691.17.1.5.1.3.1.3.1 = STRING: " -0.003907, 0.000000, 0.003907"
iso.3.6.1.4.1.8691.17.1.5.1.3.1.3.2 = STRING: " 0.003907, 0.011722, -0.015629"
iso.3.6.1.4.1.8691.17.1.5.1.3.1.4.1 = STRING: "2013-08-27 11:48:52"
iso.3.6.1.4.1.8691.17.1.5.1.3.1.4.2 = STRING: "2013-08-27 11:48:52"
```

Enabling the Watchdog

There are two watchdog parameters that may be configured: the **watchdogPeriod** and the **watchdogStatus**. The watchdog period is an integer between 0 and 255, with each number representing a single second. Watchdog status indicates and/or sets the watchdog to enabled or disabled (if disabled, the watchdog period will be represented as a zero [0]).

The default value of the watchdogPeriod is 60 (i.e., one minute), and the default value of watchdog status is 2, which means it is disabled. To set up the watchdog, you need to reset **watchdogStatus** to 1 and set the **watchdogPeriod** for the number of seconds that the watchdog will wait following a system hang. Once the the set period of time has passed during which the system remains inactive, the watchdog will automatically reboot the system. The following table shows the OIDs related to the watchdog.

Item Name	OID	File Access	Value to set
watchdogPeriod	**.1.3.6.1.4.1.8691.17.1.6.6.2.1.0**	**read-write**	An integer between 0 and 255, representing the time in sec that the watchdog will wait following a system hang.
watchdogStatus	**.1.3.6.1.4.1.8691.17.1.6.6.2.2.0**	**read-write**	1(Running), 2(Stop)

5

Programming Guide

The following topics are covered in this chapter:

- **RTC (Real Time Clock)**
- **UART**
- **WDT (Watch Dog Timer)**
  - Introduction
  - How the WDT Works
  - Examples
- **Hot-swapping Hard Disk**
- **Moxa SafeGuard**
  - Function Documentation

Desktop Management Interface (DMI)

Product information may be read using Debian Linux's Desktop Management Interface (DMI), **dmidecode**, as with the following commands:

```
moxa@MOXA:~# sudo dmidecode -s "baseboard-manufacturer"
MOXA
moxa@MOXA:~# sudo dmidecode -s "baseboard-serial-number"
TACCA1000000
```

Other keywords to retrieve DMI information are listed below:

```
bios-vendor
bios-version
bios-release-date
system-manufacturer
system-product-name
system-version
system-serial-number
system-uuid
baseboard-manufacturer
baseboard-product-name
baseboard-serial-number
baseboard-asset-tag
chassis-manufacturer
chassis-type
chassis-version
chassis-serial-number
chassis-asset-tag
processor-family
processor-manufacturer
processor-version
processor-frequency
```

RTC (Real Time Clock)

The device node for the RTC is located at **/dev/rtc**. The TC-6110-LX supports standard Linux simple RTC control. You must include **<linux/rtc.h>**.

1. The line below may be named Function: RTC_RD_TIME; it will read time information from the RTC. It will return the value on argument 3.
   ```
   int ioctl(fd, RTC_RD_TIME, struct rtc_time *time);
   ```
2. Function: RTC_SET_TIME will set the RTC time. Argument 3 will be passed to RTC.
   ```
   int ioctl(fd, RTC_SET_TIME, struct rtc_time *time);
   ```

UART

The normal tty device nodes are **/dev/ttyS0**. The TC-6110-LX supports standard Linux termios control with RS-232 **serial** ports.

Programmable LEDs

There are four programmable LEDs in the front of TC-6110. The LED device node is located at **/dev/pled**, and may be manipulated directly. Each LED can be accessed using the **/dev/pled** device node, and similarly the

SATA LEDs may be accessed directly using the **/dev/sata_pledX** device node. The examples below show you how.

Turning On or Off the LEDs

To turn on the first LED and turn off the second, third and fourth LED. You should.

```
moxa@Moxa:~# echo 1000 > /dev/pled
```

To turn off all the LED you should.

```
moxa@Moxa:~# echo 0000 > /dev/sata_pled1
```

To turn on the second LED and turn off the others. You should

```
moxa@Moxa:~# echo 0100 > /dev/pled
```

Turning On or Off the LEDs on a SATA Board

To turn on the first LED and turn off the second on SATA LED #1 . You should.

```
moxa@Moxa:~# echo 10 > /dev/sata_pled1
```

To turn off all the SATA LED #1you should.

```
moxa@Moxa:~# echo 00 > /dev/sata_pled1
```

To turn on the first LED and turn off the second on SATA LED #2 . You should.

```
moxa@Moxa:~# echo 10 > /dev/sata_pled2
```

To turn off all the SATA LED #1 you should.

```
moxa@Moxa:~# echo 00 > /dev/sata_pled2
```

Watch Dog Timer (WDT)

Introduction

The WDT can be enabled or disabled directly, using a shell script (if you wish to use the Synmap software package for this control, see the Synmap section above, **Enabling the Watchdog**). When the WDT is enabled and fails to receive a reset signal in the configured amount of time (1 to 255 seconds), the system will automatically reboot.

How the WDT Works

If you need the watchdog to run at boot time, use **moxa@MoxaL~# update-rc.d** to add it.

```
root@moxa@Moxa:~# update-rc.d watchdog defaults
update-rc.d: using dependency based boot sequencing
root@moxa@Moxa:~# ls -al /etc/rc?.d/*watchdog*
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc0.d/K01watchdog
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc1.d/K01watchdog
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc2.d/S21watchdog
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc3.d/S21watchdog
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc4.d/S21watchdog
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc5.d/S21watchdog
```

```
-> ../init.d/watchdog
lrwxrwxrwx 1 root root 18 Jul 16 11:23 /etc/rc6.d/K01watchdog
-> ../init.d/watchdog
```

The watchdog configuration file is **/etc/watchdog.conf**. By defauilt, the watchdog daemon is configured to signal the watchdog every 60 seconds.

```
…
watchdog-device = /dev/watchdog
…
interval             = 60
realtime             = yes
priority             = -10
…
```

The watchdog device IOCTL commands

IOCTL	WDIOC_GETSUPPORT
Description	This returns the support of the card itself
Input	None
Output	(struct watchdog_info *) arg
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_GETSTATUS
Description	This returns the status of the card
Input	None
Output	(int *)arg
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_GETBOOTSTATUS
Description	This returns the status of the card that was reported at bootup.
Input	None
Output	(int *)arg)
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_SETOPTIONS
Description	This lets you set the options of the card. You can either enable or disable the card this way.
Input	None
Output	(int *)arg)
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_KEEPALIVE
Description	This pings the card to tell it not to reset your computer.
Input	None
Output	None
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_SETTIMEOUT
Description	Sets the watchdog timeout
Input	arg: 1 ~ 255 seconds
Output	None
Return	On success, return 0. Otherwise, return < 0 value.

IOCTL	WDIOC_GETTIMEOUT
Description	Gets the current watchdog timeout.
Input	None
Output	arg: 1 ~ 255 seconds
Return	On success, return 0. Otherwise, return < 0 value.

Examples

This sample watchdog script, **watchdog-simple.c**, checks the watchdog every in 10 seconds.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(void)
{
	int fd = open("/dev/watchdog", O_WRONLY);
	int ret = 0;
	if (fd == -1) {
		perror("watchdog");
		exit(EXIT_FAILURE);
	}
	while (1) {
		ret = write(fd, "\0", 1);
		if (ret != 1) {
			ret = -1;
			break;
		}
		sleep(10);
	}
	close(fd);
	return ret;
}
```

Hot-Swapping Block Drives

A development library is provided to help you develop your applications. All of the code can be found at **/example/hotswap** on the software CD

Documentation Format

#define mxhtsp_close(fd) close(fd)	
Description	Closes the hotswap devices.
Parameters	*fd*: the open port
Returns	None

Function Documentation

int mxhtsp_check_partition_usage (const char * partition_name)	
Description	Gets what percentage of a partition is in use.
Parameters	*partition_name*: the name of the partition being checked. In linux, it should be

	/media/diskxpx
Returns	None

int mxhtsp_is_button_pressed (int fd, int btn_num)	
Description	Checks if a button is pressed.
Parameters	*fd*: the open port *btn_num*: the button number
Returns	1: pressed 0: not pressed -1: fail

int mxhtsp_is_disk_busy (int fd, int disk_num)	
Description	Checks if a disk is busy.
Parameters	*fd*: the open port *disk_num*: the disk number
Returns	1: busy 0: idle -1: fail

int mxhtsp_is_disk_plugged (int fd, int disk_num)	
Description	Checks if a disk is plugged in.
Parameters	*fd*: the open port *disk_num*: the disk number
Returns	1: plugged 0: unplugged -1: fail

Open the hotswap devices.	
Description	Checks if a disk is plugged in.
Returns	*fd* if successful -1: fail

int mxhtsp_set_led (int fd, int led_num, int on)	
Description	Sets the led to on/off.
Parameters	*fd*: the open port *led_num*: the led number 1 on 0 off
Returns	0: success -1: fail

Moxa SafeGuard

A development library is provided to help you develop your applications.

ATTENTION

There are two accelerometers in TC-6110-LX. The i2c address for accelerometer #1 is 0x1D, and 0x53 for accelerometer #2.

Function Documentation

int mx_accelerometer_read (GSENSOR_DATA *axis, unsigned char sensor_address)	
Description	Reads G sensor data from accelerometer
Parameters	axis: the 3-axis data structure sensor_address: the accelerometer i2c address
Returns	0: success -1: fail

int mx_accelerometer_get_state(unsigned char sensor_addr, unsigned char reg_addr)	
Description	Gets register value from specific accelerometer
Parameters	sensor_address: the i2c address of accelerometer reg_addr: the register in accelerometer
Returns	0: success -1: fail

int mx_accelerometer_set_state(unsigned char sensor_addr, unsigned char reg_addr, unsigned char value)	
Description	Sets register value to specific accelerometer
Parameters	sensor_address: the i2c address of accelerometer reg_addr: the register in accelerometer value: assigned value
Returns	0: success -1: fail

int mx_accelerometer_calibrate(unsigned char sensor_addr)	
Description	Sets calibration in specific accelerometer
Parameters	sensor_address: the i2c address of accelerometer
Returns	0: success -1: fail

int mx_accelerometer_clear_offset (unsigned char sensor_addr)	
Description	Clears calibration offset
Parameters	sensor_address: the i2c address of accelerometer
Returns	0: success -1: fail

Examples

The example file can power on accelerometer and calibrate it before reading data.

Library path: /lib/libmxdev.so

Header file path: /usr/include/mxdev.h

```
#include <stdio.h>
#include <stdlib.h>
#include <mxdev.h>

#define I2C_GSENSOR1_ADDR   0x1D
#define I2C_GSENSOR2_ADDR   0x53
```

```
#define DATA_RATE_REG      0x2C
#define POWER_CTL          0x2D
#define DATA_FORMAT_REG     0x31
#define MEASURE_BIT        0x08
#define DATA_RATE          0x0a
#define DATA_FORMAT        0x0b

int main(void)
{
   GSENSOR_DATA axis;
   int ret;
   mx_accelerometer_set_state(I2C_GSENSOR1_ADDR, POWER_CTL, MEASURE_BIT);
   mx_accelerometer_set_state(I2C_GSENSOR2_ADDR, POWER_CTL, MEASURE_BIT);

   mx_accelerometer_set_state(I2C_GSENSOR1_ADDR, DATA_RATE_REG, DATA_RATE);
   mx_accelerometer_set_state(I2C_GSENSOR2_ADDR, DATA_RATE_REG, DATA_RATE);

   //full resolution
   mx_accelerometer_set_state(I2C_GSENSOR1_ADDR, DATA_FORMAT_REG,
DATA_FORMAT);
   mx_accelerometer_set_state(I2C_GSENSOR2_ADDR, DATA_FORMAT_REG,
DATA_FORMAT);

   mx_accelerometer_calibrate(I2C_GSENSOR1_ADDR);
   sleep (1);
   mx_accelerometer_calibrate(I2C_GSENSOR2_ADDR);

   if (mx_accelerometer_read(&axis, I2C_GSENSOR1_ADDR) == 0) {
      printf("Disk1: x %f y %f z %f\n", axis.x_axis, axis.y_axis, axis.z_axis);
   }

   if (mx_accelerometer_read(&axis, I2C_GSENSOR2_ADDR) == 0) {
         printf("Disk2: x %f y %f z %f\n", axis.x_axis, axis.y_axis,
axis.z_axis);
   }
   return 0;
}
```

6

System Recovery

The TC-6110 ready-to-run embedded computers are a Windows Embedded Standard 7 platform. This chapter describes the recovery process in the event of system instability.

The following topics are covered in this chapter:

- **Recovery Environment**
- **Recovery Procedure**
  - Saving the System to the USB Drive

Overview: Setting Up the Recovery Environment

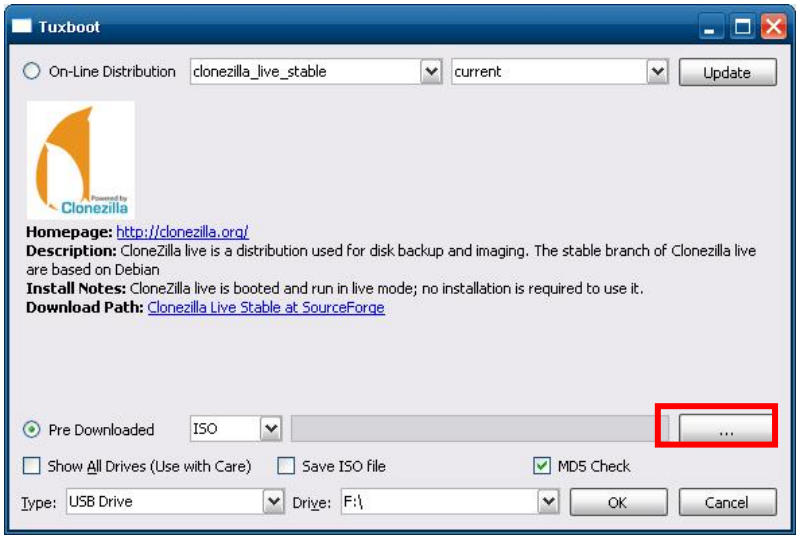
A DA-682A computer, a 4 GB (min.) USB drive, and a copy of the recovery suite are all required to set up the DA-682A's system recovery environment.
The recovery procedure itself requires only a DA-682A computer and a bootable USB drive.

The following steps describe the basic process of setting up the system recovery environment:

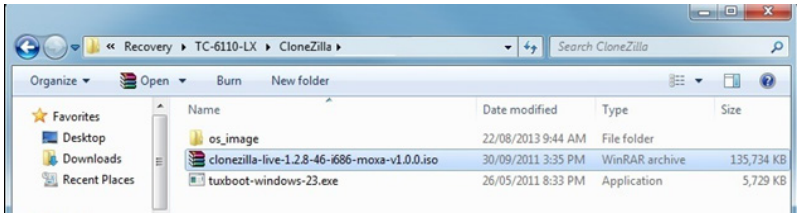
1. First, prepare the USB drive by copying over to it a bootable recovery environment; ths comes in the form of an ISO image, and can be found on your software CD.
2. Here, you may choose to create a bare-bones stock OS recovery image; if you choose this option as your recovery method, keep in mind that any applications or scripts you install later will be lost if a recovery is required.
3. Here, you will reset the BIOS so the USB port is the first boot priority. **If you are initiating a recovery from a key you have already configured, this will be your starting point.** The system will re-booted into the Clonezilla recovery environment found on the USB
4. This step describes how to create an exact copy of a fully configured system on the USB drive. This is the alternative to the stock OS recovery offered in Step 2.
5. This step describes how to perform a recovery; you may use it to run a trial recovery and test your setup.
6. This step explains how and why to return the BIOS to its original state.

Step 1: Prepare the USB Drive

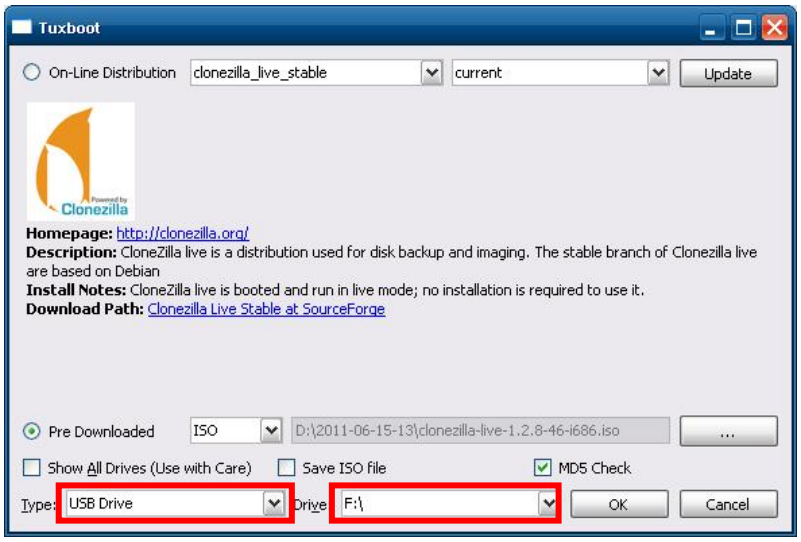
1. From the software DVD that came with your computer **start the Clonezilla imaging program** (within the current OS) by starting `tuxboot-windows-23.exe`, which is found in the `\recovery\DA-682A-LX_Recovery\clonezilla` directory.
2. At the right, select **Pre-Downloaded** and set the dropdown to **ISO**.
3. **Browse the CD to locate the Clonezilla ISO image** by clicking the button with an ellipsis (**...**).



4. Navigate the file manager to **`\Recovery\DA-682A-LX_Recovery\clonezilla`** directory on the software DVD and select the Clonezilla recovery environment's ISO image.



5. Set the **Device Type** (lower left-hand corner) as **USB Drive**, then set the **Drive** dialog to the letter under which the USB is currently mounted.



6. Click **OK**, and the Clonezilla recovery environment (plus bootloader) will be copied to your USB drive.

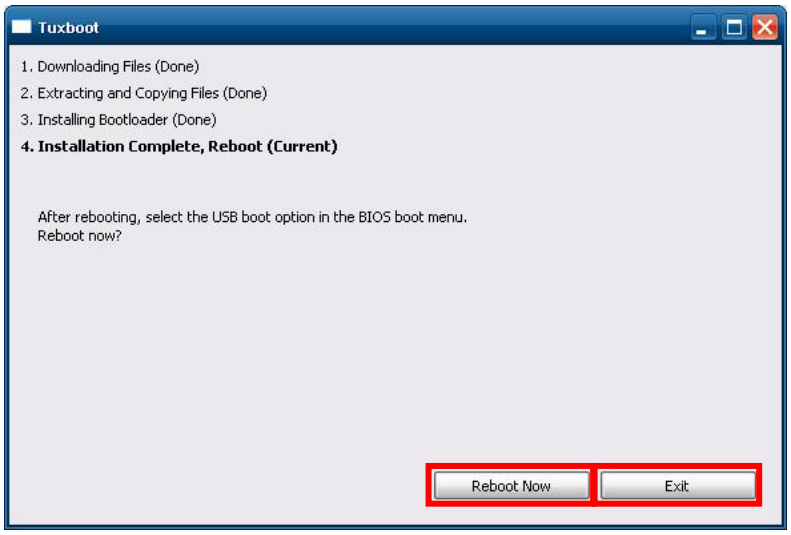
Two Types of Recovery: Base Install, and Fully Configured

Because of the naming conventions used, for any given computer only a single system image may be stored on any given USB drive. Consequently, at this point, users need to make a decision about which sort of system recovery is preferred:

A. A recovery image of **a fully configured OS**, with user-installed software applications and scripts, or
B. A recovery image of only the **basic, newly-installed root OS.**

A: To configure the recovery environment to copy over a fully configured system, users should click **Reboot Now** to close the installation environment and restart the computer. They should then proceed to the next section, **Step 3: Setting the BIOS to Boot via USB** and continue the installation of the recovery environment by continuing to **Step 4 (opt.): Create a Custom System Image**.

B: Users who want to restore the system to a clean OS image with no installed applications, scripting, or alterations of any kind, should complete this portion of the process by clicking **Exit** here and returning to the original OS. At this point, **Step 1** has been completed, and you should proceed to **Step 3: Setting the BIOS to Boot via USB**, and then go directly to **Step 4: Restoring to a Stock OS**.



ATTENTION

You must manually delete the **EFI** directory on the USB.

Step 2 (opt.): Recovering to a Stock OS

The instructions which follow describe how to set up the recovery environment that will ***restore the operating system to a pristine post-install state***. If you have installed any software on your system, then following these directions will result in ***all custom applications and code being wiped from the operating system***. If the computer has already been heavily customized with user applications and local scripts, then skip this section and instead go to the next section, **Step 3: Setting the BIOS to Boot via USB**. There, you will begin the process of copying over a full system image.

Creating a post-install rescue drive involves just two steps: preparing the USB drive, and then copying the rescue image (found on your Moxa software CD) to it.

1. First, if you haven't already, complete step 1 **by preparing the USB drive**.
2. Next, copy the stock OS image (found on the software CD that shipped with your computer) over to the USB drive; the image will be found in the **recovery** directory, `/media/cd0/recovery/`**`os_image`**, and will be copied to the USB file tree at `/media/usb0/home/partimag`. Depending on how the USB and CD have been automounted, you will use a command much like this:

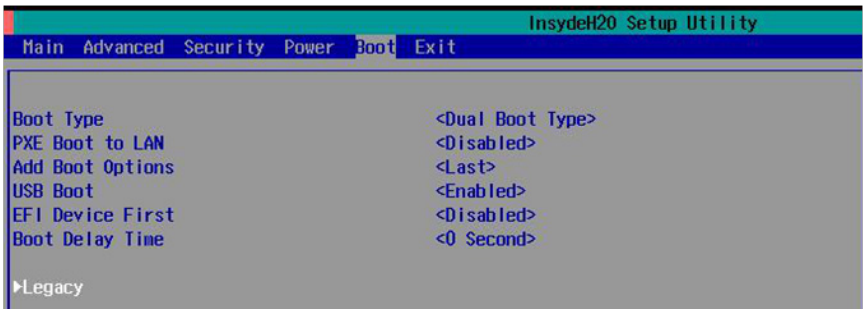
```
moxa@Moxa:~# cp /media/cd0/recovery/os_image  /media/usb0/home/partimag/
```

That's it. You have now configured a USB recovery key that will recover your computer to the stock operating system it shipped with. If you wish, you may now undertake a trial recovery To do this, continue on to **step 3, setting the BIOS to boot over USB**, skip step 4, and then go on to **step 5, performing a system recovery**.

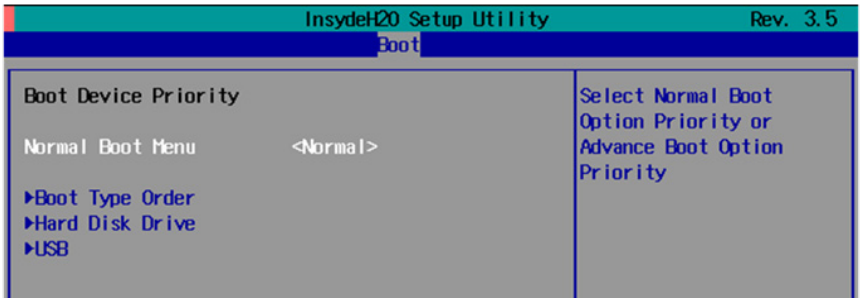
Step 3: Setting the BIOS to Boot via USB

At this stage, users will reset the BIOS so that the system boots directly from the USB. This must be done before the rest of the system recovery environment may be configured.

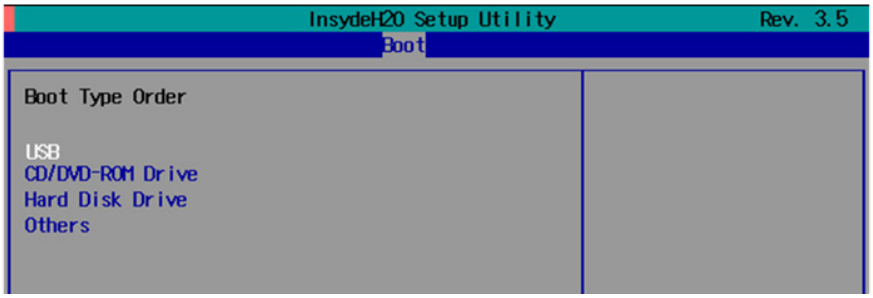
1. Turn on the computer and, during the POST process, press F2 until you hear a long beep. You should then enter the BIOS setup menu.
2. Use the arrow keys to navigate to the **Boot** tab, and then press **Enter**.
3. Use the up/down arrows to highlight **Legacy** in the boot tab's menu, and press **Enter**.



4. Use the up/down arrow keys to navigate to the **Boot Type Order** link, and then press **Enter**.



5. Use the arrows to highlight USB and then use the plus/minus signs (**+ -**) to move it to the first boot priority position. **Warning: Incorrectly configuring the boot priority will lead to recovery failure.**



6. Press F10 and then press **Enter** to save and exit the BIOS configuration interface. This should initiate the next reboot, and your system should now boot from the USB drive.

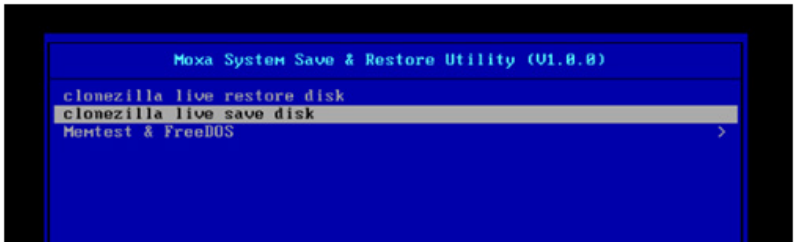
Step 4 (opt.): Create a Custom System Image

The instructions which follow are only to be used if you decided in **Step 1** of this process ***to create a full copy of an already-configured system***. If you have not yet installed any software on your system and are configuring this recovery utility to restore a bare-bones stock OS, then skip this section and instead go to **Step 4: Recovering to a Stock OS** to see how to prepare your USB with a clean OS image.

The procedure below describes a configuration for restoring a complete system that has been customized with user applications and scripts. Here, you will save to the USB drive a copy of the entire system as it is currently configured to be used as a full system recovery image should the system crash. During this process, ***all files on your USB that are mounted under*** *F:\home\partimag* ***will be overwritten***.

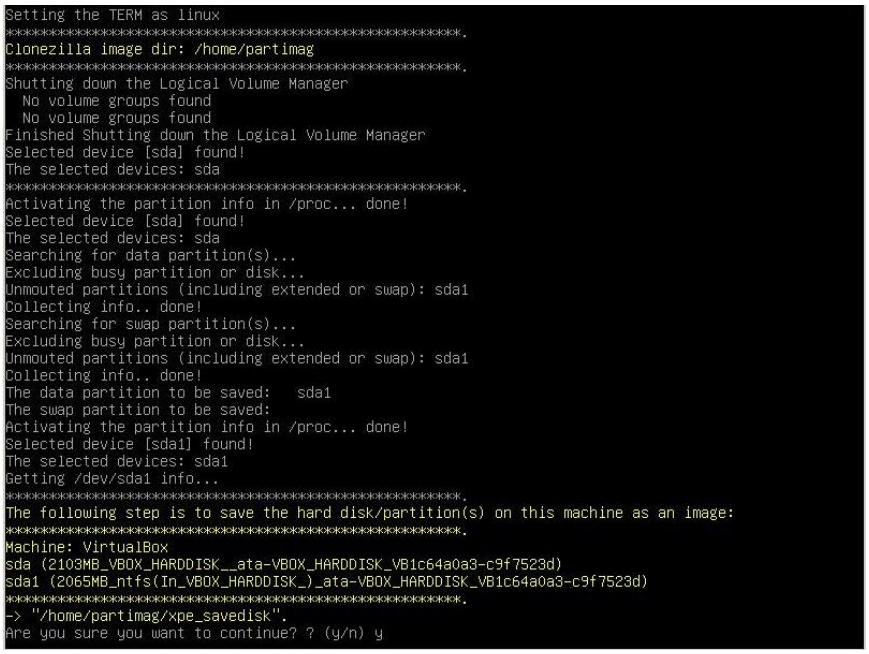
You should have already changed the BIOS settings to set the USB drive as the first boot priority. If you have not yet reset the boot priority, first return to **Step 3: Setting the BIOS to Boot via USB** and follow the directions there.

1. Once the system has launched and the DA-682A has booted the recovery environment from the USB drive, navigate to the entry **Clonezilla Live Save Disk**, and select it by pressing **Enter**. This will take you into the **recovery image creation environment**, allowing you to copy your full system setup to the USB drive.



2. The DA-682A will now boot into the image creation environment. Wait for the boot process to finish.

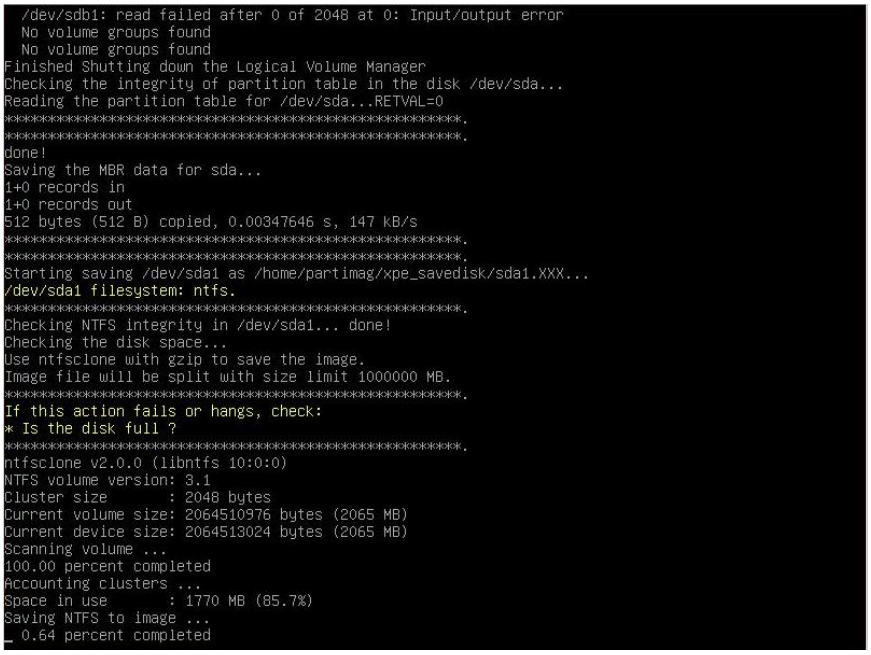
3. Once the image creation environment has completed booting up, you will be given a warning and asked if you wish to continue. Please keep in mind that if you create the recovery image, then ***any residual files currently copied to the*** **`/home/partimag`** ***directory will be deleted***. If there are any files remaining in the USB **partition image** directory and you wish to save them, you must exit the recovery environment and copy these files to another disk. If you wish to continue with the image creation, press **Y** (case insensitive) to continue (screenshot on the next page).



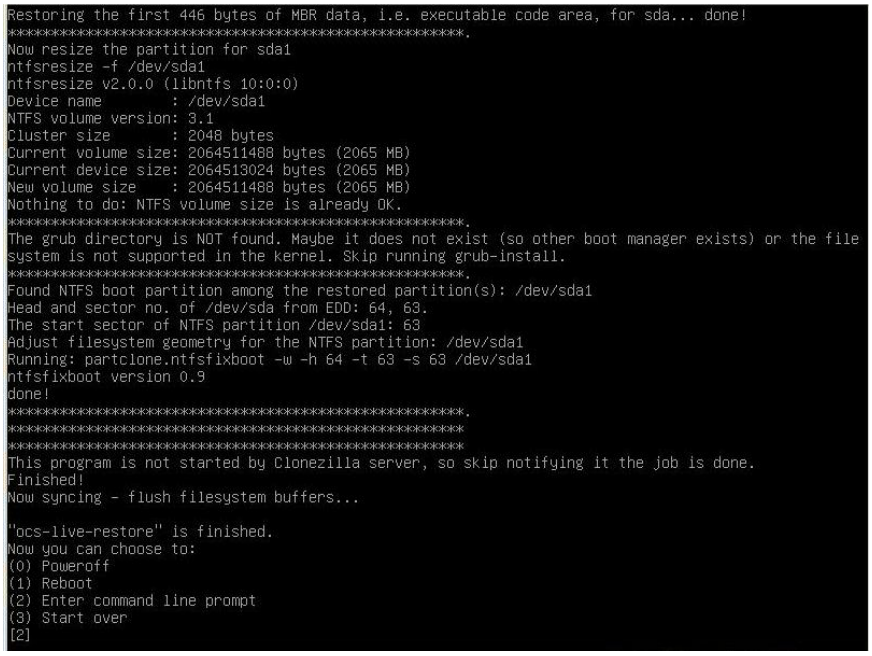
WARNING

The same filename is used for all recovery images, whether for the full system backup or for the clean OS image installation. This means that currently, it is impossible to have more than one system image per USB drive.

4. At this point, the recovery environment will copy of the entire hard drive to your USB drive. This will likely take several minutes, and perhaps as long as half an hour. Do not remove the USB drive during this time; wait patiently for the process to finish. Depending on the speed of your USB drive, this may be a good time to get a cup of coffee, or take a nap.



5. At this point you may choose to power down the computer (press **0**), reboot (press **1**), enter a console terminal (access a console TTY -- press **2**), or re-initiate the entire procedure (press **3**). ***Do not remove the USB drive until you have rebooted or powered down the system.***

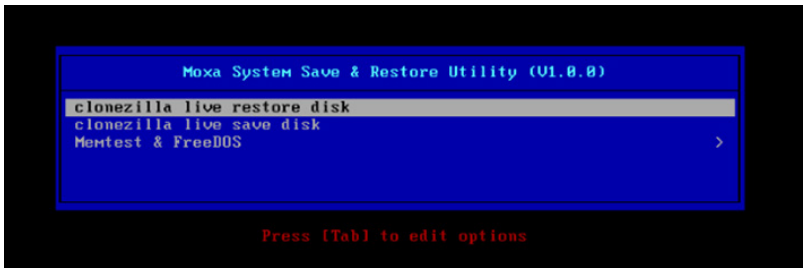


6. Once you have powered down the system and removed the USB drive, you have finished configuring the recovery environment. The USB drive should be clearly labeled and stored in a safe place. You may now continue to **Step 6, where you will reset the BIOS to its original state**, or you may go to and test the recovery procedure for successful configuration (Step 5).

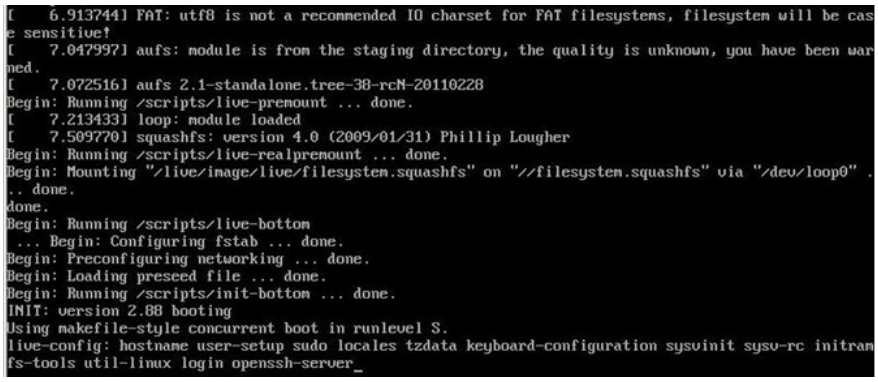
Step 5: Performing a System Recovery

Connect the USB drive to any of the DA-682A's USB ports and then reboot the computer. The system will boot from the USB into the Clonezilla boot loader.

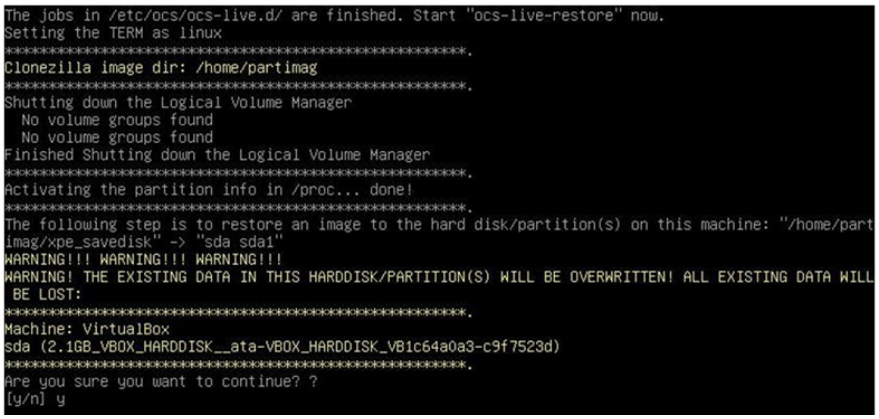
3. Select **Clonezilla Live Restore Disk** to boot into the system restoration environment.



4. Wait for the boot process to finish.

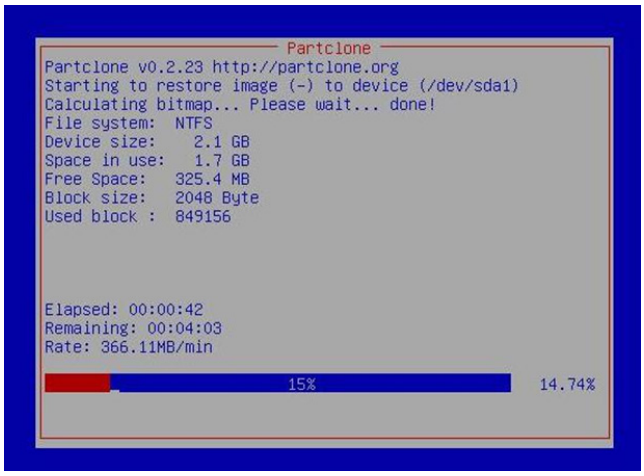


5. At this point, the system will remind you that you are about to overwrite your entire operating system with a new drive image, and ask you if you want to continue. When prompted, enter **Y** (case insensitive) from the keyboard to start the system restoration process. Any other letter or **Ctrl-C** will cancel it and exit Clonezilla.

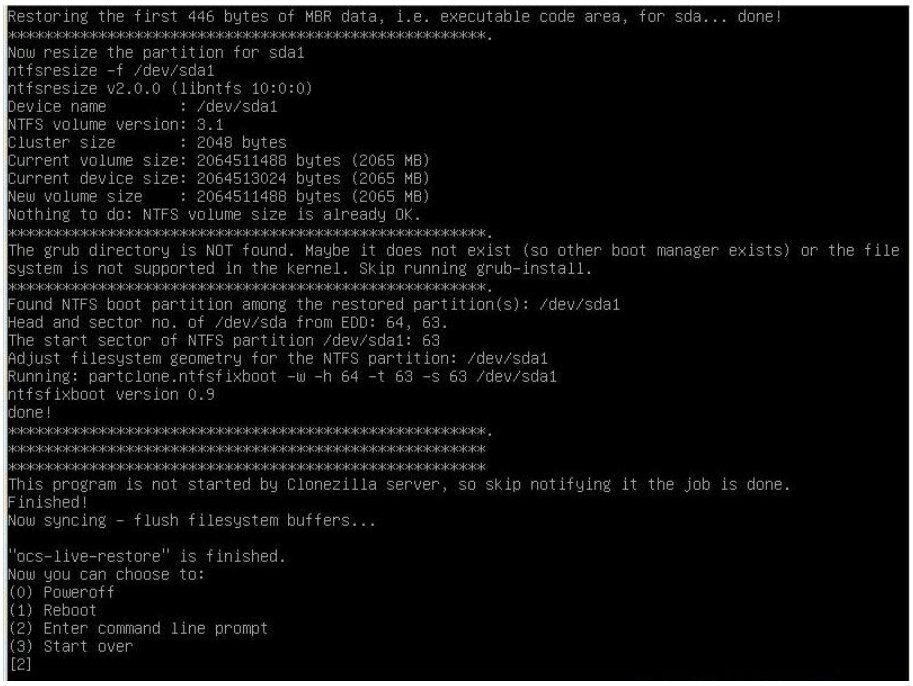


6. The system will give you another warning that you are about to overwrite your hard drive, and erase all data on the partition listed (**sda1**, in the example below). If you wish to continue, enter **Y** (case insensitive).

7. Now, Clonezilla will copy the system image you have configured on to your primary system drive. Your original system (and any stored data or configurations that were made after the recovery disk was created) will be entirely wiped clean. Wait for the process to finish; depending on the system, this should take about 10 minutes.



8. At this point, complete the restoration by selecting (**0**) **Poweroff**. This will shut down the computer; however, if the **Power Switch** remains inserted in the front panel of the computer and is left in the **ON** position, then the system will immediately initiate a soft reboot. To avoid this, users may use the switch to cut power to the computer immediately following the shutdown, or may simply remove the power switch from the front panel and then use the console to shut down the computer by pressing **0**.



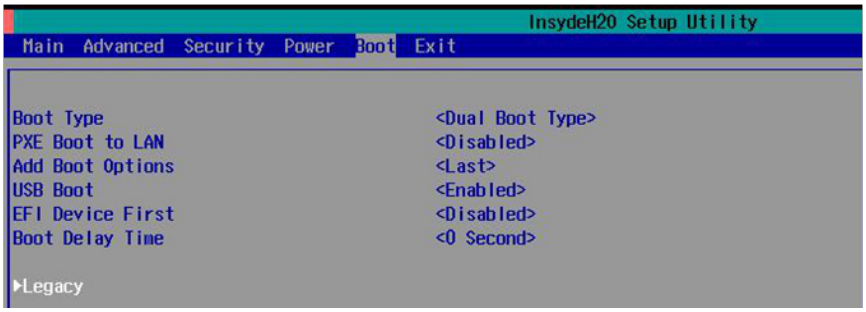
9. After the computer has powered down, remove the USB drive and store it in a safe place.

Step 6: Reset the BIOS to its Original State

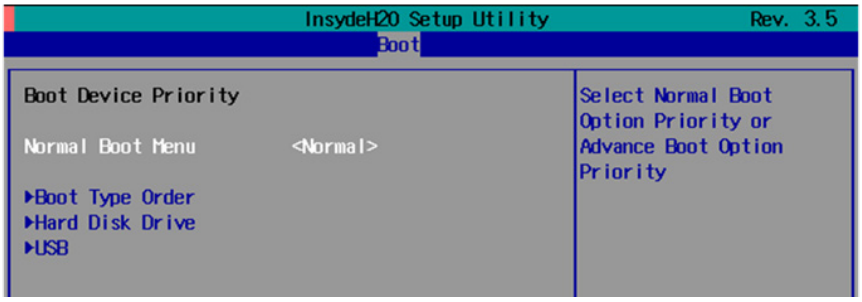
Now you will need to return the boot priority to its original configuration so that the system will boot from the original disk. This is done for two reasons; the first is security, so that the machine may not be rebooted from unauthorized USB drives

The second reason, however, is functional: currently, if the DA-682A is set to boot from the USB drive, then ***the DA-682A will hang any time a USB data drive (i.e.: non-bootable image) is inserted in the machine at boot time***. The DA-682A does not currently have the capacity to distinguish between simple USB data drives and boot-capable OS drives.

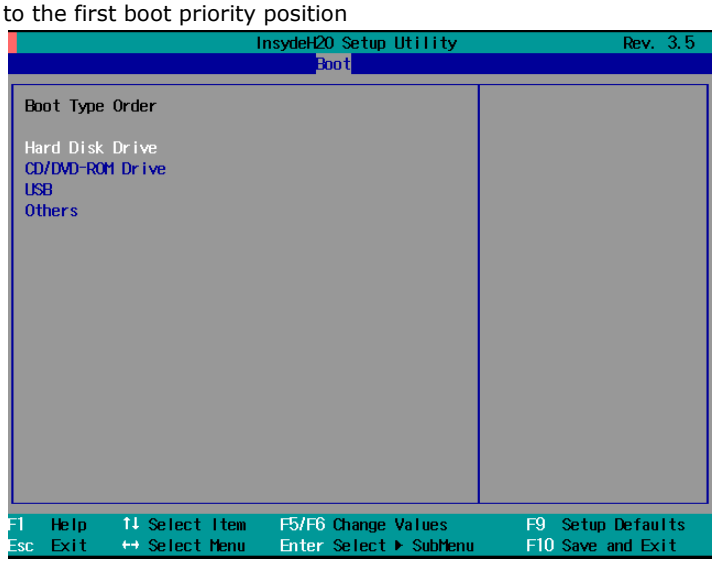
1. Reboot the computer and, during the POST process, press F2 until you hear a long beep. You should then enter the BIOS setup menu.
2. Use the left/right arrow keys to navigate to the **Boot** tab, and then press **Enter**.
3. Use the up/down arrows to highlight **Legacy** in the boot tab's menu, and press **Enter**.



4. Use the up/down arrow keys to navigate to the **Boot Type Order** link, and then press **Enter**.



5. Use the up/down arrows to highlight **Hard Disk Drive** and then use the plus/minus signs (**+ -**) to move it to the first boot priority position



7. Press F10 and then press **Enter** to save and exit the BIOS configuration interface. This should initiate the next reboot, and your system should now boot from the USB drive.

A

Software Components

Package	Version	Description
acpi	1.6-1	Displays information on ACPI devices
acpi-support-base	0.140-5	Scripts for handling base ACPI events such as the power button
acpid	1:2.0.16-1+deb7u1	Advanced Configuration and Power Interface event daemon
adduser	3.113+nmu3	Add and remove users and groups
apache2	2.2.22-13	Apache HTTP Server metapackage
apache2-mpm-prefork	2.2.22-13	Apache HTTP Server - traditional non-threaded model
apache2-utils	2.2.22-13	Utility programs for web server
apache2.2-bin	2.2.22-13	Apache HTTP Server common binary files
apache2.2-common	2.2.22-13	Apache HTTP Server common files
apt	0.9.7.9	Command line package manager
apt-listchanges	2.85.11	Package change history notification tool
apt-utils	0.9.7.9	Package management related utility programs
aptitude	0.6.8.2-1	Terminal-based package manager
aptitude-common	0.6.8.2-1	Architecture independent files for the aptitude package manager
at	3.1.13-2	Delayed job execution and batch processing
base-files	7.1wheezy1	Debian base system miscellaneous files
base-passwd	3.5.26	Debian base system master password and group files
bash	4.2+dfsg-0.1	GNU Bourne Again SHell
bash-completion	1:2.0-1	Programmable completion for the bash shell
bc	1.06.95-2+b1	The GNU bc arbitrary precision calculator language
bind9-host	1:9.8.4.dfsg.P1-6+nmu2	Version of 'host' bundled with BIND 9.X
binutils	2.22-8	GNU assembler, linker and binary utilities
bridge-utils	1.5-6	Utilities for configuring the Linux Ethernet bridge
bsdmainutils	9.0.3	Collection of more utilities from FreeBSD
bsdutils	1:2.20.1-5.3	Basic utilities from 4.4BSD-Lite
build-essential	11.5	Informational list of build-essential packages
busybox	1:1.20.0-7	Tiny utilities for sm and embedded systems
bzip2	1.0.6-4	High-quality block-sorting file compressor - utilities
ca-certificates	20130119	Common CA certificates
console-setup	1.88	Console font and keymap setup program
console-setup-linux	1.88	Linux specific part of console-setup
coreutils	8.13-3.5	GNU core utilities
cpio	2.11+dfsg-0.1	GNU cpio -- a program to manage archives of files
cpp	4:4.7.2-1	GNU C preprocessor (cpp)

cpp-4.7	4.7.2-5	GNU C preprocessor
cron	3.0pl1-124	Process scheduling daemon
dash	0.5.7-3	POSIX-compliant shell
db5.1-util	5.1.29-5	Berkeley v5.1 Database Utilities
dc	1.06.95-2+b1	The GNU dc arbitrary precision reverse-polish calculator
debconf	1.5.49	Debian configuration management system
debconf-i18n	1.5.49	Full internationalization support for debconf
debian-archive-keyring	2012.4	GNU PG archive keys of the Debian archive
debian-faq	5.0.1	Debian FAQ
debianutils	4.3.2	Miscellaneous utilities specific to Debian
dialog	1.1-20120215-2	Displays user-friendly dialog boxes from shell scripts
dictionaries-common	1.12.11	Common utilities for spelling dictionary tools
diffutils	1:3.2-6	File comparison utilities
discover	2.1.2-5.2	Hardware identification system
discover-data	2.2010.10.18	Data lists for Discover hardware detection system
dmidecode	2.11-9	SMBIOS/DMI table decoder
dmsetup	2:1.02.74-7	Linux Kernel Device Mapper userspace library
dnsutils	1:9.8.4.dfsg.P1-6+nmu 2	Clients provided with BIND
dpkg	1.16.10	Debian package management system
dpkg-dev	1.16.10	Debian package development tools
e2fslibs: i386	1.42.5-1.1	ext2/ext3/ext4 file system libraries
e2fsprogs	1.42.5-1.1	ext2/ext3/ext4 file system utilities
ethtool	1:3.4.2-1	Display or change Ethernet device settings
fakeroot	1.18.4-2	Tool for simulating superuser privileges
file	5.11-2	Determines file type using "magic" numbers
findutils	4.4.2-4	Utilities for finding files--find, xargs
firmware-realtek	0.36+wheezy.1	Binary firmware for Realtek wired and wireless network adapters
ftp	0.17-27	Classical file transfer client
g++	4:4.7.2-1	GNU C++ compiler
g++-4.7	4.7.2-5	GNU C++ compiler
gcc	4:4.7.2-1	GNU C compiler
gcc-4.7	4.7.2-5	GNU C compiler
gcc-4.7-base:	4.7.2-5	GCC, the GNU Compiler Collection (base package)
geoip-database	20130213-1	IP lookup command line tools that use the GeoIP library (country database)
gettext-base	0.18.1.1-9	GNU Internationalization utilities for the base system
gnupg	1.4.12-7	GNU privacy guard - a free PGP replacement
gpgv	1.4.12-7	GNU privacy guard - signature verification tool
grep	2.12-2	GNU grep, egrep and fgrep
groff-base	1.21-9	GNU troff text-formatting system (base system components)
grub-common	1.99-27+deb7u1	GRand Unified Bootloader (common files)
grub-pc	1.99-27+deb7u1	GRand Unified Bootloader, version 2 (PC/BIOS version)
grub-pc-bin	1.99-27+deb7u1	GRand Unified Bootloader, version 2 (PC/BIOS binaries)
grub2-common	1.99-27+deb7u1	GRand Unified Bootloader (common files for version 2)

gzip	1.5-1.1	GNU compression utilities
hdparm	9.39-1+b1	Tune hard disk parameters for high performance
host	1:9.8.4.dfsg.P1-6+nmu2	Transitional package
hostname	3.11	Utility to set/show the host name or domain name
iamerican	3.3.02-6	American English dictionary for ispell (standard version)
ibritish	3.3.02-6	British English dictionary for ispell (standard version)
ienglish-common	3.3.02-6	Common files for British and American ispell dictionaries
ifrename	30~pre9-8	Rename network interfaces based on various static criteria
ifupdown	0.7.8	High level tools to configure network interfaces
initramfs-tools	0.109.1	Generic modular initramfs generator
initscripts	2.88dsf-41	Scripts for initializing and shutting down the system
insserv	1.14.0-5	Boot sequence organizer using LSB init.d script dependency information
inst-info	4.13a.dfsg.1-10	Manage insted documentation in info format
iproute	20120521-3+b3	Networking and traffic control tools
iptables	1.4.14-3.1	Administration tools for packet filtering and NAT
iputils-ping	3:20101006-1+b1	Tools to test the reachability of network hosts
isc-dhcp-client	4.2.2.dfsg.1-5+deb70u6	ISC DHCP client
isc-dhcp-common	4.2.2.dfsg.1-5+deb70u6	Common files used by the isc-dhcp* packages
iso-codes	3.41-1	ISO language, territory, currency, script codes and their translations
ispell	3.3.02-6	International Ispell (an interactive spelling corrector)
kbd	1.15.3-9	Linux console font and keytable utilities
keyboard-configuration	1.88	System-wide keyboard preferences
klibc-utils	2.0.1-3.1	Small utilities built with klibc for early boot
kmod	9-3	Tools for managing Linux kernel modules
krb5-locales	1.10.1+dfsg-5+deb7u1	Internationalization support for MIT Kerberos
laptop-detect	0.13.7	Attempt to detect a laptop
less	444-4	Pager program similar to more
libacl1:i386	2.2.51-8	Access control list shared library
libalgorithm-diff-perl	1.19.02-2	Module to find differences between files
libalgorithm-diff-xs-perl	0.04-2+b1	Module to find differences between files (XS accelerated)
libalgorithm-merge-perl	0.08-2	Perl module for three-way merge of textual data
libapache2-mod-php5	5.4.4-14+deb7u3	Server-side, HTML-embedded scripting language (Apache 2 module)
libapr1	1.4.6-3	Apache Portable Runtime Library
libaprutil1	1.4.1-3	Apache Portable Runtime Utility Library
libaprutil1-dbd-sqlite3	1.4.1-3	Apache Portable Runtime Utility Library - SQLite3 Driver
libaprutil1-ldap	1.4.1-3	Apache Portable Runtime Utility Library - LDAP Driver
libapt-inst1.5:i386	0.9.7.9	deb package format runtime library
libapt-pkg4.12: i386	0.9.7.9	Package management runtime library

libasprintf0c2: i386	0.18.1.1-9	GNU library to use fprintf and friends in C++
libattr1: i386	1:2.4.46-8	Extended attribute shared library
libbind9-80	1:9.8.4.dfsg.P1-6+nmu2	BIND9 Shared Library used by BIND
libblkid1: i386	2.20.1-5.3	Block device id library
libboost-iostreams1.49.0	1.49.0-3.2	Boost.Iostreams Library
libbsd0: i386	0.4.2-1	Utility functions from BSD systems - shared library
libbz2-1.0: i386	1.0.6-4	High-quality block-sorting file compressor library - runtime
libc-bin	2.13-38	Embedded GNU C Library: Binaries
libc-dev-bin	2.13-38	Embedded GNU C Library: Development binaries
libc6: i386	2.13-38	Embedded GNU C Library: Shared libraries
libc6-dev: i386	2.13-38	Embedded GNU C Library: Development Libraries and Header Files
libcap2: i386	1:2.22-1.2	Support for getting/setting POSIX.1e capabilities
libclass-isa-perl	0.36-3	Report the search path for a class's ISA tree
libcomerr2: i386	1.42.5-1.1	Common error description library
libcwidget3	0.5.16-3.4	High-level terminal interface library for C++ (runtime files)
libdb5.1: i386	5.1.29-5	Berkeley v5.1 Database Libraries [runtime]
libdevmapper1.02.1: i386	2:1.02.74-7	Linux Kernel Device Mapper userspace library
libdiscover2	2.1.2-5.2	Hardware identification library
libdns88	1:9.8.4.dfsg.P1-6+nmu2	DNS Shared Library used by BIND
libdpkg-perl	1.16.10	Dpkg perl modules
libedit2: i386	2.11-20080614-5	BSD editline and history libraries
libept1.4.12	1.0.9	High-level library for managing Debian package information
libevent-2.0-5: i386	2.0.19-stable-3	Asynchronous event notification library
libexpat1: i386	2.1.0-1	XML parsing C library - runtime library
libfile-copy-recursive-perl	0.38-1	Perl extension for recursively copying files and directories
libfile-fcntllock-perl	0.14-2	Perl module for file locking with fcntl(2)
libfreetype6: i386	2.4.9-1.1	FreeType 2 font engine, shared library files
libfuse2: i386	2.9.0-2+deb7u1	Filesystem in Userspace (library)
libgc1c2	1:7.1-9.1	Conservative garbage collector for C and C++
libgcc1: i386	1:4.7.2-5	GCC support library
libgcrypt11: i386	1.5.0-5	LGPL Crypto library - runtime library
libgdbm3: i386	1.8.3-11	GNU dbm database routines (runtime version)
libgeoip1	1.4.8+dfsg-3	Non-DNS IP-to-country resolver library
libgmp10: i386	2:5.0.5+dfsg-2	Multiprecision arithmetic library
libgnutls26: i386	2.12.20-7	GNU TLS library - runtime library
libgomp1: i386	4.7.2-5	GCC OpenMP (GOMP) support library
libgpg-error0: i386	1.10-3.1	Library for common error values and messages in GnuPG components
libgpgme11	1.2.0-1.4	GPGME - GnuPG Made Easy
libgpm2: i386	1.20.4-6	General Purpose Mouse - shared library
libgssapi-krb5-2: i386	1.10.1+dfsg-5+deb7u1	MIT Kerberos runtime libraries - krb5 GSS-API Mechanism
libgssglue1: i386	0.4-2	Mechanism-switch gssapi library
libidn11: i386	1.25-2	GNU Libidn library, implementation of IETF IDN specifications

libisc84	1:9.8.4.dfsg.P1-6+nmu2	ISC Shared Library used by BIND
libisccc80	1:9.8.4.dfsg.P1-6+nmu2	Command Channel Library used by BIND
libisccfg82	1:9.8.4.dfsg.P1-6+nmu2	Configures File Handling Library used by BIND
libitm1: i386	4.7.2-5	GNU Transactional Memory Library
libiw30: i386	30~pre9-8	Wireless tools - library
libk5crypto3: i386	1.10.1+dfsg-5+deb7u1	MIT Kerberos runtime libraries - Crypto Library
libkeyutils1: i386	1.5.5-3	Linux Key Management Utilities (library)
Libklibc	2.0.1-3.1	Minimal libc subset for use with initramfs
libkmod2: i386	9-3	libkmod shared library
libkrb5-3: i386	1.10.1+dfsg-5+deb7u1	MIT Kerberos runtime libraries
libkrb5support0: i386	1.10.1+dfsg-5+deb7u1	MIT Kerberos runtime libraries - Support library
libldap-2.4-2: i386	2.4.31-1+nmu2	OpenLDAP libraries
liblocale-gettext-perl	1.05-7+b1	Module using libc functions for internationalization in Perl
liblockfile-bin	1.09-5	Support binaries for and cli utilities based on liblockfile
liblockfile1: i386	1.09-5	NFS-safe locking library
liblwres80	1:9.8.4.dfsg.P1-6+nmu2	Lightweight Resolver Library used by BIND
liblzma5: i386	5.1.1alpha+20120614-2	XZ-format compression library
liblzo2-2: i386	2.06-1	Data compression library
libmagic1: i386	5.11-2	File type determination library using "magic" numbers
libmount1	2.20.1-5.3	Block device id library
libmpc2: i386	0.9-4	Multiple precision complex floating-point library
libmpfr4: i386	3.1.0-5	Multiple precision floating-point computation
libncurses5: i386	5.9-10	Shared libraries for terminal handling
libncurses5-dev	5.9-10	Developer's libraries for ncurses
libncursesw5: i386	5.9-10	Shared libraries for terminal handling (wide character support)
libnet-telnet-perl	3.03-3	Script telnetable connections
libnewt0.52	0.52.14-11.1	Not Erik's Windowing Toolkit - text mode windowing with slang
libnfnetlink0	1.0.0-1.1	Netfilter netlink library
libnfsidmap2: i386	0.25-4	NFS idmapping library
libonig2	5.9.1-1	Oniguruma regular expressions library
libp11-kit0: i386	0.12-3	Library for loading and coordinating access to PKCS#11 modules - runtime
libpam-modules: i386	1.1.3-7.1	Pluggable Authentication Modules for PAM
libpam-modules-bin	1.1.3-7.1	Pluggable Authentication Modules for PAM - helper binaries
libpam-runtime	1.1.3-7.1	Runtime support for the PAM library
libpam0g: i386	1.1.3-7.1	Pluggable Authentication Modules library
libpcap0.8: i386	1.3.0-1	System interface for user-level packet capture
libpci3: i386	1:3.1.9-6	Linux PCI Utilities (shared library)
libpcre3: i386	1:8.30-5	Perl 5 Compatible Regular Expression Library - runtime files
libperl-dev	5.14.2-21	Perl library: development files
libperl5.14	5.14.2-21	Shared Perl library

libpipeline1: i386	1.2.1-1	Pipeline manipulation library
libpkcs11-helper1: i386	1.09-1	Library that simplifies the interaction with PKCS#11
libpopt0: i386	1.16-7	Libary for parsing cmdline parameters
libprocps0: i386	1:3.3.3-3	Library for accessing process information from /proc
libpth20	2.0.7-16	The GNU Portable Threads
libpython2.7	2.7.3-6	Shared Python runtime library (version 2.7)
libqdbm14	1.8.78-2	QDBM Database Libraries without GDBM wrapper[runtime]
libquadmath0: i386	4.7.2-5	GCC Quad-Precision Math Library
libreadline6: i386	6.2+dfsg-0.1	GNU readline and history libraries, run-time libraries
libsasl2-2: i386	2.1.25.dfsg1-6+deb7u1	Cyrus SASL - authentication abstraction library
libsasl2-modules: i386	2.1.25.dfsg1-6+deb7u1	Cyrus SASL - pluggable authentication modules
libselinux1: i386	2.1.9-5	SELinux runtime shared libraries
libsemanage-common	2.1.6-6	Common files for SELinux policy management libraries
libsemanage1: i386	2.1.6-6	SELinux policy management library
libsensors4: i386	1:3.3.2-2	Library to read temperature/voltage/fan sensors
libsepol1: i386	2.1.4-3	SELinux library for manipulating binary security policies
libsigc++-2.0-0c2a: i386	2.2.10-0.2	Type-safe Signal Framework for C++ - runtime
libslang2: i386	2.2.4-15	S-Lang programming library - runtime version
libsqlite3-0: i386	3.7.13-1+deb7u1	SQLite 3 shared library
libsqlite3-dev	3.7.13-1+deb7u1	SQLite 3 development files
libss2: i386	1.42.5-1.1	Command-line interface parsing library
libssl1.0.0: i386	1.0.1e-2	SSL shared libraries
libstdc++6: i386	4.7.2-5	GNU Standard C++ Library v3
libstdc++6-4.7-dev	4.7.2-5	GNU Standard C++ Library v3 (development files)
libswitch-perl	2.16-2	Switch statement for Perl
libtasn1-3: i386	2.13-2	Manage ASN.1 structures (runtime)
libtext-charwidth-perl	0.04-7+b1	Get display widths of characters on the terminal
libtext-iconv-perl	1.7-5	Converts between character sets in Perl
libtext-wrapi18n-perl	0.06-7	Internationalized substitute of Text::Wrap
libtimedate-perl	1.2000-1	Collection of modules to manipulate date/time information
libtinfo-dev: i386	5.9-10	Developer's library for the low-level terminfo library
libtinfo5: i386	5.9-10	Shared low-level terminfo library for terminal handling
libtirpc1: i386	0.2.2-5	Transport-independent RPC library
libtokyocabinet9: i386	1.4.47-2	Tokyo Cabinet Database Libraries [runtime]
libudev0: i386	175-7.2	libudev shared library
libusb-0.1-4: i386	2:0.1.12-20+nmu1	Userspace USB programming library
libusb-1.0-0: i386	2:1.0.11-1	Userspace USB programming library
libustr-1.0-1: i386	1.0.4-3	Micro string library: shared library
libuuid-perl	0.02-5	Perl extension for using UUID interfaces as defined in e2fsprogs
libuuid1: i386	2.20.1-5.3	Universy Unique ID library
libwrap0: i386	7.6.q-24	Wietse Venema's TCP wrappers library
libx11-6:	2:1.5.0-1+deb7u1	X11 client-side library
libx11-data	2:1.5.0-1+deb7u1	X11 client-side library

libx86-1: i386	1.1+ds1-10	x86 real-mode library
libxapian22	1.2.12-2	Search engine library
libxau6: i386	1:1.0.7-1	X11 authorization library
libxcb1: i386	1.8.1-2+deb7u1	X C Binding
libxdmcp6: i386	1:1.1.1-1	X11 Display Manager Control Protocol library
libxext6: i386	2:1.3.1-2+deb7u1	X11 miscellaneous extension library
libxml2: i386	2.8.0+dfsg1-7+nmu1	GNOME XML library
libxmuu1: i386	2:1.1.1-1	X11 miscellaneous micro-utility library
linux-base	3.5	Linux image base package
linux-image-3.2.0-4-686-pae	3.2.46-1	Linux 3.2 for 64-bit PCs
linux-image-686-page	3.2+46	Linux for modern PCs (meta-package)
linux-libc-dev: i386	3.2.46-1	Linux support headers for userspace development
locales	2.13-38	Embedded GNU C Library: National Language (locale) data [support]
lockfile-progs	0.1.17	Programs for locking and unlocking files and mailboxes
login	1:4.1.5.1-1	System login tools
logrotate	3.8.1-4	Log rotation utility
lsb-base	4.1+Debian8+deb7u1	Linux Standard Base 4.1 init script functionality
lsb-release	4.1+Debian8+deb7u1	Linux Standard Base version reporting utility
lsof	4.86+dfsg-1	Utility to list open files
m4	1.4.16-3	A macro processing language
make	3.81-8.2	A utility for Directing compilation.
man-db	2.6.2-1	On-line manual pager
manpages	3.44-1	Manual pages about using a GNU/Linux system
manpages-dev	3.44-1	Manual pages about using GNU/Linux for development
mawk	1.3.3-17	A pattern scanning and text processing language
mime-support	3.52-1	MIME files 'mime.types' & 'mailcap', and support programs
mlocate	0.23.1-1	Quickly find files on the filesystem based on their name
module-init-tools	9-3	Transitional dummy package (module-init-tools to kmod)
mount	2.20.1-5.3	Tools for mounting and manipulating filesystems
moxa-safeguard-tc-6110	1.0	Moxa SafeGuard monitors vibration and temperature with independent sensors, giving the system operators important data for on-site operation conditions and problem causes for data corruption. In addition, users can also use the G-sensor as a dual-purpose site monitoring tool.
multiarch-support	2.13-38	Transitional package to ensure multiarch compatibility
mutt	1.5.21-6.2	Text-based mailreader supporting MIME, GPG, PGP and threading
ncurses-base	5.9-10	Basic terminal type definitions
ncurses-bin	5.9-10	Terminal-related programs and man pages
ncurses-term	5.9-10	Additional terminal type definitions
net-tools	1.60-24.2	The NET-3 networking toolkit
netbase	5.0	Basic TCP/IP networking system
netcat-traditional	1.10-40	TCP/IP swiss army knife
netsnmp	1.0	The Simple Network Management Protocol (SNMP)

		provides a framework for the exchange of management information between agents (servers) and clients.
nfs-common	1:1.2.6-4	NFS support files common to client and server
ntpdate	1:4.2.6.p5+dfsg-2	Client for setting system time from NTP servers
openbsd-inetd	0.20091229-2	OpenBSD Internet Superserver
openssh-blacklist	0.4.1+nmu1	List of default blacklisted OpenSSH RSA and DSA keys
openssh-blacklist-extra	0.4.1+nmu1	List of non-default blacklisted OpenSSH RSA and DSA keys
openssh-client	1:6.0p1-4	Secure shell (SSH) client, for secure access to remote machines
openssh-server	1:6.0p1-4	Secure shell (SSH) server, for secure access from remote machines
openssl	1.0.1e-2	Secure Socket Layer (SSL) binary and related cryptographic tools
openvpn	2.2.1-8+deb7u2	Virtual private network daemon
os-prober	1.58	Utility to detect other OSes on a set of drives
passwd	1:4.1.5.1-1	Change and administer password and group data
patch	2.6.1-3	Apply a diff file to an original
pciutils	1:3.1.9-6	Linux PCI Utilities
perl	5.14.2-21	Larry W's Practical Extraction and Report Language
perl-base	5.14.2-21	Minimal Perl system
perl-modules	5.14.2-21	Core Perl modules
php5	5.4.4-14+deb7u3	Server-side, HTML-embedded scripting language (metapackage)
php5-cli	5.4.4-14+deb7u3	Command-line interpreter for the php5 scripting language
php5-common	5.4.4-14+deb7u3	Common files for packages built from the php5 source
pm-utils	1.4.1-9	Utilities and scripts for power management
pmount	0.9.23-2	Mount removable devices as normal user
powermgmt-base	1.31	Common utils and configs for power management
ppp	2.4.5-5.1+b1	Point-to-Point Protocol (PPP) - daemon
pppconfig	2.3.18+nmu4	A text menu based utility for configuring ppp
pppoe	3.8-3	PPP over Ethernet driver
pppoeconf	1.20	Configures PPPoE/ADSL connections
procps	1:3.3.3-3	/proc file system utilities
proftpd-basic	1.3.4a-4+nmu1	Versatile, virtual-hosting FTP daemon - binaries
proftpd-mod-vroot	0.9.2-2+b2	ProFTPD module mod_vroot
psmisc	22.19-1+deb7u1	Utilities that use the proc file system
python	2.7.3-4	Interactive high-level object-oriented language (default version)
python-apt	0.8.8.2	Python interface to libapt-pkg
python-apt-common	0.8.8.2	Python interface to libapt-pkg (locales)
python-chardet	2.0.1-2	Universal character encoding detector
python-debian	0.1.21	Python modules to work with Debian-related data formats
python-debianbts	1.11	Python interface to Debian's Bug Tracking System
python-fpconst	0.7.2-5	Utilities for handling IEEE 754 floating point special values
python-minimal	2.7.3-4	Minimal subset of the Python language (default version)

python-reportbug	6.4.4	Python modules for interacting with bug tracking systems
python-soappy	0.12.0-4	SOAP Support for Python
python-support	1.0.15	Automated rebuilding support for Python modules
python2.7	2.7.3-6	Interactive high-level object-oriented language (version 2.7)
python2.7-minimal	2.7.3-6	Minimal subset of the Python language (version 2.7)
readline-common	6.2+dfsg-0.1	GNU readline and history libraries, common files
rpcbind	0.2.0-8	Converts RPC program numbers into universal addresses
rsyslog	5.8.11-3	Reliable system and kernel logging daemon
sed	4.2.1-10	The GNU sed stream editor
sensible-utils	0.0.7	Utilities for sensible alternative selection
sgml-base	1.26+nmu4	SGML infrastructure and SGML catalog file support
sqlite3	3.7.13-1+deb7u1	Command line interface for SQLite 3
ssh	1:6.0p1-4	Secure shell client and server (metapackage)
ssl-cert	1.0.32	Simple debconf wrapper for OpenSSL
strace	4.5.20-2.3	A system c tracer
sudo	1.8.5p2-1+nmu1	Provide limited super user privileges to specific users
sysstat	10.0.5-1	System performance tools for Linux
sysv-rc	2.88dsf-41	System-V-like runlevel change mechanism
sysvinit	2.88dsf-41	System-V-like init utilities
sysvinit-utils	2.88dsf-41	System-V-like utilities
tar	1.26+dfsg-0.1	GNU version of the tar archiving utility
task-english	3.14.1	General English environment
task-ssh-server	3.14.1	SSH server
tasksel	3.14.1	Tool for selecting tasks for instation on Debian systems
tasksel-data	3.14.1	Official tasks used for instation of Debian systems
tcpd	7.6.q-24	Wietse Venema's TCP wrapper utilities
tcpdump	4.3.0-1	Command-line network traffic analyzer
telnet	0.17-36	The telnet client
telnetd	0.17-36	The telnet server
tftpd	0.17-18	Trivial file transfer protocol server
time	1.7-24	GNU time program for measuring CPU resource usage
traceroute	1:2.0.18-3	Traces the route taken by packets over an IPv4/IPv6 network
tzdata	2013c-0wheezy1	Time zone and daylight-saving time data
ucf	3.0025+nmu3	Update Configuration File: preserve user changes to config files.
udev	175-7.2	/dev/ and hotplug management daemon
update-inetd	4.43	inetd configuration file updater
usbmount	0.0.22	Automatic mount and unmount USB mass storage devices
usbutils	1:005-3	Linux USB utilities
util-linux	2.20.1-5.3	Miscellaneous system utilities
util-linux-locales	2.20.1-5.3	Locales files for util-linux
vbetool	1.1-2	Run real-mode video BIOS code to alter hardware state
vim	2:7.3.547-7	Vi IMproved - enhanced vi editor

vim-common	2:7.3.547-7	Vi IMproved - Common files
vim-runtime	2:7.3.547-7	Vi IMproved - Runtime files
vim-tiny	2:7.3.547-7	Vi IMproved - enhanced vi editor - compact version
w3m	0.5.3-8	WWW browsable pager with excellent tables/frames support
wamerican	7.1-1	American English dictionary words for /usr/share/dict
watchdog	5.12-1	System health checker and software/hardware watchdog handler
wget	1.13.4-3	Retrieves files from the web
whiptail	0.52.14-11.1	Displays user-friendly dialog boxes from shell scripts
whois	5.0.23	Intelligent WHOIS client
xauth	1:1.0.7-1	X authentication utility
xkb-data	2.5.1-3	X Keyboard Extension (XKB) configuration data
xml-core	0.13+nmu2	XML infrastructure and XML catalog file support
xz-utils	5.1.1alpha+20120614-2	XZ-format compression utilities
zlib1g: i386	1:1.2.7.dfsg-13	Compression library - runtime

B

Moxa MIB File for TC-6110-LX

```
--
-- MOXA-SYS-MIB.txt
--

    MOXA-SYS-MIB DEFINITIONS ::= BEGIN

        IMPORTS
            enterprises, IpAddress, Integer32, OBJECT-TYPE, MODULE-IDENTITY,
            NOTIFICATION-TYPE
                FROM SNMPv2-SMI
            DisplayString
                FROM SNMPv2-TC;

        -- 1.3.6.1.4.1.8691.17.1
        moxaSystem MODULE-IDENTITY
            LAST-UPDATED "201301031111Z"       -- January 03, 2013 at 11:11 GMT
            ORGANIZATION
                "Moxa Techonology , Software Research Department"
            CONTACT-INFO
                "This mib is being maintained by the Moxa System Software R&D who handle
product line.

                postal:   Taiwan,Taipei,ShienTien
                        P.O. Box 222
                        Phone:(02)8919-1230

                email:    technical_support@moxa.com"
            DESCRIPTION
                "MIB script for all serial product of Embedded Communication &
Computing .Dep."
            REVISION "201301031111Z"       -- January 03, 2013 at 11:11 GMT
            DESCRIPTION
                "Added LED, Power policy, Accelerometer Objects
                Modify Sensor value SYNTAX ( Gauge32 -> Integer32)"
            REVISION "201212221327Z"       -- December 22, 2012 at 13:27 GMT
            DESCRIPTION
                "Added SystemInfoMgmt"
            REVISION "201203211854Z"       -- March 21, 2012 at 18:54 GMT
            DESCRIPTION
                "This file defines the private Moxa product MIB."
            ::= { embeddedComputer 1 }
```

```
--
-- Type definitions
--

        Minutes ::= Integer32
        KBytes ::= INTEGER
        Second ::= Integer32

--
-- Node definitions
--

        -- 1.3.6.1.4.1.8691
        moxa OBJECT IDENTIFIER ::= { enterprises 8691 }

        -- 1.3.6.1.4.1.8691.17
        embeddedComputer OBJECT IDENTIFIER ::= { moxa 17 }

        -- 1.3.6.1.4.1.8691.17.1.1
        productInfoMgmt OBJECT IDENTIFIER ::= { moxaSystem 1 }

        -- 1.3.6.1.4.1.8691.17.1.1.1
        productName OBJECT-TYPE
            SYNTAX DisplayString
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Showing product name, eg. UC7110-LX/IA240-LX/DA683-LX/DA683-XPE."
            ::= { productInfoMgmt 1 }

        -- 1.3.6.1.4.1.8691.17.1.1.2
        productDesc OBJECT-TYPE
            SYNTAX DisplayString
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Showing product short description.(if one exists)."
            ::= { productInfoMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.1.3
        productVersion OBJECT-TYPE
            SYNTAX DisplayString
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Showing product version eg. 1.0/1.0.1"
            ::= { productInfoMgmt 3 }

        -- 1.3.6.1.4.1.8691.17.1.1.4
        productBuildDate OBJECT-TYPE
            SYNTAX DisplayString
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Showing product last build date, the format is YYMMDDHH.
```

```
            eg. 2012/01/23 19:22 -> 12012319."
        ::= { productInfoMgmt 4 }

    -- 1.3.6.1.4.1.8691.17.1.2
    systemInfoMgmt OBJECT IDENTIFIER ::= { moxaSystem 2 }

    -- 1.3.6.1.4.1.8691.17.1.2.1
    systemObject OBJECT IDENTIFIER ::= { systemInfoMgmt 1 }

    -- 1.3.6.1.4.1.8691.17.1.2.1.1
    systemCpuUsage OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Show CPU usage rate (0-100 %). Eg. 38"
        ::= { systemObject 1 }

    -- 1.3.6.1.4.1.8691.17.1.2.1.3
    systemMemUsage OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Show memory usage rate (0-100 %). Eg. 57"
        ::= { systemObject 3 }

    -- 1.3.6.1.4.1.8691.17.1.2.1.5
    systemUptime OBJECT-TYPE
        SYNTAX Minutes
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The amount of time since this host was last initialized."
        ::= { systemObject 5 }

    -- 1.3.6.1.4.1.8691.17.1.2.1.6
    systemTotalUptime OBJECT-TYPE
        SYNTAX Minutes
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The amount of time from total boot up time."
        ::= { systemObject 6 }

    -- 1.3.6.1.4.1.8691.17.1.2.3
    systemStorageObject OBJECT IDENTIFIER ::= { systemInfoMgmt 3 }

    -- 1.3.6.1.4.1.8691.17.1.2.3.1
    systemMemorySize OBJECT-TYPE
        SYNTAX KBytes
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The amount of physical main memory contained by the host. Eg. 524288
```

```
            Note that this is same as hrMemorySize in HOST-RESOURCE."
        ::= { systemStorageObject 1 }

    -- 1.3.6.1.4.1.8691.17.1.2.3.2
    systemVolumeCount OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Show total volume count."
        ::= { systemStorageObject 2 }

    -- 1.3.6.1.4.1.8691.17.1.2.3.3
    systemVolumeTable OBJECT-TYPE
        SYNTAX SEQUENCE OF SystemVolumeEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of File System Volume and their values."
        ::= { systemStorageObject 3 }

    -- 1.3.6.1.4.1.8691.17.1.2.3.3.1
    systemVolumeEntry OBJECT-TYPE
        SYNTAX SystemVolumeEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry containing a device and its statistics."
        INDEX { systemVolumeIndex }
        ::= { systemVolumeTable 1 }

    SystemVolumeEntry ::=
        SEQUENCE {
            systemVolumeIndex
                Integer32,
            systemVolumeName
                OCTET STRING,
            systemVolumeLabel
                OCTET STRING,
            systemVolumeSize
                OCTET STRING,
            systemVolumeAvail
                OCTET STRING
         }

    -- 1.3.6.1.4.1.8691.17.1.2.3.3.1.1
    systemVolumeIndex OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reference index for each observed device."
        ::= { systemVolumeEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.2.3.3.1.2
```

```
        systemVolumeName OBJECT-TYPE
            SYNTAX OCTET STRING
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The name of the volume.
                   Eg. /dev/sda1
                   Eg. C:"
            ::= { systemVolumeEntry 2 }

        -- 1.3.6.1.4.1.8691.17.1.2.3.3.1.3
        systemVolumeLabel OBJECT-TYPE
            SYNTAX OCTET STRING
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The label of the volume.
                   Eg. System
                   Eg. Data"
            ::= { systemVolumeEntry 3 }

        -- 1.3.6.1.4.1.8691.17.1.2.3.3.1.4
        systemVolumeSize OBJECT-TYPE
            SYNTAX OCTET STRING
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The total size of the volume.
                   Eg. 100.9 MB
                   Eg. 3.6 GB"
            ::= { systemVolumeEntry 4 }

        -- 1.3.6.1.4.1.8691.17.1.2.3.3.1.5
        systemVolumeAvail OBJECT-TYPE
            SYNTAX OCTET STRING
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The available size of the volume.
                   Eg. 965 KB
                   Eg. 844.8 MB"
            ::= { systemVolumeEntry 5 }

        -- 1.3.6.1.4.1.8691.17.1.4
        biosMgmt OBJECT IDENTIFIER ::= { moxaSystem 4 }

        -- 1.3.6.1.4.1.8691.17.1.4.1
        biosVersion OBJECT-TYPE
            SYNTAX DisplayString
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Showing the BIOS version. eg. V1.00S01"
            ::= { biosMgmt 1 }
```

```
-- 1.3.6.1.4.1.8691.17.1.4.2
biosSaveSetting OBJECT-TYPE
    SYNTAX INTEGER
        {
        none(0),
        apply(1),
        discard(2)
        }
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "Write 1 to save bios setting, and read 0 mean setting had been applied."
    ::= { biosMgmt 2 }

-- 1.3.6.1.4.1.8691.17.1.4.3
biosSettingStatus OBJECT-TYPE
    SYNTAX INTEGER
        {
        same(0),
        modified(1)
        }
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Showing compare of bios CMOS setting and bios new setting."
    ::= { biosMgmt 3 }

-- 1.3.6.1.4.1.8691.17.1.4.4
bootSequence OBJECT IDENTIFIER ::= { biosMgmt 4 }

-- 1.3.6.1.4.1.8691.17.1.4.4.1
bootDeviceStatus OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Showing the current support boot device."
    ::= { bootSequence 1 }

-- 1.3.6.1.4.1.8691.17.1.4.4.2
firstBootDevice OBJECT-TYPE
    SYNTAX Integer32 (1..99)
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "read show current first boot device, write set boot device."
    ::= { bootSequence 2 }

-- 1.3.6.1.4.1.8691.17.1.4.8
powerFeature OBJECT IDENTIFIER ::= { biosMgmt 8 }

-- 1.3.6.1.4.1.8691.17.1.4.8.1
pwrOnAfterPwrFail OBJECT-TYPE
    SYNTAX INTEGER
        {
```

```
            off(0),
            on(1),
            former(2)
            }
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "Select power on after power fail behavior."
        ::= { powerFeature 1 }

    -- 1.3.6.1.4.1.8691.17.1.4.8.3
    pwrLanWakeUp OBJECT-TYPE
        SYNTAX INTEGER
            {
            disbale(0),
            enable(1)
            }
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "Enable/Disable wake on LAN functionality."
        ::= { powerFeature 3 }

    -- 1.3.6.1.4.1.8691.17.1.5
    sensorMgmt OBJECT IDENTIFIER ::= { moxaSystem 5 }

    -- 1.3.6.1.4.1.8691.17.1.5.1
    sensorObject OBJECT IDENTIFIER ::= { sensorMgmt 1 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.1
    tempSensorsTable OBJECT-TYPE
        SYNTAX SEQUENCE OF TempSensorsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of temperature sensors and their values."
        ::= { sensorObject 1 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.1.1
    tempSensorsEntry OBJECT-TYPE
        SYNTAX TempSensorsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry containing a device and its statistics."
        INDEX { tempSensorsIndex }
        ::= { tempSensorsTable 1 }

    TempSensorsEntry ::=
        SEQUENCE {
            tempSensorsIndex
                Integer32,
            tempSensorsDevice
                DisplayString,
            tempSensorsValue
```

```
                Integer32
        }

    -- 1.3.6.1.4.1.8691.17.1.5.1.1.1.1
    tempSensorsIndex OBJECT-TYPE
        SYNTAX Integer32 (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reference index for each observed device."
        ::= { tempSensorsEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.1.1.2
    tempSensorsDevice OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The name of the temperature sensor we are reading."
        ::= { tempSensorsEntry 2 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.1.1.3
    tempSensorsValue OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The temperature of this sensor in mC."
        ::= { tempSensorsEntry 3 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.2
    voltSensorsTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VoltSensorsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of voltage sensors and their values."
        ::= { sensorObject 2 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.2.1
    voltSensorsEntry OBJECT-TYPE
        SYNTAX VoltSensorsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry containing a device and its statistics."
        INDEX { voltSensorsIndex }
        ::= { voltSensorsTable 1 }

    VoltSensorsEntry ::=
        SEQUENCE {
            voltSensorsIndex
                Integer32,
            voltSensorsDevice
                DisplayString,
```

```
            voltSensorsValue
                Integer32
        }

    -- 1.3.6.1.4.1.8691.17.1.5.1.2.1.1
    voltSensorsIndex OBJECT-TYPE
        SYNTAX Integer32 (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reference index for each observed device."
        ::= { voltSensorsEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.2.1.2
    voltSensorsDevice OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The name of the device we are reading."
        ::= { voltSensorsEntry 2 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.2.1.3
    voltSensorsValue OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The voltage in mV."
        ::= { voltSensorsEntry 3 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3
    accelerometerTable OBJECT-TYPE
        SYNTAX SEQUENCE OF AccelerometerEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of accelerometer and their values."
        ::= { sensorObject 3 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3.1
    accelerometerEntry OBJECT-TYPE
        SYNTAX AccelerometerEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry containing a device and its statistics."
        INDEX { accelerometerIndex }
        ::= { accelerometerTable 1 }

    AccelerometerEntry ::=
        SEQUENCE {
            accelerometerIndex
                Integer32,
            accelerometerAxis
```

```
                DisplayString,
            accelerometerValue
                DisplayString,
            accelerometerTimestamp
                DisplayString
        }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3.1.1
    accelerometerIndex OBJECT-TYPE
        SYNTAX Integer32 (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reference index for each observed device."
        ::= { accelerometerEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3.1.2
    accelerometerAxis OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The name of the accelerometer axis we are reading."
        ::= { accelerometerEntry 2 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3.1.3
    accelerometerValue OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The accelerometer value in mG."
        ::= { accelerometerEntry 3 }

    -- 1.3.6.1.4.1.8691.17.1.5.1.3.1.4
    accelerometerTimestamp OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The timestamp when accelerometer measured."
        ::= { accelerometerEntry 4 }

    -- 1.3.6.1.4.1.8691.17.1.6
    peripheralMgmt OBJECT IDENTIFIER ::= { moxaSystem 6 }

    -- 1.3.6.1.4.1.8691.17.1.6.1
    perIoMgmt OBJECT IDENTIFIER ::= { peripheralMgmt 1 }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1
    ioObject OBJECT IDENTIFIER ::= { perIoMgmt 1 }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1.1
    ioDiNumber OBJECT-TYPE
        SYNTAX Integer32 (0..65535)
```

```
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of digital input pin in current system."
        ::= { ioObject 1 }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1.2
    ioDiTable OBJECT-TYPE
        SYNTAX SEQUENCE OF IoDiEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of digital input and their values."
        ::= { ioObject 2 }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1.2.1
    ioDiEntry OBJECT-TYPE
        SYNTAX IoDiEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "An entry containing a digital input pin and its statistics."
        INDEX { diIndex }
        ::= { ioDiTable 1 }

    IoDiEntry ::=
        SEQUENCE {
            diIndex
                Integer32,
            diPort
                Integer32,
            diValue
                INTEGER,
            diTrapEnable
                INTEGER
         }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1.2.1.1
    diIndex OBJECT-TYPE
        SYNTAX Integer32 (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Reference index for each digital input pin."
        ::= { ioDiEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.6.1.1.2.1.2
    diPort OBJECT-TYPE
        SYNTAX Integer32 (0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The port number of digital input pin."
        ::= { ioDiEntry 2 }
```

```
        -- 1.3.6.1.4.1.8691.17.1.6.1.1.2.1.3
        diValue OBJECT-TYPE
            SYNTAX INTEGER
                {
                low(0),
                high(1)
                }
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The digital input status, 0 is low, 1 is high."
            ::= { ioDiEntry 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.2.1.4
        diTrapEnable OBJECT-TYPE
            SYNTAX INTEGER
                {
                disable(0),
                enable(1)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Agent will send trap message when digital input pin status changed
and this object enbeled."
            ::= { ioDiEntry 4 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.3
        ioDoNumber OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Number of digital output pin in current system."
            ::= { ioObject 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.4
        ioDoTable OBJECT-TYPE
            SYNTAX SEQUENCE OF IoDoEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Table of digital output and their values."
            ::= { ioObject 4 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.4.1
        ioDoEntry OBJECT-TYPE
            SYNTAX IoDoEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "An entry containing a digital output pin and its statistics."
            INDEX { doIndex }
            ::= { ioDoTable 1 }
```

```
        IoDoEntry ::=
            SEQUENCE {
                doIndex
                    Integer32,
                doPort
                    Integer32,
                doValue
                    INTEGER
             }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.4.1.1
        doIndex OBJECT-TYPE
            SYNTAX Integer32 (1..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Reference index for each digital output pin."
            ::= { ioDoEntry 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.4.1.2
        doPort OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The port number of digital output pin."
            ::= { ioDoEntry 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.1.4.1.3
        doValue OBJECT-TYPE
            SYNTAX INTEGER
                {
                low(0),
                high(1)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "The digital output status, 0 is low, 1 is high."
            ::= { ioDoEntry 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.2
        ioNotification OBJECT IDENTIFIER ::= { perIoMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.1.2.1
        ioDiChange NOTIFICATION-TYPE
            STATUS current
            DESCRIPTION
                "This trap is sent when digital input pin status changed."
            ::= { ioNotification 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.2
        perLedMgmt OBJECT IDENTIFIER ::= { peripheralMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.2.1
```

```
        ledNumber OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Description."
            ::= { perLedMgmt 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.2.2
        ledTable OBJECT-TYPE
            SYNTAX SEQUENCE OF LedEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Description."
            ::= { perLedMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.2.2.1
        ledEntry OBJECT-TYPE
            SYNTAX LedEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Description."
            INDEX { ledIndex }
            ::= { ledTable 1 }

        LedEntry ::=
            SEQUENCE {
                ledIndex
                    Integer32,
                ledPort
                    Integer32,
                ledValue
                    INTEGER
             }

        -- 1.3.6.1.4.1.8691.17.1.6.2.2.1.1
        ledIndex OBJECT-TYPE
            SYNTAX Integer32 (1..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Description."
            ::= { ledEntry 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.2.2.1.2
        ledPort OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Description."
            ::= { ledEntry 2 }
```

```
        -- 1.3.6.1.4.1.8691.17.1.6.2.2.1.3
        ledValue OBJECT-TYPE
            SYNTAX INTEGER
                {
                off(0),
                on(1)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Description."
            ::= { ledEntry 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.3
        perSerialMgmt OBJECT IDENTIFIER ::= { peripheralMgmt 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.3.1
        uartNumber OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Number of internal UART in current system."
            ::= { perSerialMgmt 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.3.2
        uartConfigTable OBJECT-TYPE
            SYNTAX SEQUENCE OF UartConfigEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Table of internal UART and their values."
            ::= { perSerialMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.3.2.1
        uartConfigEntry OBJECT-TYPE
            SYNTAX UartConfigEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "An entry containing a UART port and its statistics."
            INDEX { uartIndex }
            ::= { uartConfigTable 1 }

        UartConfigEntry ::=
            SEQUENCE {
                uartIndex
                    Integer32,
                uartType
                    INTEGER
             }

        -- 1.3.6.1.4.1.8691.17.1.6.3.2.1.1
        uartIndex OBJECT-TYPE
            SYNTAX Integer32 (1..65535)
```

```
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Reference index for each UART port."
            ::= { uartConfigEntry 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.3.2.1.2
        uartType OBJECT-TYPE
            SYNTAX INTEGER
                {
                rs232(0),
                rs485w2(1),
                rs422(2),
                rs485w4(3)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "The UART mode, 0 is RS232, 1 is RS485 2 wires, 2 is RS422, 3 is RS485
4 wires."
            ::= { uartConfigEntry 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.4
        perUsbMgmt OBJECT IDENTIFIER ::= { peripheralMgmt 4 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.1
        usbObject OBJECT IDENTIFIER ::= { perUsbMgmt 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.1.1
        usbNumber OBJECT-TYPE
            SYNTAX Integer32 (0..65535)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The number of ports regardless of their current state
                in the usb general port table"
            ::= { usbObject 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.1.3
        usbDeviceTable OBJECT-TYPE
            SYNTAX SEQUENCE OF UsbDeviceEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "A list of USB device ports. Usually the device has
                only one USB device port"
            ::= { usbObject 3 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.1.3.1
        usbDeviceEntry OBJECT-TYPE
            SYNTAX UsbDeviceEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Status and parameter values for the USB device port."
```

```
        INDEX { usbDeviceIndex }
        ::= { usbDeviceTable 1 }

    UsbDeviceEntry ::=
        SEQUENCE {
            usbDeviceIndex
                Integer32,
            usbDeviceVendorID
                OCTET STRING,
            usbDeviceProductID
                OCTET STRING,
            usbDeviceActiveClass
                INTEGER
         }

    -- 1.3.6.1.4.1.8691.17.1.6.4.1.3.1.1
    usbDeviceIndex OBJECT-TYPE
        SYNTAX Integer32 (1..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The index is identical to usbPortIndex for the
            correspondent USB port"
        ::= { usbDeviceEntry 1 }

    -- 1.3.6.1.4.1.8691.17.1.6.4.1.3.1.2
    usbDeviceVendorID OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The USB device port vendor HEX-formatted string as it
            is provided to the USB host by the USB device"
        ::= { usbDeviceEntry 2 }

    -- 1.3.6.1.4.1.8691.17.1.6.4.1.3.1.3
    usbDeviceProductID OBJECT-TYPE
        SYNTAX OCTET STRING
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The product ID HEX-formatted string as it is provided
            to the USB host by the USB device"
        ::= { usbDeviceEntry 3 }

    -- 1.3.6.1.4.1.8691.17.1.6.4.1.3.1.4
    usbDeviceActiveClass OBJECT-TYPE
        SYNTAX INTEGER
            {
            other(1),
            hid(2),
            mass(3)
            }
        MAX-ACCESS read-only
        STATUS current
```

```
            DESCRIPTION
                "This object returns USB Device Class type of the
                active configuration"
            ::= { usbDeviceEntry 4 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.1.4
        usbPlugTrapEnable OBJECT-TYPE
            SYNTAX INTEGER
                {
                disable(0),
                enable(1)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Agent will send trap message when USB device inserted or removed and
this object enbeled."
            ::= { usbObject 4 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.2
        usbNotification OBJECT IDENTIFIER ::= { perUsbMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.4.2.1
        usbPlugEvent NOTIFICATION-TYPE
            STATUS current
            DESCRIPTION
                "This trap is sent when USB device inserted or removed."
            ::= { usbNotification 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.6
        perSystemMgmt OBJECT IDENTIFIER ::= { peripheralMgmt 6 }

        -- 1.3.6.1.4.1.8691.17.1.6.6.2
        systemWatchdog OBJECT IDENTIFIER ::= { perSystemMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.6.6.2.1
        watchdogPeriod OBJECT-TYPE
            SYNTAX Second (0..255)
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Watchdog period, 0 means disable watchdog monitor program; otherwise
enable watchdog monitor program and configure the expired time."
            ::= { systemWatchdog 1 }

        -- 1.3.6.1.4.1.8691.17.1.6.6.2.2
        watchdogStatus OBJECT-TYPE
            SYNTAX INTEGER
                {
                running(1),
                stopped(2)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
```

```
                "To show the watchdog monitor program status."
            ::= { systemWatchdog 2 }

        -- 1.3.6.1.4.1.8691.17.1.7
        powerMgmt OBJECT IDENTIFIER ::= { moxaSystem 7 }

        -- 1.3.6.1.4.1.8691.17.1.7.2
        powerPolicy OBJECT-TYPE
            SYNTAX INTEGER
                {
                balanced(1),
                power_saver(2),
                high_performance(3)
                }
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Current system power policy."
            ::= { powerMgmt 2 }

        -- 1.3.6.1.4.1.8691.17.1.9
        notificationMgmt OBJECT IDENTIFIER ::= { moxaSystem 9 }

        -- 1.3.6.1.4.1.8691.17.1.9.1
        moxaSystemTrapIP OBJECT-TYPE
            SYNTAX IpAddress
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Set Trap IP address. eg. 192.168.1.100"
            ::= { notificationMgmt 1 }

        -- 1.3.6.1.4.1.8691.17.1.9.2
        moxaSystemTrapCommunity OBJECT-TYPE
            SYNTAX OCTET STRING (SIZE (0..127))
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Trap community. eg. public"
            ::= { notificationMgmt 2 }

    END
--
-- MOXA-SYS-MIB.txt
--
```

C

Sample Scripts & Firewall Rules

Here are the sample scripts referenced in this manual that were too long to include in the text.

The following topics are covered in this appendix:

- **A Sample Initialization Script**
- **A Sample Firewall**

A Sample Initialization Script

```
#! /bin/sh
# Copyright (c) XXXX <<Your Name Here>>
# All rights reserved.
#
#
# /etc/init.d/<<name of your script here>>
#   and its symbolic link
# /usr/sbin/rc<<name of your script here>>

### BEGIN INIT INFO
# Provides:         <<name of your script here>>
# Required-Start:    $network
# Required-Stop:
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
# Short-Description: The <<name of your script here>> daemon provides....
# Description:       The <<name of your script here>> daemon is ...
#    that is active in runlevels 3 and 5.
#
### END INIT INFO

# Check for missing binaries
<<NAME OF YOUR SCRIPT HERE>>_BIN=/usr/bin/<<name of your script here>>
test -x $<<NAME OF YOUR SCRIPT HERE>>_BIN || { echo "$<<NAME OF YOUR SCRIPT HERE>>_BIN
not installed";
       if [ "$1" = "stop" ]; then exit 0;
       else exit 5; fi; }

# Check for existence of needed config file and read it
<<NAME OF YOUR SCRIPT HERE>>_CONFIG=/etc/<<name of your script here>>.cfg
test -r $<<NAME OF YOUR SCRIPT HERE>>_CONFIG || { echo "$<<NAME OF YOUR SCRIPT HERE>>_CONFIG
not existing";
       if [ "$1" = "stop" ]; then exit 0;
       else exit 6; fi; }

# Read config
. $<<NAME OF YOUR SCRIPT HERE>>_CONFIG

# Load the rc.status script for this service.
. /etc/rc.status

# Reset status of this service
rc_reset

case "$1" in
    start)
       echo -n "Starting <<name of your script here>> "
       ## Start daemon with startproc(8). If this fails
       ## the return value is set appropriately by startproc.
       startproc $<<NAME OF YOUR SCRIPT HERE>>_BIN

       # Remember status and be verbose
```

```
    rc_status -v
    ;;
stop)
    echo -n "Shutting down <<name of your script here>> "
    ## Stop daemon with killproc(8) and if this fails
    ## killproc sets the return value according to LSB.

    killproc -TERM $<<NAME OF YOUR SCRIPT HERE>>_BIN

    # Remember status and be verbose
    rc_status -v
    ;;
restart)
    ## Stop the service and regardless of whether it was
    ## running or not, start it again.
    $0 stop
    $0 start

    # Remember status and be quiet
    rc_status
    ;;
reload)
    # If it supports signaling:
    echo -n "Reload service bar "
    killproc -HUP $BAR_BIN
    #touch /var/run/<<NAME OF YOUR SCRIPT HERE>>.pid
    rc_status -v

    ## Otherwise if it does not support reload:
    #rc_failed 3
    #rc_status -v
    ;;
status)
    echo -n "Checking for service <<name of your script here>> "
    ## Check status with checkproc(8), if process is running
    ## checkproc will return with exit status 0.

    # Return value is slightly different for the status command:
    # 0 - service up and running
    # 1 - service dead, but /var/run/  pid  file exists
    # 2 - service dead, but /var/lock/ lock file exists
    # 3 - service not running (unused)
    # 4 - service status unknown :-(
    # 5--199 reserved (5--99 LSB, 100--149 distro, 150--199 appl.)

    # NOTE: checkproc returns LSB compliant status values.
    checkproc $<<NAME OF YOUR SCRIPT HERE>>_BIN
    # NOTE: rc_status knows that we called this init script with
    # "status" option and adapts its messages accordingly.
    rc_status -v
    ;;
*)
    ## If no parameters are given, print which are avaiable.
    echo "Usage: $0 {start|stop|status|restart|reload}"
```

```
      exit 1
      ;;
esac
rc_exit
```

A Sample Firewall

```
#!/bin/bash
# If you put this shell script in the /home/nat.sh
# Remember to chmod 744 /home/nat.sh
# Edit the rc.local file to make this shell startup automatically.
# vi /etc/rc.local
# Add a line in the end of rc.local /home/nat.sh
EXIF= "eth0"  #This is an external interface for setting up a valid IP address.
EXNET= "192.168.4.0/24"  #This is an internal network address.
# Step 1. Insert modules.
# Here 2> /dev/null means the standard error messages will be dump to null device.
modprobe ip_tables  2> /dev/null
modprobe ip_nat_ftp  2> /dev/null
modprobe ip_nat_irc  2> /dev/null
modprobe ip_conntrack  2> /dev/null
modprobe ip_conntrack_ftp  2> /dev/null
modprobe ip_conntrack_irc  2> /dev/null
# Step 2. Define variables, enable routing and erase default rules.
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
export PATH
echo "1" > /proc/sys/net/ipv4/ip_forward
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -Z
/sbin/iptables -F -t nat
/sbin/iptables -X -t nat
/sbin/iptables -Z -t nat
/sbin/iptables -P INPUT ACCEPT
/sbin/iptables -P OUTPUT ACCEPT
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -t nat -P PREROUTING ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P OUTPUT ACCEPT
# Step 3. Enable IP masquerade.
#ehco 1 > /proc/sys/net/ipv4/ip_forward
#modprobe ipt_MASQUERADE
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```