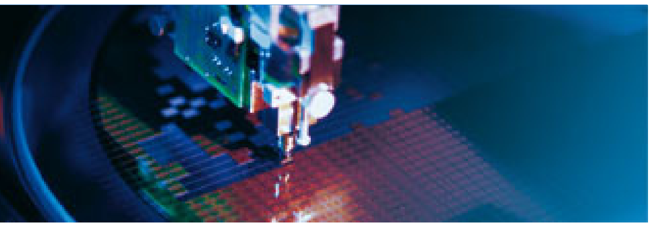
APPLICATION NOTE



An0075

ELD - Eurotech Linux Distribution

Rev 1.0 - January 2010 - ETH_An0075_AN1.0

DIGITAL TECHNOLOGIES FOR A BETTER WORLD

www.eurotech.com

EUROTECH

Disclaimer

The information in this manual has been carefully checked and is believed to be accurate. Eurotech assumes no responsibility for any infringements on patents or other third party rights, which may result from its use.

Eurotech assumes no responsibility for any inaccuracies that may be contained within this document, and makes no commitment to update or keep current the information contained within this manual.

Eurotech reserves the right to make improvements to this document and/or product at any time, without prior notice.

Trademarks

All trademarks both marked and not marked appearing in this document are the property of their respective owners.

Technical Assistance

For any technical questions, or if you cannot isolate a problem with your device, or for any enquiry about repair and returns policies, feel free to contact your local Eurotech Technical Support Team.

See the third and back covers for full contact details.

Revision History

REVISION	DESCRIPTION	DATE
1.0	First release	January 2010

Table of Contents

General Description

The *Eurotech Linux Distribution* (ELD) is a Linux distribution that has been tailor-made for Eurotech products with an emphasis towards specific markets, for example industrial, medical, defence, etc.

Specific attention has been given to all aspects that make up a robust Linux distribution.

Some of these key features are:

- Storage partition management
- Configuration management
- Counter-measures taken to avoid improper usage of fragile mass storage devices (DOM and CF)

System operation

Default runlevel: 3

Start-up

The *start-up* sequence occurs after the kernel has been loaded and executed.
The shell-script sequence is the ordered as shown below:

1	`mountfs`	Initialize mount points
2	`udev`	Start udevd, populate /dev and load drivers
3	`modules`	Load system specific device drivers
4	`syslog`	Start system log daemon
5	`firstboot`	Perform one-time configuration
6	`messagebus`	Initialize DBus messages daemon
7	`ppp`	Initialize PPP interfaces (if any configured
8	`network`	Configure Ethernet interfaces
9	`wireless`	Start Wi-Fi services
10	`gpsd`	Start GPS daemon
11	`sshd`	Start Open-BSD Secure Shell server
12	`thttpd`	Start thttpd web server
13	`ntpd`	Start NTP server
14	`portmap`	RPC port-mapper
15	`ntpdate`	NTP sync service
16	`local`	Additional initialization stuff

Shutdown

The shutdown sequence starts after invoking `poweroff`, `shutdown` or `halt` from the command line.
The shutdown sequence performs the following steps:

1. Sends the `TERM` signal to all active processes
2. Sends the `KILL` signal to any remaining active processes

Suspend and Resume

The system does not support Suspend mode or Resume, thus have not been implemented.

Kernel and modules

The Linux kernel is based on the Vanilla kernel release 2.6.22, with patches and drivers for Eurotech-specific devices.

Following is a list of the main kernel features available/implemented:

- CPU architectures:
  - AMD Geode Media-GX
  - AMD Geode GX/LX
  - Intel Pentium (i586)
  - PXA255 XScale processor (ARM architecture)
  - PXA270 XScale processor (ARM architecture)
- Audio support
- Networking support:
  - Networking TCP/IP stack support with:
    - TCP/IP with IPv4 and IPv6
    - Wireless 802.11, with WEP and 802.11i with CCMP and TKIP encryption (WPA, WPA2, WPA-PSK, etc)
  - NFS client/server support
  - SMB support
  - CIFS support
- Serial ports support for:
  - CPU PC-like standard serial ports
  - GSM/GPRS serial port
  - GPS serial port
  - COM-1274 multi-serial board
- CAN support for:
  - Intel 82527 CAN
  - Philips SJA1000
- Sensor temperature support:
  - MAX1618 for CPU temperature
- USB 2.0 support with the most common features:
  - Mass Storage
  - USB serial converter
  - etc
- ATA/ATAPI support
- PCI bus support
- ISA bus support
- File systems support:
  - ext2
  - ext3
  - ReiserFS
  - Aufs (Another Union FS)
  - ISO9660 and Aufs for CDROM
  - FAT and VFAT

Disk layout

The system is equipped with a 512MB Disk-On-Module, which is partitioned as follows:

PARTITION	SIZE (MB)	MOUNTPOINT	FILESYSTEM
hda1	79	/	ext3
Hda2	4.3	/etc	ext3
Hda3	412.5	/var	ext3

The partitions system implements the following features:

- The operating system is contained within its own partition and is mounted in read-only mode.
- The configuration data is contained within one or two overlapping directories, maintaining the factory, saved, and running configuration data.
- Some devices (/dev) need to be modified at run time by various applications, thus, even if the devices are contained in a read-only partition, it is possible to change them without any special techniques.
- A directory (/var/user) is available for user specific needs i.e. logging and storing information.

File system structure

The File systems structure is composed of three partitions:

- `OS` partition
- `Conf` partition
- `User` partition

The *OS* Partition

The OS partition contains the Operating System. It is read-only as it is believed that no one at run-time should be able to modify the OS root file system. This is the only partition modified by a system upgrade.

However, to enhance flexibility, a command does exist “`atomcmd`“, this in one shot places the root file system into read-write mode, executes a command and then resets the root partition to read-only mode.

The syntax is:

```
atomcmd _mount_point_ command
```

Therefore, to modify the root password, the user has to enter the following:

```
atomcmd / passwd
```

The *Conf* Partition

This partition has been explicitly created to manage configuration matters.

The partition is layered by means of a UnionFS file system (Newer releases now use the Aufs file system); this has been done to achieve a few goals.

To modify the usual configuration files, usually located under the `/etc` tree, that is within the read-only area in a comfortable way.

It will not lose the factory default settings, the settings the system has been built with.

To differentiate between the factory settings (`factory conf`), the settings saved for the system in use (`saved conf`) and the settings that are currently in use but that have not been saved yet (`running conf`).

Also the operating system upgrade must remain working.

The use of a UnionFS file system allows the use of a glass-pane stack paradigm, where in the system, 2 layers are put over the `/etc` directory. One is the directory `/.m/etc` this is where the real (physical) disk partition is mounted. The second is a RAM based file system, mounted on `/.m/rd/etc`; this disappears at shutdown along with any modified contents.

In the end, users will only see a single `/etc` directory, however, when modifications are made, these are actually done on the upper frame of the stack, the one that resides in RAM. As long as the system is powered, users will see (and the system will see) the last modification made.

When the system is shutdown, the modifications are lost as they were on a volatile file system.

If the `conf-save` command is used, the modifications are copied from the upper layer to the middle layer, the one actually mounted on the disk partition. These modifications are permanent, therefore, at reboot you will find the configurations previously saved.

However, these activities do not modify the 'real' `/etc` tree, so underneath the factory settings still remain.

The command `conf-factory` deletes all the modifications from the middle and upper layers, so, through the glass, users will see the real `/etc` again with the original factory configuration files.

Commands `conf-export` and `conf-import` can be used to export and import the "`saved conf`" to a file; this can be used to replicate or backup the saved configuration files, and therefore can be used for duplicating systems.

The *User* Partition

This partition holds the largest quantity of free space on the storage media and is mounted read-write.

The physical partition is mounted on `/.m/var`, and then, always with UnionFS, stacked over `/var`.

This allows users to save some useful structures under `/var` (`/var/run`, `/var/lock`, `/var/log`, etc) without the risk of modifying them and disposing of the free space. Basically everything written under `/var` will go onto this partition and will remain intact when the system is upgraded.

Services and configuration

The configuration of the distribution may be done at different levels:

Level	Description
distribution	This happens when users alter the distribution composition (i.e. change the graphical subsystem). This is a critical task and can be only performed by Eurotech.
package	This is when you want to change, for all your production, some specific characteristics that will not usually be changed in the field (i.e. change the port on which the internal web server can be accessed).
System / device	This is the typical configuration performed at installation time to distinguish between the different systems, for instance by changing the IP Address of the network interface.

User accounts

The only available account is '`root`'; the root user can log in via `ssh` (if enabled) or local console using the default password '`root`'.

Available services

At system start-up the following services are available:

- Secure shell server (`ssh`), on TCP Port 22
- thttpd web server, on TCP Port 80
- *gpsd*, on TCP Port 2947

System configuration

The system configuration is contained in the `/etc/sysconfig` directory and is organized in two main categories:

- Communication: containing configuration data related to the physical level communication
- System: containing the main system configuration data

Each category corresponds to a sub directory of */etc/sysconfig* containing a set of configuration files.

Directory	Contains	Configuration Files	Notes

/etc/sysconfig/communication	gprs	APN="ibox.tim.it"
		AUTH="noauth"
		USERNAME=""
		PASSWORD=""
			Contains all the parameters necessary to establish a GPRS connection using pppd

	wireless	ESSID=""
		MODE=""
		ENC=""
		CHANNEL=""
			Contains all the parameters necessary to associate the wireless module to a Wi-Fi network

/etc/sysconfig/system	autorun	SCRIPT=/var/autorun	Contains the path of a user definable script executed at boot time after the start-up procedure. (By default /var/autorun is an empty dummy script)

	mixer	MASTER_VOLUME="50"
		PCM_VOLUME="50"
		MIC_VOLUME="0"
		MIC_REC_VOLUME="50"
			Contains system mixer settings.
			Each Volume is shown as a percentage.

	network	ETH0_DHCP="no"
		ETH0_IP="10.100.10.100”
		ETH0_NETMASK="255.0.0.0"
		ETH0_BROADCAST="10.255.255.255"
		ETH1_DHCP="no"
		ETH1_IP=""
		ETH1_NETMASK=""
		ETH1_BROADCAST=""
		GATEWAY=""
		DNS1=""
		DNS2=""
			Contains network settings for all the system interfaces.
			ETH0_xxx keys correspond to Wired network settings.
			ETH1_xxx keys correspond to Wireless network settings if present.
			
			Network Default Configuration:
			• Interface: eth0
			• Type: Ethernet
			• IP Address: 10.100.10.100
			• Netmask: 255.0.0.0

Configuration management

On the system, three different configuration layers coexist, one overlapping the other: factory configuration, saved configuration and runtime configuration.

Configuration	Description
Factory	This is the out-of-the-box system configuration; it is possible to reset the system back to this configuration by executing conf-factory command.
Runtime	This is the “living” system configuration as modified by the user whenever a file is saved in the configuration directory. This configuration is stored in the RAM layer, therefore will be lost if rebooted. This layer provides the user with the possibility to test different configurations without the risk of losing a working configuration.
Saved	The saved configuration is the persistent user configuration that is created by saving the runtime configuration to the file-system. This layer is written to the Disk-On-Module, thus surviving a system reboot or power failure. The conf-save command is used to save the runtime configuration as the saved configuration. The conf-save command still preserves the integrity of the factory configuration; therefore it is still possible to reset the device to the factory settings.

Configuration import and export

Saved configurations can be exported and imported using the ZMODEM transfer protocol.

For this, users will need to have `sz/rz`, `zssh` installed on their computer and a network connection with the system (it is also possible to import and export the configuration through a serial connection using `Minicom` or `Kermit` instead of `zssh`).

Exporting configurations from system using *zssh*:

1. Go to the system command line
2. Execute the command “conf-export filename”
3. Press “ctrl + space”
4. Execute command “rz”
5. Logout

The configuration will be saved in the current working directory with the name `filename` assigned in step 2.

Importing configurations to the system using *zssh*

1. Go to the system command line
2. Execute command “conf-import”
3. Press “ctrl + space”
4. Execute command “sz filename”
5. Reboot

After a reboot the system will be configured with the imported configuration as long as no error messages appeared after step 4.

Logging facilities

Basic logging facilities are provided by the `Busybox syslogd` daemon replacement.

Monitoring facilities

No specific software for system monitoring is provided.

Demo software

No demo software is provided.

Test software

No test software is provided.

Custom system software

No custom system software is provided.

Software packages

Installed packages

Following is a complete list of all packages and relative licenses of the ELD (Eurotech Linux Distribution). For each of its products, Eurotech provides a compressed archive containing, in addition to the ELD binary files, a file named `packages.list`, which is a catalog of all the packages installed in the system.

Package	Website	License
`aircrack-ng-0.9.1`	www.aircrack-ng.org	GPLv2, OpenSSL
`alsa-lib-1.0.13`	www.alsa-project.org	LGPLv2
`alsa-utils-1.0.13`	www.alsa-project.org	GPLv2
`bash-3.0`	www.gnu.org/software/bash	GPLv2
`bluez-hcidump-1.31`	www.bluez.org	GPLv2
`bluez-libs-3.2`	www.bluez.org	GPLv2
`bluez-utils-3.2`	www.bluez.org	GPLv2, LGPLv2
`busybox-1.2.1`	www.busybox.net	GPLv2
`bzip2-1.0.2`	www.bzip.org	BSD
`confscripts-1.0`	Eurotech internal developing	GPLv2
`coreutils-5.3.0`	www.gnu.org/software/coreutils	GPLv2
`dbus-0.91`	www.freedesktop.org/wiki/Software/dbus	Academic Free License v2.1, GPLv2
`dhclient-2.0pl5`	www.isc.org/software/dhcp	BSD
`dosfstools-2.10`	www.daniel-baumann.ch/software/dosfstools	GPLv2
`dprocs-1.0`	oss.sgi.com/projects/gmemusage	GPL
`e2fsprogs-1.35`	e2fsprogs.sourceforge.net	GPLv2
`elvis-tiny-2.2.0`	elvis.the-little-red-haired-girl.org	Clarified Artistic License
`eurotech-demos-`*`eurotech-product-name`*	Eurotech internal developing	GPLv2
`eurotech-test-`*`eurotech-product-name`*	Eurotech internal developing	GPLv2
`expat-2.0.0`	expat.sourceforge.net	MIT License
`gawk-3.1.4`	www.gnu.org/software/gawk	GPLv2
`gdb-6.4`	www.gnu.org/software/gdb	GPLv2, LGPLv2
`gpsd-2.33`	gpsd.berlios.de	BSD
`grep-2.5.1`	www.gnu.org/software/grep	GPLv2
`grub-0.97`	www.gnu.org/software/grub	GPLv2
`gzip-1.3.5`	www.gzip.org	GPLv2
`hddtemp-0.3-beta15`	savannah.nongnu.org/projects/hddtemp	GPLv2
`ioapps-1.0`	Linux device Driver 2nd edition - examples	GPLv2
`iperf-2.0.2`	iperf.sourceforge.net	BSD
`kernel-2.6.22-LDB`	www.kernel.org	GPLv2[1]
`juicer-1.0, juicer-base-1.0`	Eurotech internal developing	GPLv2
`ld-2.3.2`	www.gnu.org/software/binutils	GPLv2
`less-382`	www.gnu.org/software/less	GPLv2
`libao-0.8.6`	www.xiph.org/ao	GPLv2

[1] Due to U.S. Exports Regulations, all cryptographic software on this site is subject to the following legal notice:
This site includes publicly available encryption source code which, together with object code resulting from the compiling of publicly available source code, may be exported from the United States under License Exception "TSU" pursuant to 15 C.F.R. Section 740.13(e).
This legal notice applies to cryptographic software only. Please see the Bureau of Industry and Security for more information about current U.S. regulations.

libc-2.3.2	www.gnu.org/software/libc	LGPLv2
libcrypt-1.0	www.gnu.org/software/libc	LGPLv2
libdl-2.3.2	www.gnu.org/software/libc	LGPLv2
libgcc-1.0	gcc.gnu.org	LGPLv2
libid3tag-0.15.0b	www.underbit.com/products/mad	GPLv2
libm-2.3.2	www.gnu.org/software/libc	LGPLv2
libmad-0.15.0b	www.underbit.com/products/mad	GPLv2
libnsl-2.3.2	www.gnu.org/software/libc	LGPLv2
libnss-2.3.2	www.gnu.org/software/libc	LGPLv2
libpcap-0.9.3	www.tcpdump.org	BSD
libpthread-0.10	www.gnu.org/software/libc	LGPLv2
libresolv-2.3.2	www.gnu.org/software/libc	LGPLv2
librt-2.3.2	www.gnu.org/software/libc	LGPLv2
libstdc++-1.0	gcc.gnu.org/libstdc++	LGPLv2
libthread_db-1.0	www.gnu.org/software/libc	LGPLv2
libusb-0.1.12	www.libusb.org	LGPLv2
libutil-2.3.2	www.gnu.org/software/libc	LGPLv2
lighttpd-1.4.12	www.lighttpd.net	BSD
lrzsz-0.12.20	www.ohse.de/uwe/software/lrzsz.html	GPLv2
ltrace-0.3.36	www.ltrace.org	GPLv2
minicom-2.1	alioth.debian.org/projects/minicom	GPLv2
mpg321-0.2.10	mpg321.sourceforge.net	GPLv2
ncurses-5.4	www.gnu.org/software/ncurses	GPLv2
net-tools-1.60	freshmeat.net/projects/net-tools	GPLv2
ntp-4.2.0	www.ntp.org	NTP
openobex-1.3.4	dev.zuckschwerdt.org/openobex	GPLv2, LGPLv2
openssh-4.0p1	www.openssh.org	BSD
openssl-0.9.8h	www.openssl.org	OpenSSL[2]
pciutils-2.2.4	mj.ucw.cz/pciutils.html	GPLv2
portmap-5-26	neil.brown.name/portmap	BSD
ppp-2.4.3	www.samba.org/ppp	BSD, GPL, GPLv2, LGPLv2
procps-3.2.7	procps.sourceforge.net	GPLv2, LGPLv2
qdecoder-7.1.1	www.qdecoder.org/	LGPLv2
readline-4.3	cnswww.cns.cwru.edu/php/chet/readline/rltop.html	GPLv2
sed-4.1.2	www.gnu.org/software/sed	GPLv2
setserial-2.17	setserial.sourceforge.net/	GPL
strace-4.5.15	strace.sourceforge.net	BSD, GPLv2
stress-1.0	weather.ou.edu/~apw/projects/stress/	GPLv2
sudo-1.6.8	www.sudo.ws	ISC-style, UCB
sysvinit-2.86	freshmeat.net/projects/sysvinit	GPLv2
tar-1.13.93	www.gnu.org/software/tar	GPLv2
tcpdump-3.9.3	www.tcpdump.org	BSD
thttpd-2.25b	www.acme.com/software/thttpd	BSD, GPLv2
udev-100	www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html	GPLv2
update-rc.d-0.7	ipkgfind.handhelds.org/details.phtml?package=update-rc.d	GPLv2
usbutils-0.72	www.linux-usb.org	GPLv2
util-linux-2.12r	ftp://ftp.kernel.org/pub/linux/utils/util-linux/	GPL, GPLv2, SUN
zlib-1.2.3	www.zlib.net	zlib

[2] Apache-style licence

Auxiliary packages

The following is a list of packages (and relative licenses) required for the LDB building:

Package	Website	License
clay-1.0.0-r5	*Eurotech internal developing*	*BSD*
clearsilver-0.10.3-r0	*www.clearsilver.net/*	*Apache License v2.0*
fakeroot-1.2.13	*freshmeat.net/projects/fakeroot*	*GPL*
gettext-native-0.14.1	*www.gnu.org/software/gettext*	*GPLv2*
gnu-config	*savannah.gnu.org/projects/config*	*GPLv2*
libelf-0.8.6	*www.mr511.de/software*	*LGPLv2*
m4-1.4.2	*www.gnu.org/software/m4*	*GPLv2*
pkgconfig-0.15.0	*pkg-config.freedesktop.org*	*GPLv2*
quilt-0.42	*savannah.nongnu.org/projects/quilt*	*GPLv2*

System maintenance

System maintenance includes installing or upgrading/downgrading system software or performing custom actions on the target system.

Maintenance Software - MS

System maintenance is performed using the Maintenance Software (MS) which is provided by Eurotech. MS needs to be run from a USB pen-drive; this must be prepared as described in the next section.

MS is a small Linux distribution specifically made to operate on the target system without involving existing installed software.

MS is distributed in the form of a Zip archive whose content must be extracted to the pen-drive’s root directory. The archive contains the following files:

File	Description
linux	Maintenance Software (MS)
PART1.IMG	The actual firmware that will be written to the target system.
action	A text file that contains the action to be performed after boot.
syslinux.cfg	A configuration file used by the `bootloader` (DO NOT EDIT)

When the system boots from the USB pen-drive, the MS initializes the target system and passes control to a procedure which performs the action that is declared in the action file.

Possible actions are as follows:

Action	Description
`install`	Initializes the target system and install the Operating System. This includes partitioning the internal storage device and creating a new file system on it. The does not preserve any existing data.
`update`	Installs the Operating System without partitioning and or creating a new file system. Preserves existing user data under */var/user*.
`custom`	If there is a file named *custom.sh* in the base directory of the pen-drive it will be executed. The *custom.sh* file should actually be a Bash script.

The `install` and `update` actions are predefined procedures that cannot be modified by the user, but can be invoked in the `custom` script.

MS pen-drive setup

The Maintenance Software needs to be installed on a USB pen-drive[3] (preferably USB 2.0 compliant) with at least 128MB of free space.

To create the *MS pen-drive*, from a Linux system, follow these steps (as *root* user):

Step #		Command
1.	Create a “FAT16” or “FAT32” *bootable* partition on the pen-drive using the *fdisk* command *Note*: remember to set up the *bootable* flag!	`fdisk /dev/sdX`
2a.	Install the *ms-sys* package and put the MBR on the pen-drive. “`sdX`” is the device name that corresponds to the target pen-drive. *Note:* the *ms-sys* package is available from: http://ms-sys.sourceforge.net/	`ms-sys -s /dev/sdX`

2b.	Another way to place the MBR in its place is using mbr.bin a 512 byte file from the *syslinux* package.	`locate mbr.bin`
		`cat /somepath/share/syslinux/mbr.bin > /dev/sdX`

3.	Create an “FAT16” or “FAT32” partition on the pen-drive using the *mkfs.vfat* command *Note:* "-F 32" for FAT32 and "-F 16" for FAT16	`mkfs.vfat -F 32 /dev/sdX1`
4.	Make it bootable	`syslinux -s /dev/sdX1`

The USB-key prepared in this way is *called MS USB pen-driver* or *MS USB key*.

System software management

Using the same MS USB pen-drive, the user can perform different actions on the target system, simply by changing the content of the “action” file. The “action” file must contain one of the following commands:

- install
- clone
- custom

Each command involves a different procedure on the target system, as explained in the following sections.

[3] Note: sometimes the *USB pen-drive* is also called *USB-key*.

Installing/upgrading procedure

After preparing the *MS USB pen-drive*, as described in the previous paragraph, extract the content of the MS zip file into the base directory of the pen-drive.

Note: For Installing or Upgrading the “action” file must only contain the word “install”

To install or upgrade the system software, perform the following actions:

1. Turn off the target system.
2. Plug the previously prepared pen-drive into the target systems USB port.
3. Turn on the target system.
4. Now the following process should occur:
    - Firstly the Red LED 4 will start blinking.
    - Secondly the Green LED 3 will light up.
5. At this point the upgrade procedure will have been correctly completed.
6. Cycle the system power; it will now automatically load the new firmware.

Note:
If steps 4 or 5 fail, the procedure failed, the system is not upgraded and it will load the previous firmware when next booted.

Warning:
Power loss during the firmware writing step can be very dangerous and may compromise the firmware image

Cloning procedure

The *cloning* procedure is used to store the 2nd and the 3rd partitions of the target system to the MS USB key. These two partitions contain the user customizations and data (see the “Disk Layout” paragraph).

Note that the 1st partition is not involved in this procedure, because it is already contained in PART1.IMG file of the Eurotech MS release package.

Note: For Cloning the “action” file must only contain the word “clone”

When the cloning procedure is terminated (a big “OK” appears on the screen), two new files will have been written in the MS USB key:

- *partc2.img* (for the 2nd partition)
- *partc3.img* (for the 3rd partition)

Now the USB pen-drive is ready for the install procedure, as described in the previous paragraph (remember to change the content of the “action” file from *clone* to *install*).

Custom actions procedure

Custom actions can be performed by creating a Bash script called `custom.sh`, this must then be placed in the base directory of the pen-drive.

Note: For Custom actions the “action” file must only contain the word “custom”

After MS has initialized the target system, its resources are made available as follows:

- DOM is mapped to `/dev/hda`
- Pen-drive is mapped to `/dev/sda`. Its contents are available under `/mnt/usb`

⚠ **Warning:**
Custom scripts in MS have full access to the target system and its resources.
Improper use of the facilities supplied in the MS and used in a script may damage the existing software and/or setup of the target system.
Users have full responsibility for the consequences of implementing and using custom maintenance procedures.

Backup and restore

No backup / restore facilities or procedures have been set up.

Fallback and recovery

No fallback / recovery facilities or procedures have been set up.

Software development support: Sandbox

Sandbox is the response to the challenge of cross-compilation and a resolution for the libc-not-up-to-date nightmare.

By isolating the compilation activities inside a well-defined area, Sandbox helps the user with the daunting task of building up an application. The difficulties arise when cross compilation becomes necessary, which is to build an application for a non-x86 platform when using an x86 machine. The cross-compilation chain of programs, called the toolchain, is a combination of compilers, utilities, kernel headers, etc.

Also if we have to build an x86 application on an x86 machine, that is without the need of cross compilation, the tracking process could be difficult: we must link with libraries with release numbers less or equal to those installed on our target system.

The solution to this is a pre-baked environment containing the same toolchain used to build up the distribution and the same libraries used inside the target.

The Sandbox is a little self-contained Linux distribution that permits all of this.

The installer is a self extracting archive, to install the Sandbox execute it (as root user) and follow the onscreen procedure. During this procedure two parameters are required: the installation directory and the Sandbox user. The Sandbox contains both the compiler for the host machine (gcc, g++, etc), and one or more tool chains for the target platforms, for example usable through `arm-eurotech-linux-gcc`, `mips-eurotech-linux-gcc`, `i386-eurotech-linux-gcc` etc.

Supposing that the Sandbox is installed in the directory `/var/lib/sandbox`, the most important directory for the user is `/var/lib/sandbox/workspace`. The rest of the directories are Sandbox system folders, these are read-only. Inside the workspace directory, the user will position his applications in order to use the cross-compilation facilities provided by the Sandbox.

There are three possible methods to compile “`helloworld.c`” inside the workspace directory:

Command	Description
`gcc -o helloworld helloworld.c`	The output executable will be compiled using the native compiler of the host machine.
`sbx gcc -o helloworld helloworld.c`	The output executable will be compiled using the Sandbox native compiler.
`sbx i386-eurotech-linux-gcc -o helloworld helloworld.c`	The output executable will be compiled using the x86 cross-compiler of the Sandbox. This file can be executed on the system device. The toolchain available to the user is the same one used for the whole operating system build process.

Warning:
To compile and correctly build software for the system, users must use the i386-eurotech-linux-gcc cross compiler.

The commands installed with the *Sandbox* are:

Command	Description
`sbx`	Run this command inside *Sandbox*. The *Sandbox* root is defined by the environment variable SBX_ROOT. The user that is defined by the environment variable SBX_USER executes the command.
`sbx-sudo`	Same as sbx, but the command is executed with root privileges.
`sbx-sh`	Jump inside the *Sandbox* for an interactive session with the privileges of the user defined by the environment variable SBX_USER. Exit to go out.
`sbx-su`	Same as sbx-sh, but the command is executed with root privileges

Warning:
All the commands above are installed in *suid* mode. Thus are executed always with super user (root) privileges.
Even if many precautions have been taken to prevent any malicious usage, these commands still represent a security risk.

Libraries

Libraries for cross compilation are provided with *Sandbox*, inside the `/toolchains` directory, it is possible to find the proper toolchain for system's architecture.

For example: if system architecture is i386, all the necessary libraries and includes will be found in the `/toolchains/i386` directory.

Any additional library the developer wants to add to the Sandbox can be placed in this part of the file-system.

Additional tools

It is possible to personalize the tools present in the *Sandbox* by using the APT packaging system; all the packages will be downloaded from the Debian repository thus the computer must be connected to Internet.

Refer to *apt-get(8)* man pages for usage information.

Cross-compiler version

- GCC: 4.0.2
- GLIBC: 2.3.2

Main *Sandbox* directories

This is an example for *i386* architecture (all is similar for *ARM* and *MIPS* architectures):

- Cross-compiler directory: `/toolchains/i386/bin`
- Tool-chain directory: `/toolchains/i386-root`
- Cross-compilation libraries: `/toolchains/i386-root/lib`
- Cross-compilation headers: `/toolchains/i386-root/include`
- Work directory: `/workspace`

Appendix

ELD - Eurotech Linux Distribution building procedure

The *Eurotech Linux Distribution (ELD)* can be customized and rebuild by the user itself, according its proper requirements.

ELD building details

To compile a new ELD the following software is required:

- product-development sandbox

The product-development sandbox, an extension of the standard product sandbox, contains some additional packages to build the ELD. As for the standard product sandbox, the installer file is a self extracting archive: execute it as root user and then follow the onscreen instructions. For further details see the chapter regarding the installation procedure of the standard sandbox.

Note:
The *product-development sandbox* is customized for each Eurotech product. Ask Eurotech to obtain this software according to the particular product used.

After installing the product-development sandbox and moving to the sandbox installation directory (default: `/var/lib/sandbox`), execute the following procedure:

```
cd /var/lib/sandbox/workspace/distro
sbx make PRODUCT=eurotech-product-name
```

If you have doubts about the name of your Eurotech product (here simply called `eurotech-product-name`), contact the technical support.
After a while the `PART1.IMG` file will be created in the directory:

```
/var/lib/sandbox/workspace/distro/eurotech-product-name
```

Use this file with the MS USB pen-drive to install the new ELD into your system (see the chapter: System Maintenance).

ELD customization details

The ELD is based on the `openembedded` software framework. All the ELD configuration files and recipes are contained in the following directories:

```
../sandbox/bitbake/org.eurotech.dev/conf/distro/eurotech-product-name.conf
../sandbox/bitbake/org.eurotech.dev/packages
```

For further information, manuals and FAQ see the websites:

http://wiki.openembedded.net/index.php/Main_Page

http://developer.berlios.de/projects/bitbake/

MS – Maintenance Software building procedure

As seen in one of the previous paragraphs, the Maintenance Software consists of a small USB-key bootable Linux distribution, specifically designed for system upgrading/customizing/maintaining operations.

As well as all the Linux distributions, the MS is composed of a root file system and a Linux kernel, both of them entirely contained in a single file called `linux`.

To compile a new MS the following software is required:

- Product-development sandbox
- Linux kernel sources

The *product-development sandbox,* an extension of the standard *product sandbox*, contains some additional packages to build the MS root file system, called *juicer* distribution. Finally the Linux kernel is compiled using the *juicer* file system as `initramfs`, to obtain the final USB-key bootable file `linux`.

Note:
Both the *product-development sandbox* and the *Linux kernel sources* are customized for each Eurotech product. Ask Eurotech to obtain this software according to the particular product used.

MS building details

The following steps allow users to build the USB-key bootable MS file `linux for an x86 architecture`:

1. Compile the ELD as described in the previous chapter:

   ```
   cd /var/lib/sandbox/workspace/distro
   sbx make PRODUCT=eurotech-product-name
   ```

2. Compile the *juicer* file system:

   ```
   sbx-sudo bitbake juicer
   sbx-sudo bitbake juicer –c raw_sysroot
   sbx-sudo makedev eurotech-product-name/raw_sysroot/dev
   ```

3. Uncompress the *juicer* linux kernel sources:

   ```
   cp ....../linux-2.6-juicer.tar.bz2 /var/lib/sandbox/distro
   cd /var/lib/sandbox/workspace/distro
   tar xfj linux-2.6-juicer.tar.bz2
   ```

4. Prepare the linux kernel configuration file:

   ```
   cd linux-2.6
   sbx make distclean
   cp arch/x86/configs/duracor[4]-juicer_defconfig .config
   ```

   Inside the .config file, change the macro `CONFIG_INITRAMFS_SOURCE` to the following value:

   ```
   CONFIG_INITRAMFS_SOURCE=”/workspace/distro/eurotech-product-
   name/raw_sysroot”
   ```

   then:

   ```
   make menuconfig
   ```

   save the configuration file and exit.

5. Compile the final USB-key bootable Linux kernel:

   ```
   sbx make
   ```

6. Rename the new kernel file in `linux`:

   ```
   cp arch/x86/boot/bzImage[5]
   mv bzImage linux
   ```

[4] For other architectures (ARM, MIPS, …) contact the Eurotech technical support.

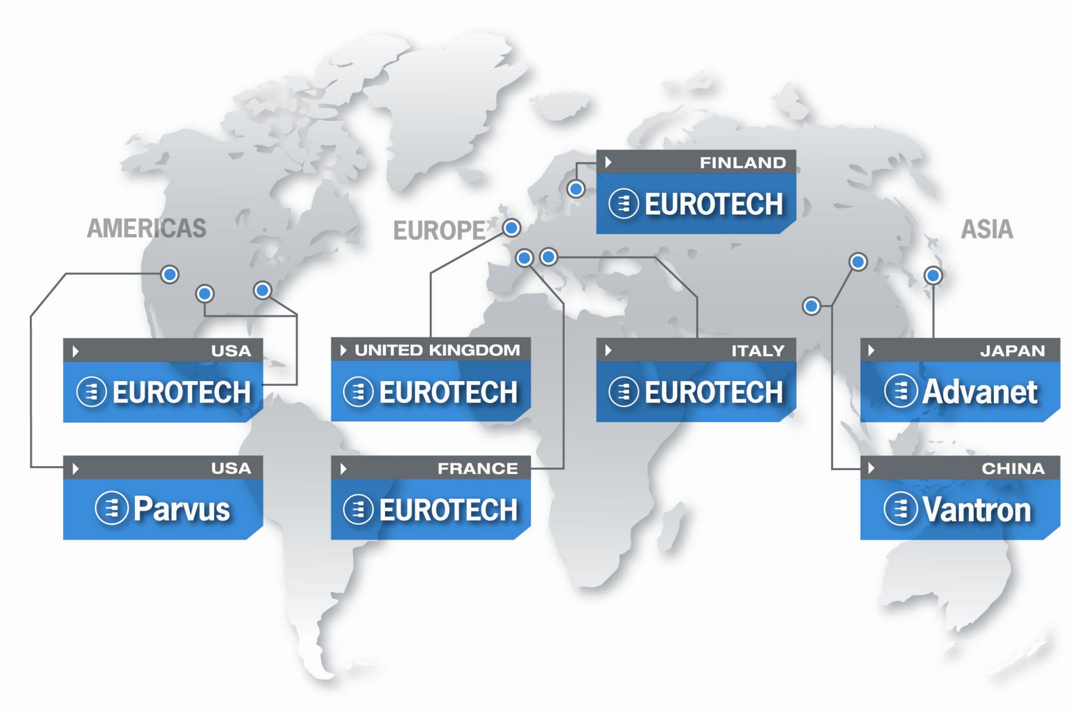
[5] This is an *x86* architecture example. For *ARM* and *MIPS* architecture refers to the proper directory names.

Manual Revision History

Revision	Description	Date
1.0	First release	February 2010

(This page has been intentionally left blank)

Eurotech Worldwide Presence



AMERICAS

USA

EUROTECH
Toll free +1 888.941.2224
Tel. +1 301.490.4007
Fax +1 301.490.4582
E-mail: sales.us@eurotech.com
E-mail: support.us@eurotech.com
Web: www.eurotech-inc.com

PARVUS
Tel. +1 800.483.3152
Fax +1 801.483.1523
E-mail: sales@parvus.com
E-mail: tsupport@parvus.com
Web: www.parvus.com

EUROPE

Italy

EUROTECH
Tel. +39 0433.485.411
Fax +39 0433.485.499
E-mail: sales.it@eurotech.com
E-mail: support.it@eurotech.com
Web: www.eurotech.com

United Kingdom

EUROTECH
Tel. +44 (0) 1223.403410
Fax +44 (0) 1223.410457
E-mail: sales.uk@eurotech.com
E-mail: support.uk@eurotech.com
Web: www.eurotech.com

France

EUROTECH
Tel. +33 04.72.89.00.90
Fax +33 04.78.70.08.24
E-mail: sales.fr@eurotech.com
E-mail: support.fr@eurotech.com
Web: www.eurotech.com

Finland

EUROTECH
Tel. +358 9.477.888.0
Fax +358 9.477.888.99
E-mail: sales.fi@eurotech.com
E-mail: support.fi@eurotech.com
Web: www.eurotech.com

ASIA

Japan

ADVANET
Tel. +81 86.245.2861
Fax +81 86.245.2860
E-mail: sales@advanet.co.jp
E-mail: tsupport@advanet.co.jp
Web: www.advanet.co.jp

China

VANTRON
Tel. +86 28.85.12.39.30
Fax +86 28.85.12.39.35
E-mail: sales@vantrontech.com.cn
E-mail: support.cn@eurotech.com
Web: www.vantrontech.com.cn

To find your nearest contact refer to: www.eurotech.com/contacts

EUROTECH

www.eurotech.com

EUROTECH HEADQUARTERS
Via Fratelli Solari 3/a
33020 Amaro (Udine) – ITALY
Phone: +39 0433.485.411
Fax: +39 0433.485.499

For full contact details go to: www.eurotech.com/contacts