Technical Manual

Kontron embedded Linux for Atom Processor

Revision 01

Revision History			
Rev.	**Description**	**Date**	**Author**
01	Initial Release. Complete revision of the old technical manual.	17.08.10	M. Kaiser

Table of Contents

Preface

This document describes the special structure and technical details of the Kontron embedded Linux image.

Software Components

Main Components

Name	Version	Home Page	Description
Linux kernel	2.6.31.4	www.kernel.org	The central part of the Linux OS. Contains system services and drivers.
X-server Xorg	1.3.0 7.5	www.x.org	X Window System X.org, provides a client/server interface between display hardware and the desktop environment while also providing both the windowing infrastructure and a standardized application interface (API). In short, X.org, a project derived from XFree86, is an open source X11-based desktop infrastructure. Included are plenty of video and input device drivers and X related applications.
IceWM	1.2.37	www.icewm.org	IceWM is a window manager for the X11 Window System. The goal of IceWM is speed, simplicity, and not getting in the user's way.
tsharc	3.0.4	www.microchip.-com	Hampshire (now microchip) touch screen driver and calibration tool
xpdf	3.02	www.foolabs.com	tiny open-source pdf reader
joe	3.5	sourceforge.net	joe is a powerful console-based ASCII-text-screen editor.

Name	Version	Home Page	Description
nano	2.0.6	www.nano-editor.org	nano is a small emulation of the pico editor.
Firefox	1.5.0.9	www.mozilla.org	the mozilla firefox open-source web-browser
Java VM	1.6.0.17	www.sun.com	Java allows the use of OS inde-pendent applets in the internet browser.
splashy	0.3.13	alioth.debian.org	userspace boot-splash utility
xvkbd	3.0	xvkbd	virtual keyboard for X11

Other Typical Tools and Libraries

These are the not so prominent features of a Linux image.

Name	Version	Name	Version
binutils	2.20	diffutils	2.8.1
file	5.03	e2fsprogs	1.41.8
gcc	4.4.3	expat	2.0.1
glibc	2.11.1	expect	5.43.0
gmp	4.3.1	findutils	4.4.2
mpfr	2.4.1	flex	2.5.35
zlib	1.2.3	fontconfig	2.3.2
directfb	1.0.1	freetype	2.1.10
autoconf	2.64	gawk	3.1.7
automake	1.11	gdbm	1.8.3
bash	4.0	gettext	0.17
bison	2.4.1	grep	2.5.4
bzip2	1.0.5	groff	1.20.1
coreutils	7.4	grub	0.97
dejagnu	1.4.4	gzip	1.3.12

Software Components

Name	Version	Name	Version
iana_etc	2.30	sed	4.2.1
inetutils	1.6	shadow	4.1.4.2
iproute2	2.6.29-1	syslogd	1.5
kbd	1.15	sysvinit	2.86
less	429	tar	1.22
libdrm	2.4.14	tcl	8.5.7
libpng	1.2.34	texinfo	4.13a
libtool	2.2.6a	udev	145
m4	1.4.13	util_linux	2.16
make	3.81	xterm	223
man	1.6e	glib	2.12.12
manpages	2.43	jpeg	6b
mesa	7.5.2	tiff	3.8.2
mktemp	1.5	cairo	1.2.4
module_init_tools	3.10	pango	1.12.3
ncurses	5.7	atk	1.18.0
patch	2.5.9	gtk+	2.10.13
perl	5.10.0	libidl	0.8.13
pixman	0.16.2	zip	2.32
pkg_config	0.23	giflib	4.1.4
procps	3.2.8	lesstif	0.95.2
psmisc	22.8		
readline	6.0		

Embedded Linux Image

The embedded image, running on an ext2 file system requires at least a 1GB Compact Flash (CF) card (current size 720 MB) and is designed to run in one of four possible screen resolutions: 800x480, 800x600, 1024x768, or 1280x1024. The resolution has to match the display resolution and is not meant to be changed while the system is running.

Image Structure

The image is based on the LFS (Linux from scratch) and BLFS (beyond Linux from Scratch) projects (www.linuxfromscratch.org), a toolchain is available from Kontron on request.

To better protect the image from data corruption by power-fail the CF is mounted read-only. The directories /etc and /var have to have write access, though, because most Linux processes store logging or configuration information there. The solution is to mount /etc and /var on a RAM file system while the persistent configuration information is stored in a different directory.

Root File System Top-Level Directories	
Directory	**Description**
/bin	system tools, open to all users
/boot	kernel and bootloader
/dev	device nodes
/.etc	persistent configuration files; permanent configuration changes have to be stored here; they are copied to /etc during the boot process
/etc	mounted on a RAM file system, configuration changes made here are lost after a reboot
/lib	kernel driver modules
/opt	third party software, mostly
/proc	plenty of information about the running kernel
/sbin	system tools, available to the administrator (root)

Root File System Top-Level Directories	
/sys	information about kernel drivers
/tmp	mount point of the RAM file system
/usr	applications and tools, but not essential system tools
/.var	log files can be stored here if necessary
/var	log files

Advanced Configuration

General Remarks

This section describes configuration tasks that are not available in the IceWM menu. It assumes that the reader is familiar with the basic handling of the Kontron Linux image as it is laid down in the “Quick Manual”.

Since the image is mounted read-only it has to be remounted to a writable state before any files can be changed. To do this open a terminal window then type

“mount -o remount,rw /”.

If big files are copied, Linux will often return to the command prompt before finishing internally. To make sure all buffers are empty issue the command

“sync”.

When you are done, it is good practice to switch back to read-only with

“mount -o remount,ro /”.

Do not simply pull the power plug because doing so might result in the loss of data or a corrupted file system.

X-server

The first process started by the Linux kernel after has been loaded is /sbin/init. The behavior of the init process is determined by the configuration file /etc/inittab. (See the listing in the appendix p.17.) In the first line the default runlevel is determined (5 in this case). The X-server is started when

Linux boots to the runlevels 4 or 5 (last line). To avoid booting into the GUI the easiest way is to change the default runlevel. Set it to 3:

“id:3:initdefault:”

and the X-server will not be started automatically.

Note: You cannot change /etc/inittab on the CF from which you have booted the system. The /etc/inittab of the running system is a copy of /.etc/inittab residing in a RAM file system. It has to be distinguished from the file /etc/inittab that is used by /sbin/init early in the boot process.

Boot splash screen

The picture displayed by splashy as the boot splash screen is located in

/usr/local/share/splashy/themes/default.

To set your own splash image simply replace the file

background.png.

IceWM configuration

The appearance and some of the functionality of the IceWM window manager is controlled by several human readable configuration files located in

/usr/local/share/icewm.

This section explains a few of them.

keys

An example of this file is given in the appendix (p.17). With the help of this file it is possible to connect hotkeys with Linux commands. The syntax is very simple:

- Put a new entry on a new line.
- Start the entry with the keyword “key”.
- Next comes the definition of the key combination.
- The last item is a Linux command that should be executed.

key "Alt+Ctrl+t" xterm /bin/bash ,

e.g., instructs IceWM to start an xterm with a bash shell if the user presses the key combination **<Alt>+<Ctrl>+t**.

menu

This file defines the content and appearance of the IceWM menu (see p.18). This is the syntax:

- Put a new entry on a new line.
- Start the entry with one of the keywords “prog”, “menu”, or “separator”.
  - “separator” draws a line to structure groups of menu entries.
  - “prog” defines commands:

    prog <name shown in the menu> <icon> <Linux command>.
  - “menu” allows to organize related commands into sub-folders:

    menu <name shown in the menu> folder { … several “prog”/”menu” … }

toolbar

This file defines the programs that appear in the IceWM toolbar. The syntax is exactly the same syntax as the “prog” entries in the menu file above.

preferences and winoptions; change IceWM background image

Do not touch these. At least not before you have visited the IceWM project homepage www.icewm.org and read the documentation there. If you wish to change the IceWM background image, just replace the file

/usr/lib/X11/icewm/peeklogo.xpm

with your new image file.

If you do not have an image in xpm format, Linux offers a great tool for this job:

“convert [-geometry 800x600] <some_image.some_format> peeklogo.xpm”.

The “geometry” option is only needed if the original image has a size other

than 800x600.

Boot Sequence

This chapter explains the Linux boot in general and also points out the unique Kontron modifications.

GRUB

The *GR*and *U*nified *B*ootloader is usually installed to the master boot record (MBR) of the boot medium. It supports several commonly used file systems. It is configured by a text file (/boot/grub/menu.lst) that can even be edited during boot. It finds the packed Linux kernel on the CF, unpacks and loads it.

Linux kernel

The Linux kernel identifies the hardware components and loads the appropriate drivers. Finally, it starts the first process, /sbin/init.

init

The init process reads the configuration file /etc/inittab (p.17). The second line defines the system initialization script (/etc/rc.d/init.d/rc sysinit).

X-server

The X-server is started in virtual terminal 6. The applications started automatically are defined in “.xinitrc”.

Build your own Kernel

The Kontron Linux image does not contain header files or anything that allows to build your own application or a new kernel on the system itself. It is possible, though, to install the Kontron toolchain on a Linux system and use it to build additional components.

Kontron Toolchain

Glibc, the GNU C library, is a free implementation of the standard C library developed by the GNU-project together with the GNU compiler collection. In addition to the functions of the C-Standard it contains a lot of extensions and performance gains. One of design goals of glibc is the portability. So it is available for many platforms. GNU/Linux uses glibc as its standard C library.

GCC is the abbreviation coming from the IT-scene for a collection of applications with the original meaning GNU C Compiler. Nowadays, GCC is more than just a C-compiler, GCC stands for GNU Compiler Collection. This collection contains compilers for C, C++, Java, Objective-C, Fortran, Treelang and Ada.

The toolchain actually consists of a number of components. The main one is the compiler itself gcc, which can be native to the host or a cross-compiler. This is supported by binutils, a set of tools for manipulating binaries, e.g., ar or ld. These components are the minimum you need for compiling the kernel.

This image follows the principles laid down in LFS (Linux from scratch) and BLFS (beyond Linux from scratch) with some ideas from CLFS (cross Linux from scratch) for the design of the cross-compile environment.

A collection of source archives and shell scripts to build the toolchain is available on request. The Kontron toolchain CD contains a user manual that explains the installation and usage of the toolchain in more detail.

Kernel Sources and Patches

The current Kontron Linux image uses kernel version 2.6.31.4. The last number denotes the patch level. If you wish to take a different kernel it is

recommended to stick with version 2.6.31 and use the appropriate patch level. (Kernel sources are available from www.kernel.org.) Otherwise you would probably have to also update some of the other components of the image due to instability or incompatibility.

One area of application where you might want to use a different kernel is real time Linux. The OSADL project (www.osadl.org) provides real time patches for selected Linux kernel releases. There is an OSADL patch for kernel 2.6.31.12 which can be used in the context of the current Kontron Linux image.

Configuration

Before you can build a new kernel you have to create a configuration file for it. A good starting point would be the configuration file on the Kontron image CF (/boot/config-2.6.31.4). Unpack the kernel source archive then change to the Linux source directory.

If necessary, apply the OSADL patch:

```
cp patch-2.6.31.12-rt21.bz2 <Linux dir>/
cd <Linux dir>
tar xf  patch-2.6.31.12-rt21.bz2
patch -Np1 -i  patch-2.6.31.12-rt21
```

Copy the Kontron configuration file if you like:

```
cp /<CF mount point>/boot/config-2.6.31.4 <Linux dir>/.config
```

Start the kernel configuration:

```
make xconfig
```

(Note that xconfig requires Qt.)

Select the drivers and features you need for your kernel then save the result. The script stores the new configuration in a file named “.config”. If you are using the Kontron scripts to build the kernel, you have to copy this file into a safe place:

cp .config ../../config/config-<kernel-version>

Otherwise just “make” the kernel now.

Kernel Build and Deployment

The following steps should be performed within the toolchain chroot environment (read the toolchain user manual for details):

make

On a multiprocessor machine it is possible to speed up the build process by

make -jN,

where N = 2*(number of processors)+1. Here “make” tries to perform the build in N parallel threads. The kernel make files are optimized to support this kind of parallel processing.

Finally, build the modules and install them:

make modules_install

You do not need the debug information in the driver modules:

find /lib/modules/<kernel version>/ -exec strip –strip-debug {} \;

Now leave the chroot environment and copy the created files and directories to the image CF. Maybe it would be a good idea to save the old kernel.

cp <CF dir>/boot/vmlinuz <CF dir>/boot/old_kernel

cp <Linux dir>/arch/i386/boot/bzImage <CF dir>/boot/vmlinuz

cp -a <toolchain dir>/lib/modules/<kernel version> <CF dir>/lib/modules/

sync

Appendix

/etc/inittab

```
id:5:initdefault:
si::sysinit:/etc/rc.d/init.d/rc sysinit
l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
su:S016:once:/sbin/sulogin
1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:45:once:/usr/bin/startx
```

/usr/local/share/icewm/keys

```
key "Alt+Ctrl+t" xterm /bin/bash
key "Alt+Ctrl+F1" chvt "1"
key "Alt+Ctrl+F2" chvt "2"
key "Alt+Ctrl+F3" chvt "3"
key "Alt+Ctrl+F4" chvt "4"
key "Alt+Ctrl+F5" chvt "5"
key "Alt+Ctrl+F6" chvt "6"
key "Alt+Ctrl+F7" chvt "7"
key "Alt+Ctrl+F8" chvt "8"
key "Alt+Ctrl+F9" chvt "9"
key "Alt+Ctrl+b" setXFreeBlankTime
key "Alt+Ctrl+i" setIP
key "Alt+Ctrl+f" firefox
key "Alt+Ctrl+h" setFirefoxHomepage
key "Alt+Ctrl+k" xvkbd
key "Alt+Ctrl+c" caltouch
key "Alt+Ctrl+v" setXvkbd
key "Alt+Ctrl+e" doReboot
key "Alt+Ctrl+s" setVGAResolution
key "Alt+Ctrl+w" setFirefox
key "Alt+Ctrl+u" setBrowserStartupMode
key "Alt+Ctrl+q" xterm -e version
```

```
key "Alt+Ctrl+d" setxkbmap de
key "Alt+Ctrl+n" setxkbmap us
```

/usr/local/share/icewm/menu

```
prog xterm xterm xterm /bin/bash
prog firefox firefox /usr/bin/firefox
prog xvkbd xterm /usr/bin/xvkbd -always-on-top
prog TouchCalib xterm /usr/bin/caltouch

separator
menu Browser_Prefs folder {
    prog setStartupHomepage xterm setFirefoxHomepage
    prog saveBrowserPrefs xterm firefox_saveprefs
    prog restoreBrowserDefaultPrefs xterm firefox_restoreprefs
    prog autoStartBrowser xterm setFirefox
    prog setBrowserStartupMode xterm setBrowserStartupMode
}

menu Network_Prefs folder {
    prog SetupIP xterm setIP
    prog SetupEurocontroller xterm setEurocontroller
}

menu Editors folder {
    prog joe xterm xterm -e /usr/bin/joe
    prog nano xterm xterm -e /usr/bin/nano
}

menu System folder {
    prog SetBlankTime xterm setXFreeBlankTime
    prog xkill xterm xkill
    prog "Mount USB disk" xterm xterm -e /usr/bin/usb_mount -hold
    prog "Free USB disk" xterm xterm -e /usr/bin/usb_umount
    prog "Mount USB CD-ROM" xterm xterm -e /usr/bin/cdr_mount
-hold
    prog "Free USB CD-ROM" xterm xterm -e /usr/bin/cdr_umount
}

menu Other folder {
    prog xcalc xcalc xcalc
    prog xpdf pdf xpdf
}

prog Reboot xterm doReboot
```