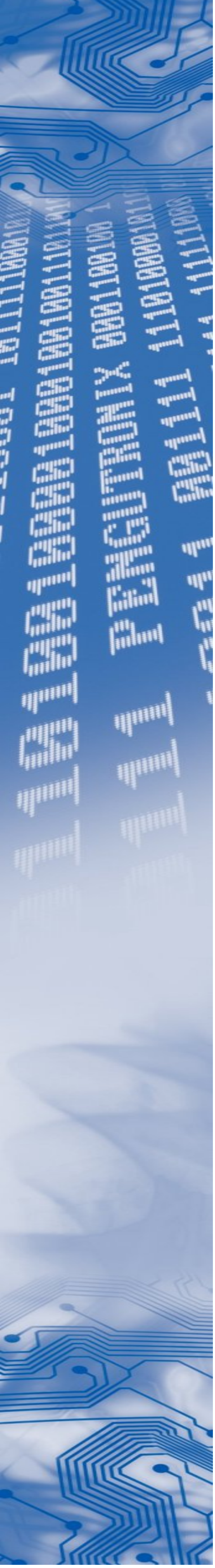


OSELAS.Support
OSELAS.Training
OSELAS.Development
OSELAS.Services

Quickstart Manual
OSELAS.BSP( )
FriendlyARM mini6410

Pengutronix e. K.
Peiner Straße 6–8
31137 Hildesheim

+49 (0)51 21 / 20 69 17 – 0 (Fon)
+49 (0)51 21 / 20 69 17 – 55 55 (Fax)

info@pengutronix.de

Cut Here and Stick on your Monitor

Don't Panic

Contents

Part I

OSELAS Quickstart for FriendlyARM mini6410

1 Getting a working Environment

1.1 Download Software Components

In order to follow this manual, some software archives are needed. There are several possibilities how to get these: either as part of an evaluation board package or by downloading them from the Pengutronix web site.

The central place for OSELAS related documentation is `http://www.oselas.com`. This website provides all required packages and documentation (at least for software components which are available to the public).

To build OSELAS.BSP-Pengutronix-Mini6410-2011.07.0, the following archives have to be available on the development host:

- ptxdist-2011.07.0.tar.bz2
- OSELAS.BSP-Pengutronix-Mini6410-2011.07.0.tar.gz
- OSELAS.Toolchain-2011.03.0.tar.bz2

If they are not available on the development system yet, it is necessary to get them.

1.2 PTXdist Installation

The PTXdist build system can be used to create a root filesystem for embedded Linux devices. In order to start development with PTXdist it is necessary to install the software on the development system.

This chapter provides information about how to install and configure PTXdist on the development host.

1.2.1 Main Parts of PTXdist

The most important software component which is necessary to build an OSELAS.BSP( ) board support package is the `ptxdist` tool. So before starting any work we'll have to install PTXdist on the development host.

PTXdist consists of the following parts:

The `ptxdist` Program: `ptxdist` is installed on the development host during the installation process. `ptxdist` is called to trigger any action, like building a software packet, cleaning up the tree etc. Usually the `ptxdist` program is used in a *workspace* directory, which contains all project relevant files.

A Configuration System: The config system is used to customize a *configuration*, which contains information about which packages have to be built and which options are selected.

Patches: Due to the fact that some upstream packages are not bug free – especially with regard to cross compilation – it is often necessary to patch the original software. PTXdist contains a mechanism to automatically apply patches to packages. The patches are bundled into a separate archive. Nevertheless, they are necessary to build a working system.

Package Descriptions: For each software component there is a “recipe” file, specifying which actions have to be done to prepare and compile the software. Additionally, packages contain their configuration sniplet for the config system.

Toolchains: PTXdist does not come with a pre-built binary toolchain. Nevertheless, PTXdist itself is able to build toolchains, which are provided by the OSELAS.Toolchain() project. More in-deep information about the OSELAS.Toolchain() project can be found here: `http://www.pengutronix.de/oselas/toolchain/index_en.html`

Board Support Package This is an optional component, mostly shipped aside with a piece of hardware. There are various BSP available, some are generic, some are intended for a specific hardware.

1.2.2 Extracting the Sources

To install PTXdist, at least two archives have to be extracted:

ptxdist-2011.07.0.tar.bz2 The PTXdist software itself

ptxdist-2011.07.0-projects.tgz Generic projects (optional), can be used as a starting point for self-built projects.

The PTXdist packet has to be extracted into some temporary directory in order to be built before the installation, for example the `local/` directory in the user’s home. If this directory does not exist, we have to create it and change into it:

```
$ cd
$ mkdir local
$ cd local
```

Next step is to extract the archive:

```
$ tar -xjf ptxdist-2011.07.0.tar.bz2
```

and if required the generic projects:

```
$ tar -xzf ptxdist-2011.07.0-projects.tgz
```

If everything goes well, we now have a PTXdist-2011.07.0 directory, so we can change into it:

```
$ cd ptxdist-2011.07.0
$ ls -lF
total 509
-rw-r--r--   1 jb user  18361 Jul 12 01:18 COPYING
-rw-r--r--   1 jb user   3914 Jul 12 01:18 CREDITS
-rw-r--r--   1 jb user 115540 Jul 12 01:18 ChangeLog
-rw-r--r--   1 jb user     57 Jul 12 01:18 INSTALL
-rw-r--r--   1 jb user   2531 Jul 12 01:18 Makefile.in
-rw-r--r--   1 jb user   4252 Jul 12 01:18 README
-rw-r--r--   1 jb user  63516 Jul 12 01:18 TODO
-rwxr-xr-x   1 jb user     28 Jul 12 01:18 autogen.sh*
drwxr-xr-x   2 jb user     72 Jul 12 01:18 bin/
drwxr-xr-x  10 jb user    296 Jul 12 01:18 config/
-rwxr-xr-x   1 jb user 212213 Jul 12 10:20 configure*
-rw-r--r--   1 jb user  12515 Jul 12 01:18 configure.ac
drwxr-xr-x  10 jb user    248 Jul 12 01:18 generic/
drwxr-xr-x 217 jb user   7304 Jul 12 01:18 patches/
```

```
drwxr-xr-x   2 jb user   1240 Jul 12 01:18 platforms/
drwxr-xr-x   4 jb user    112 Jul 12 01:18 plugins/
drwxr-xr-x   6 jb user  53048 Jul 12 01:18 rules/
drwxr-xr-x   8 jb user    912 Jul 12 01:18 scripts/
drwxr-xr-x   2 jb user    512 Jul 12 01:18 tests/
```

1.2.3 Prerequisites

Before PTXdist can be installed it has to be checked if all necessary programs are installed on the development host. The configure script will stop if it discovers that something is missing.

The PTXdist installation is based on GNU autotools, so the first thing to be done now is to configure the packet:

```
$ ./configure
```

This will check your system for required components PTXdist relies on. If all required components are found the output ends with:

```
[...]
checking whether /usr/bin/patch will work... yes

configure: creating ./config.status
config.status: creating Makefile
config.status: creating scripts/ptxdist_version.sh
config.status: creating rules/ptxdist-version.in

ptxdist version 2011.07.0 configured.
Using '/usr/local' for installation prefix.

Report bugs to ptxdist@pengutronix.de
```

Without further arguments PTXdist is configured to be installed into `/usr/local`, which is the standard location for user installed programs. To change the installation path to anything non-standard, we use the `--prefix` argument to the `configure` script. The `--help` option offers more information about what else can be changed for the installation process.

The installation paths are configured in a way that several PTXdist versions can be installed in parallel. So if an old version of PTXdist is already installed there is no need to remove it.

One of the most important tasks for the `configure` script is to find out if all the programs PTXdist depends on are already present on the development host. The script will stop with an error message in case something is missing. If this happens, the missing tools have to be installed from the distribution befor re-running the `configure` script.

When the `configure` script is finished successfully, we can now run

```
$ make
```

All program parts are being compiled, and if there are no errors we can now install PTXdist into it's final location. In order to write to `/usr/local`, this step has to be performed as user *root*:

```
$ sudo make install
[enter password]
[...]
```

If we don't have root access to the machine it is also possible to install PTXdist into some other directory with the `--prefix` option. We need to take care that the `bin/` directory below the new installation dir is added to our `$PATH` environment variable (for example by exporting it in `~/.bashrc`).

The installation is now done, so the temporary folder may now be removed:

```
$ cd ../../
$ rm -fr local
```

1.2.4 Configuring PTXdist

When using PTXdist for the first time, some setup properties have to be configured. Two settings are the most important ones: Where to store the source packages and if a proxy must be used to gain access to the world wide web.

Run PTXdist's setup:

```
$ ptxdist setup
```

Due to PTXdist is working with sources only, it needs various source archives from the world wide web. If these archives are not present on our host, PTXdist starts the `wget` command to download them on demand.

Proxy Setup

To do so, an internet access is required. If this access is managed by a proxy `wget` command must be adviced to use it. PTXdist can be configured to advice the `wget` command automatically: Navigate to entry *Proxies* and enter the required addresses and ports to access the proxy in the form:

`<protocol>://<address>:<port>`

Source Archive Location

Whenever PTXdist downloads source archives it stores these archives in a project local manner. If we are working with more than one project, every project would download its own required archives. To share all source archives between all projects PTXdist can be configured to use only one archive directory for all projects it handles: Navigate to menu entry *Source Directory* and enter the path to the directory where PTXdist should store archives to share between projects.

Generic Project Location

If we already installed the generic projects we should also configure PTXdist to know this location. If we already did so, we can use the command `ptxdist projects` to get a list of available projects and `ptxdist clone` to get a local working copy of a shared generic project.

Navigate to menu entry *Project Searchpath* and enter the path to projects that can be used in such a way. Here we can configure more than one path, each part can be delemited by a colon. For example for PTXdist's generic projects and our own previous projects like this:

`/usr/local/lib/ptxdist-2011.07.0/projects:/office/my_projects/ptxdist`

Leave the menu and store the configuration. PTXdist is now ready for use.

1.3 Toolchains

Before we can start building our first userland we need a cross toolchain. On Linux, toolchains are no monolithic beasts. Most parts of what we need to cross compile code for the embedded target comes from the *GNU Compiler Collection*, `gcc`. The gcc packet includes the compiler frontend, `gcc`, plus several backend tools (cc1, g++, ld etc.) which actually perform the different stages of the compile process. `gcc` does not contain the assembler, so we also need the *GNU Binutils package* which provides lowlevel stuff.

Cross compilers and tools are usually named like the corresponding host tool, but with a prefix – the *GNU target*. For example, the cross compilers for ARM and powerpc may look like

- `arm-softfloat-linux-gnu-gcc`
- `powerpc-unknown-linux-gnu-gcc`

With these compiler frontends we can convert e.g. a C program into binary code for specific machines. So for example if a C program is to be compiled natively, it works like this:

```
$ gcc test.c -o test
```

To build the same binary for the ARM architecture we have to use the cross compiler instead of the native one:

```
$ arm-softfloat-linux-gnu-gcc test.c -o test
```

Also part of what we consider to be the ”toolchain” is the runtime library (libc, dynamic linker). All programs running on the embedded system are linked against the libc, which also offers the interface from user space functions to the kernel.

The compiler and libc are very tightly coupled components: the second stage compiler, which is used to build normal user space code, is being built against the libc itself. For example, if the target does not contain a hardware floating point unit, but the toolchain generates floating point code, it will fail. This is also the case when the toolchain builds code for i686 CPUs, whereas the target is i586.

So in order to make things working consistently it is necessary that the runtime libc is identical with the libc the compiler was built against.

PTXdist doesn’t contain a pre-built binary toolchain. Remember that it’s not a distribution but a development tool. But it can be used to build a toolchain for our target. Building the toolchain usually has only to be done once. It may be a good idea to do that over night, because it may take several hours, depending on the target architecture and development host power.

1.3.1 Using Existing Toolchains

If a toolchain is already installed which is known to be working, the toolchain building step with PTXdist may be omitted.

⚠ The OSELAS.BoardSupport() Packages shipped for PTXdist have been tested with the OSELAS.Toolchains() built with the same PTXdist version. So if an external toolchain is being used which isn’t known to be stable, a target may fail. Note that not all compiler versions and combinations work properly in a cross environment.

Every OSELAS.BoardSupport() Package checks for its OSELAS.Toolchain it’s tested against, so using a different toolchain vendor requires an additional step:

Open the OSELAS.BoardSupport() Package menu with:

```
$ ptxdist platformconfig
```

and navigate to `architecture ---> toolchain` and `check for specific toolchain vendor`. Clear this entry to disable the toolchain vendor check.

Preconditions an external toolchain must meet:

- it shall be built with the configure option `--with-sysroot` pointing to its own C libraries.
- it should not support the *multilib* feature as this may confuse PTXdist which libraries are to select for the root filesystem

If we want to check if our toolchain was built with the `--with-sysroot` option, we just run this simple command:

```
$ mytoolchain-gcc -v 2>&1 | grep with-sysroot
```

If this command **does not** output anything, this toolchain was not built with the `--with-sysroot` option and cannot be used with PTXdist.

1.3.2 Building a Toolchain

PTXdist handles toolchain building as a simple project, like all other projects, too. So we can download the OSELAS.Toolchain bundle and build the required toolchain for the OSELAS.BoardSupport() Package.

Building any toolchain of the OSELAS.Toolchain-2011.03 is tested with PTXdist-2011.03.0. Pengutronix recommends to use this specific PTXdist to build the toolchain. So, it might be essential to install more than one PTXdist revision to build the toolchain and later on the Board Support Package if the latter one is made for a different PTXdist revision.

A PTXdist project generally allows to build into some project defined directory; all OSELAS.Toolchain projects that come with PTXdist are configured to use the standard installation paths mentioned below.

All OSELAS.Toolchain projects install their result into `/opt/OSELAS.Toolchain-2011.03/`.

Usually the `/opt` directory is not world writeable. So in order to build our OSELAS.Toolchain into that directory we need to use a root account to change the permissions. PTXdist detects this case and asks if we want to run `sudo` to do the job for us. Alternatively we can enter:
`mkdir /opt/OSELAS.Toolchain-2011.03`
`chown <username> /opt/OSELAS.Toolchain-2011.03`
`chmod a+rwx /opt/OSELAS.Toolchain-2011.03`.

We recommend to keep this installation path as PTXdist expects the toolchains at `/opt`. Whenever we go to select a platform in a project, PTXdist tries to find the right toolchain from data read from the platform configuration settings and a toolchain at `/opt` that matches to these settings. But that's for our convenience only. If we decide to install the toolchains at a different location, we still can use the *toolchain* parameter to define the toolchain to be used on a per project base.

1.3.3 Building the OSELAS.Toolchain for OSELAS.BSP-Pengutronix-Mini6410-2011.07.0

To compile and install an OSELAS.Toolchain we have to extract the OSELAS.Toolchain archive, change into the new folder, configure the compiler in question and start the build.

The required compiler to build the OSELAS.BSP-Pengutronix-Mini6410-2011.07.0 board support package is

`arm-1136jfs-linux-gnueabi_gcc-4.5.2_glibc-2.13_binutils-2.21_kernel-2.6.36-sanitized`

So the steps to build this toolchain are:

In order to build any of the OSELAS.Toolchains, the host must provide the tool *fakeroot*. Otherwise the message `bash: fakeroot: command not found` will occur and the build stops.

```
$ tar xf OSELAS.Toolchain-2011.03.0.tar.bz2
$ cd OSELAS.Toolchain-2011.03.0
$ ptxdist select ptxconfigs/↵
      arm-1136jfs-linux-gnueabi_gcc-4.5.2_glibc-2.13_binutils-2.21_kernel-2.6.36-sanitized.ptxconfig
$ ptxdist go
```

At this stage we have to go to our boss and tell him that it's probably time to go home for the day. Even on reasonably fast machines the time to build an OSELAS.Toolchain is something like around 30 minutes up to a few hours.

Measured times on different machines:

- Single Pentium 2.5 GHz, 2 GiB RAM: about 2 hours
- Turion ML-34, 2 GiB RAM: about 1 hour 30 minutes
- Dual Athlon 2.1 GHz, 2 GiB RAM: about 1 hour 20 minutes
- Dual Quad-Core-Pentium 1.8 GHz, 8 GiB RAM: about 25 minutes

Another possibility is to read the next chapters of this manual, to find out how to start a new project.

When the OSELAS.Toolchain project build is finished, PTXdist is ready for prime time and we can continue with our first project.

1.3.4 Protecting the Toolchain

All toolchain components are built with regular user permissions. In order to avoid accidential changes in the toolchain, the files should be set to read-only permissions after the installation has finished successfully. It is also possible to set the file ownership to root. This is an important step for reliability, so it is highly recommended.

1.3.5 Building Additional Toolchains

The OSELAS.Toolchain-2011.03.0 bundle comes with various predefined toolchains. Refer the `ptxconfigs/` folder for other definitions. To build additional toolchains we only have to clean our current toolchain project, removing the current `selected_ptxconfig` link and creating a new one.

```
$ ptxdist clean
$ rm selected_ptxconfig
$ ptxdist select ptxconfigs/any_other_toolchain_def.ptxconfig
$ ptxdist go
```

All toolchains will be installed side by side architecture dependent into directory

`/opt/OSELAS.Toolchain-2011.03/architecture_part`.

Different toolchains for the same architecture will be installed side by side version dependent into directory

`/opt/OSELAS.Toolchain-2011.03/architecture_part/version_part`.

2 Building a root filesystem for the mini6410

2.1 Extracting the Board Support Package

In order to work with a PTXdist based project we have to extract the archive first.

```
$ tar -zxf OSELAS.BSP-Pengutronix-Mini6410-2011.07.0.tar.gz
$ cd OSELAS.BSP-Pengutronix-Mini6410-2011.07.0
```

PTXdist is project centric, so now after changing into the new directory we have access to all valid components.

```
total 36
-rw-r--r-- 1 jbe ptx 1053 Jul 12 19:44 Changelog
-rw-r--r-- 1 jbe ptx 2330 May 31 21:15 FAQ
-rw-r--r-- 1 jbe ptx  177 May 31 21:15 README
drwxr-xr-x 3 jbe ptx 4096 Jul 12 19:44 configs
drwxr-xr-x 3 jbe ptx 4096 May 31 21:15 documentation
drwxr-xr-x 3 jbe ptx 4096 Jul 12 19:44 local_src
drwxr-xr-x 3 jbe ptx 4096 Jul 12 19:44 projectroot
drwxr-xr-x 2 jbe ptx 4096 Jun  5 13:58 protocol
drwxr-xr-x 2 jbe ptx 4096 Jul 12 19:44 rules
```

Notes about some of the files and directories listed above:

ChangeLog Here you can read what has changed in this release. Note: This file does not always exist.

documentation If this BSP is one of our OSELAS BSPs, this directory contains the Quickstart you are currenly reading in.

configs A multiplatform BSP contains configurations for more than one target. This directory contains the platform configuration files.

projectroot Contains files and configuration for the target's runtime. A running GNU/Linux system uses many text files for runtime configuration. Most of the time the generic files from the PTXdist installation will fit the needs. But if not, customized files are located in this directory.

rules If something special is required to build the BSP for the target it is intended for, then this directory contains these additional rules.

patches If some special patches are required to build the BSP for this target, then this directory contains these patches on a per package basis.

tests Contains test scripts for automated target setup.

2.2 Selecting a Userland Configuration

First of all we have to select a userland configuration. This step defines what kind of applications will be built for the hardware platform. The OSELAS.BSP-Pengutronix-Mini6410-2011.07.0 comes with a predefined configuration we select in the following step:

```
$ ptxdist select configs/ptxconfig
info: selected ptxconfig:
      'configs/ptxconfig'
```

2.3 Selecting a Hardware Platform

Before we can build this BSP, we need to select one of the possible platforms to build for. In this case we want to build for the mini6410:

```
$ ptxdist platform configs/platform-friendlyarm-mini6410/platformconfig
info: selected platformconfig:
       'configs/platform-friendlyarm-mini6410/platformconfig'
```

Note: If you have installed the OSELAS.Toolchain() at its default location, PTXdist should already have detected the proper toolchain while selecting the platform. In this case it will output:

```
found and using toolchain:
'/opt/OSELAS.Toolchain-2011.03/arm-1136jfs-linux-gnueabi/↵
    gcc-4.5.2-glibc-2.13-binutils-2.21-kernel-2.6.36-sanitized/bin'
```

If it fails you can continue to select the toolchain manually as mentioned in the next section. If this autodetection was successful, you can omit the step of the next section and continue to build the BSP.

2.4 Selecting a Toolchain

If not automatically detected, the last step in selecting various configurations is to select the toolchain to be used to build everything for the target.

```
$ ptxdist toolchain /opt/OSELAS.Toolchain-2011.03/arm-1136jfs-linux-gnueabi/↵
      gcc-4.5.2-glibc-2.13-binutils-2.21-kernel-2.6.36-sanitized/bin
```

2.5 Building the Root Filesystem

Now everything is prepared for PTXdist to compile the BSP. Starting the engines is simply done with:

```
$ ptxdist go
```

PTXdist does now automatically find out from the `selected_ptxconfig` and `selected_platformconfig` files which packages belong to the project and starts compiling their *targetinstall* stages (that one that actually puts the compiled binaries into the root filesystem). While doing this, PTXdist finds out about all the dependencies between the packages and builds them in the correct order.

While the command `ptxdist go` is running we can watch it building all the different stages of a package. In the end the final root filesystem for the target board can be found in the `platform-mini6410/root/` directory and a bunch of **.ipk* packets in the `platform-mini6410/packages/` directory, containing the single applications the root filesystem consists of.

2.6 Building an Image

After we have built a root filesystem, we can make an image, which can be flashed to the target device. To do this call

```
$ ptxdist images
```

PTXdist will then extract the content of priorly created **.ipk* packages to a temporary directory and generate an image out of it. PTXdist supports following image types:

- **hd.img:** contains grub bootloader, kernel and root files in a ext2 partition. Mostly used for x86 target systems.
- **root.jffs2:** root files inside a jffs2 filesystem.
- **uRamdisk:** a barebox/u-boot loadable Ramdisk
- **initrd.gz:** a traditional initrd RAM disk to be used as initrdramfs by the kernel
- **root.ext2:** root files inside a ext2 filesystem.
- **root.squashfs:** root files inside a squashfs filesystem.
- **root.tgz:** root files inside a plain gzip compressed tar ball.
- **root.ubi:** root files inside a ubi volume.

The to be generated image types and addtional options can be defined with

```
$ ptxdist platformconfig
```

Then select the submenu “`image creation options`”. The generated image will be placed into `platform-mini6410/images/`.

Only the content of the **.ipk* packages will be used to generate the image. This means that files which are put manually into the `platform-mini6410/root/` will not be enclosed in the image. If custom files are needed for the target, install them with PTXdist.

3 Document Revisions

2011/07/12	Initial Revision ×

4 Getting help

Below is a list of locations where you can get help in case of trouble. For questions how to do something special with PTXdist or general questions about Linux in the embedded world, try these.

4.1 Mailing Lists

4.1.1 About PTXdist in Particular

This is an English language public mailing list for questions about PTXdist. See

`http://www.pengutronix.de/mailinglists/index_en.html`

how to subscribe to this list. If you want to search through the mailing list archive, visit

`http://www.mail-archive.com/`

and search for the list *ptxdist*. Please note again that this mailing list is just related to the PTXdist as a software. For questions regarding your specific BSP, see the following items.

4.1.2 About Embedded Linux in General

This is a German language public mailing list for general questions about Linux in embedded environments. See

`http://www.pengutronix.de/mailinglists/index_de.html`

how to subscribe to this list. Note: You can also send mails in English.

4.2 News Groups

4.2.1 About Linux in Embedded Environments

This is an English newsgroup for general questions about Linux in embedded environments.

comp.os.linux.embedded

4.2.2 About General Unix/Linux Questions

This is a German newsgroup for general questions about Unix/Linux programming.

de.comp.os.unix.programming

4.3 Chat/IRC

About PTXdist in particular

irc.freenode.net:6667

Create a connection to the **irc.freenode.net:6667** server and enter the chatroom **#ptxdist**. This is an English room to answer questions about PTXdist. Best time to meet somebody there is at European daytime.

4.4 FriendlyARM mini6410 specific Mailing List

oselas@community.pengutronix.de

This is a community mailing list open for everyone for all mini6410's board support package related questions. Refer our page at

`http://www.pengutronix.de/mailinglists/index_en.html`

to subscribe to this mailing list.

Note: Please be aware that we cannot answer hardware only related questions on this list.

4.5 Commercial Support

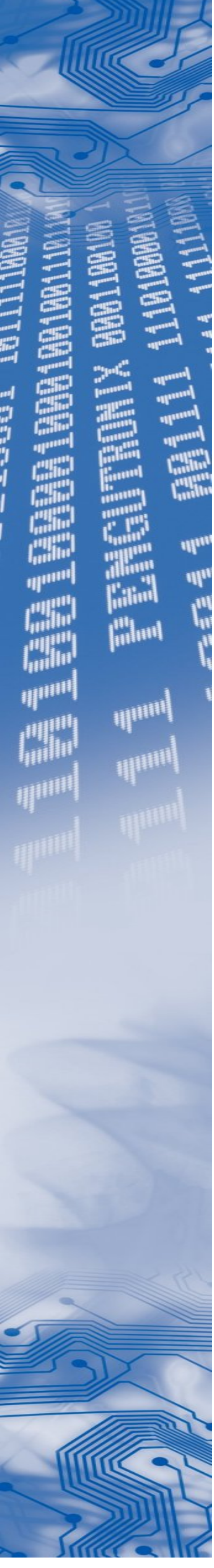
You can order immediate support through customer specific mailing lists, by telephone or also on site. Ask our sales representative for a price quotation for your special requirements.

Contact us at:

Pengutronix
Peiner Str. 6-8
31137 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 55 55

or by electronic mail:

sales@pengutronix.de



This is a Pengutronix Quickstart Manual

Pengutronix e.K.
Peiner Str. 6-8
31137 Hildesheim
Germany
Phone: +49 - 51 21 / 20 69 17 - 0
Fax: +49 - 51 21 / 20 69 17 - 55 55