SOFTWARE
RELEASE NOTES

Kontron VME/VPX Fedora 25 Remix Release Notes

ID 17228 for VX6060, VX3030, VM6050, VX3035, VX3040, VX6080, VM6052, VM6054, VX305x, VX6090, VX6064, VX6058 Boards

SD.DT.G53-2e - September 2017

POSSIBILITIES START HERE kontron

Kontron VME/VPX Fedora 25 Remix Release Notes

Disclaimer

Kontron would like to point out that the information contained in this manual may be subject to alteration, particularly as a result of the constant upgrading of Kontron products. This document does not entail any guarantee on the part of Kontron with respect to technical processes described in the manual or any product characteristics set out in the manual. Kontron assumes no responsibility or liability for the use of the described product(s), conveys no license or title under any patent, copyright or mask work rights to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement unless otherwise specified. Applications that are described in this manual are for illustration purposes only. Kontron makes no representation or warranty that such application will be suitable for the specified use without further testing or modification. Kontron expressly informs the user that this manual only contains a general description of processes and instructions which may not be applicable in every individual case. In cases of doubt, please contact Kontron.

Lise-Meitner-Str. 3-5

86156 Augsburg

Germany

www.kontron.com

REVISION HISTORY

PUBLICATION TITLE:	Kontron VME/VPX Fedora 25 Remix Release Notes	
DOC. ID:	SD.DT.G53-2e	
Revision	Brief Description of Changes	Date of Issue
2e	New Fedora 25 Release for VME/VPX Boards	09-2017
1e	Updated sections: - Chapter 2 - Release Content - Chapter 4 - Required Configuration - Chapter 5 - Using the Live Image and Installing it on a Hard Disk News sections: - Chapter 7 - VxFabric - Chapter 8 - SM750 Graphical Chipset Support	02-2017
0e	Initial Issue	07-2016

Customer Support

Find Kontron contacts by visiting: http://www.kontron.com/support.

Customer Service

As a trusted technology innovator and global solutions provider, Kontron extends its embedded market strengths into a services portfolio allowing companies to break the barriers of traditional product lifecycles. Proven product expertise coupled with collaborative and highly-experienced support enables Kontron to provide exceptional peace of mind to build and maintain successful products.

For more details on Kontron's service offerings such as: enhanced repair services, extended warranty, Kontron training academy, and more visit http://www.kontron.com/support-and-services/services.

Customer Comments

If you have any difficulties using this manual, discover an error, or just want to provide some feedback, contact Kontron support. Detail any errors you find. We will correct the errors or problems as soon as possible and post the revised manual on our website.

SYMBOLS

The following symbols may be used in this manual:

⚠DANGER	DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.
⚠WARNING	WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.
⚠CAUTION	CAUTION indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	NOTICE indicates a property damage message.
	**Electric Shock!** This symbol and title warn of hazards due to electrical shocks (> 60 V) when touching products or parts of them. Failure to observe the precautions indicated and/or prescribed by the law may endanger your life/health and/or result in damage to your material.
	**ESD Sensitive Device!** This symbol and title inform that the electronic boards and their components are sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.
	**HOT Surface!** Do NOT touch! Allow to cool before servicing.
	**Laser!** This symbol inform of the risk of exposure to laser beam from an electrical device. Eye protection per manufacturer notice shall review before servicing.
	This symbol indicates general information about the product and the user manual. This symbol also indicates detail information about the specific product configuration.
	This symbol precedes helpful hints and tips for daily use.

FOR YOUR SAFETY

Your new Kontron product was developed and tested carefully to provide all features necessary to ensure its compliance with electrical safety requirements. It was also designed for a long fault-free life. However, the life expectancy of your product can be drastically reduced by improper treatment during unpacking and installation. Therefore, in the interest of your own safety and of the correct operation of your new Kontron product, you are requested to conform with the following guidelines.

High Voltage Safety Instructions

As a precaution and in case of danger, the power connector must be easily accessible. The power connector is the product's main disconnect device.

⚠CAUTION

Warning!
All operations on this device must be carried out by sufficiently skilled personnel only.

⚠CAUTION

Caution, Electric Shock!
Before installing a non hot-swappable Kontron product into a system always ensure that your mains power is switched off. This also applies to the installation of piggybacks. Serious electrical shock hazards can exist during all installation, repair, and maintenance operations on this product. Therefore, always unplug the power cable and any other cables which provide external voltages before performing any work on this product.
Earth ground connection to vehicle's chassis or a central grounding point shall remain connected. The earth ground cable shall be the last cable to be disconnected or the first cable to be connected when performing installation or removal procedures on this product.

Special Handling and Unpacking Instructions

ESD Sensitive Device!
Electronic boards and their components are sensitive to static electricity. Therefore, care must be taken during all handling operations and inspections of this product, in order to ensure product integrity at all times

Do not handle this product out of its protective enclosure while it is not used for operational purposes unless it is otherwise protected.

Whenever possible, unpack or pack this product only at EOS/ESD safe work stations. Where a safe work station is not guaranteed, it is important for the user to be electrically discharged before touching the product with his/her hands or tools. This is most easily done by touching a metal part of your system housing.

It is particularly important to observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings etc. If the product contains batteries for RTC or memory backup, ensure that the product is not placed on conductive surfaces, including anti-static plastics or sponges. They can cause short circuits and damage the batteries or conductive circuits on the product.

GENERAL INSTRUCTIONS ON USAGE

In order to maintain Kontron's product warranty, this product must not be altered or modified in any way. Changes or modifications to the product, that are not explicitly approved by Kontron and described in this manual or received from Kontron's Technical Support as a special handling instruction, will void your warranty.

This product should only be installed in or connected to systems that fulfill all necessary technical and specific environmental requirements. This also applies to the operational temperature range of the specific board version, that must not be exceeded. If batteries are present, their temperature restrictions must be taken into account.

In performing all necessary installation and application operations, only follow the instructions supplied by the present manual.

Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the product then re-pack it in the same manner as it was delivered.

Special care is necessary when handling or unpacking the product. See Special Handling and Unpacking Instruction.

ENVIRONMENTAL PROTECTION STATEMENT

This product has been manufactured to satisfy environmental protection requirements where possible. Many of the components used (structural parts, printed circuit boards, connectors, batteries, etc.) are capable of being recycled.

Final disposition of this product after its service life must be accomplished in accordance with applicable country, state, or local laws or regulations.

Environmental protection is a high priority with Kontron.
Kontron follows the DEEE/WEEE directive.
You are encouraged to return our products for proper disposal.

The Waste Electrical and Electronic Equipment (WEEE) Directive aims to:

- Reduce waste arising from electrical and electronic equipment (EEE)
- Make producers of EEE responsible for the environmental impact of their products, especially when they become waste
- Encourage separate collection and subsequent treatment, reuse, recovery, recycling and sound environmental disposal of EEE

Improve the environmental performance of all those involved during the lifecycle of EEE

Table Of Contents

1 / Overview

Linux, the Open Source Operating System is now taking a significant share of the OS market in Defense and Aerospace, after having taken ground initially in the enterprise server sector.

The goal of this document is to help you to use the Kontron VME/VPX Fedora 25 Remix live image and optionally install it on a storage device.

2 / Release Content

The release consists of a unique live Image file: **KVX-Live_FC25_ID17XXX.iso**

This live image is based on standard Fedora 25 live Xfce (**Fedora-Xfce-Live-x86_64-25-1.3.iso**), but remixed with additional Fedora development packages and Kontron specific packages for VME/VPX boards.

As with a standard live image, it can be used in a "live" way but can also be installed on a storage device to have an installed system with the same set of packages than the live image.

The Kontron specific packages are:

- **kvx_bsp**: the "Kontron VME VPX Board Support Package" that contains:
  - **CPLD** related drivers:
    - **cpld**: core platform driver
    - **cpld_i2c**: subdriver for I2C bus (local /dev/i2c-22, and backplane /dev/i2c-23 and /dev/i2c-24)
    - **cpld_leds**: subdriver for front panel LEDs
    - **cpld_gpio**: subdriver for GPIOs
    - **cpld_wdt**: subdriver for watchdog timer
  - **Services**:
    - **kvx_bsp**: for BSP configuration at boot time
    - **rtc2**: for the additional low power RTC device
  - **kvx_board**: to monitor/manage other boards in the chassis through backplane I2C bus
  - **sensors**: to get the temperature, voltage, or current from all the available sensors (CPU cores and additional sensors)
  - **vpdtool**: to get VPD inventory info (Order code, E.C. level, features, ...)
  - **cpldtool**: to get/set the configuration managed by the CPLD
  - **sysvartool**: to get the status of the PBIT (Power-On Built-In Tests)
  - **cpld_update**: to update the CPLD using an JEDEC image file
  - **fmram**: to read/write the FMRAM device.
  - **several debug tools**: i2c_transfer (low level raw I2C tool), io (to access memory), port (to access I/O ports)
  - **Documentation**: man pages

- **vmetoolkit**: the VME drivers and utilities for VME boards (unused on VPX boards):
  - **almadriver**: the Alma PCI to VME bridge driver
  - **almavmechan**: the utility to manage ingoing and outgoing VME channels
  - **several test tools** with their source files as examples: Dma / Dma2 / MasterSgl, GEN_intr / RECV_intr, mbm3k, probeVmeBoard
  - **Documentation**: man pages

- **sm750module**: the SM750 Silicon Motion graphics chipset modules for the Xorg server and framebuffer. The official ones from SiliconMotion, but repackaged as RPM with additional files and rebuilt from sources by Kontron (only used on Xeon-D architecture: VX305x, VX6090, VX6064, VX6058)
  - **siliconmotion_drv.so**: the core Xorg module
  - **smiddk750_drv.so**: the Xorg module for the SM750 chipset (other .so modules unused)
  - **lynxfb.ko**: framebuffer driver (kernel module)
  - **Service**: sm750 service to configure Xorg for the SM750 chip
  - **Documentation**: man page (siliconmotion)

On a running system, the version of each package can be displayed with:

```
# rpm -q <packagename>
```

and the changelog with

```
# rpm -q --changelog <packagename>
```

If these packages are rebuilt and installed on a Linux distribution without RPM support, their version can be found in the following text files:

```
/etc/kvx_bsp_release
/etc/vmetoolkit_release
/etc/sm750module_release
```

In any case the version of the KVX release can be displayed with:

```
cat /etc/fedora-release
```

which gives the following output (example):

```
KVX ID17178 Fedora Remix 25 (Generic)
```

meaning that the release ID is 17178 (built the 178th day of 2017), based on a Generic Fedora 25, but Remixed with Kontron packages.

3 / Associated Documentation

A copy of this document is available on the live image under **/opt/kvx/release-notes/**, however it may not be fully up-to-date due to release packaging schedule. With a graphics configuration; the document can be open with the "**KVX Release Notes**" icon on liveuser desktop, or from the applications menu (under both **System** and **Documentation** submenus).

Kontron Documentation

Hardware

- VX3030 User's Guide CA.DT.A87
- VX3030 Hardware Release Notes CA.DT.A88
- VX3035 User's Guide CA.DT.A95
- VX3035 Hardware Release Notes CA.DT.A96
- VX304x User's Guide CA.DT.A98
- VX304x Hardware Release Notes CA.DT.A99
- VX305x 3U VPX Computing Node User's Guide CA.DT.B25
- VX305x Hardware Release Notes CA.DT.B26
- VX6058 User's Guide CA.DT.B36
- VX6060 User's Guide CA.DT.A76
- VX6060 Hardware Release Notes CA.DT.A77
- VX6064 User's Guide . . . . . Not available
- VX6080 User's Guide CA.DT.B14
- VX6080 Hardware Release Notes CA.DT.B18
- VX6090 User's Guide CA.DT.B20
- VM6050 User's Guide CA.DT.A93
- VM6050 Hardware Release Notes CA.DT.A94
- VM6052/VM6054 User's Guide CA.DT.B19
- VM6052/VM6054 Release Notes CA.DT.B17

Firmware

- VX3030 BIOS User Manual SD.DT.F81
- VX3030 PBIT User's Guide SD.DT.F94
- VX3035 BIOS User Manual SD.DT.F97
- VX3035 PBIT User's Guide SD.DT.G08
- VX304x BIOS User Manual SD.DT.F96
- VX304x PBIT User's Guide SD.DT.G14
- VX305xv BIOS User Manual SD.DT.G50
- VX305x PBIT User's Guide SD.DT.G51
- VX6060 BIOS User Manual SD.DT.F69
- VX6060 PBIT User's Guide SD.DT.F78
- VX6090 BIOS User Manual SD.DT.G66
- VX6090 PBIT User's Guide SD.DT.G65
- VM6050 BIOS User Manual SD.DT.F89
- VM6050 PBIT User's Guide SD.DT.F88

- VM6052/VM6054 BIOS User Manual SD.DT.G34
- VM6052/VM6054 PBIT User's Guide SD.DT.G35

Fedora Documentation

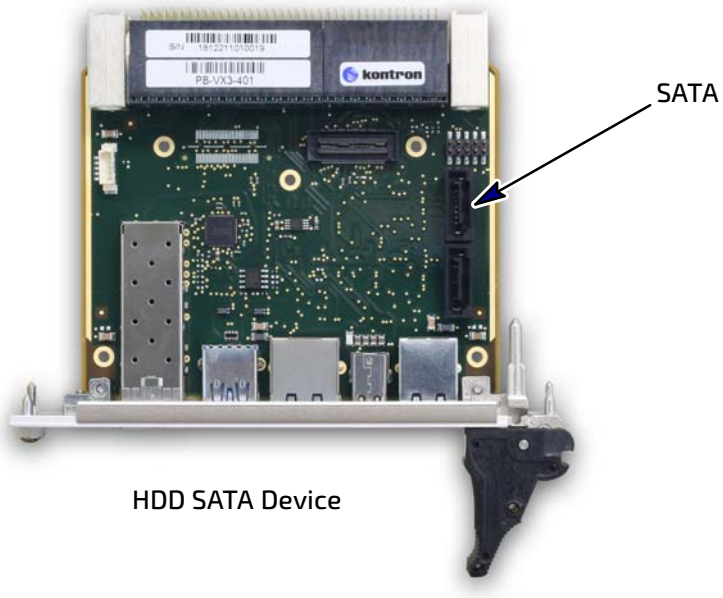
- **Documentation available at** http://fedoraproject.org/

4 / Required Configuration

4.1 Hardware Requirements

- A Kontron VME or VPX board: VX3030, VX3035, VX3040, VX305x, VX6058, VX6060, VX6064, VX6080, VX6090, VM6050, VM6052 or VM6054.
- A storage device to optionally install the live image. Depending on the board type: a SATA disk, SSD module, onboard storage, ...

  On VX305x: a SATA disk connected to the SATA connectors available on VX305x-RTM board, or an optional onboard SSD disk (present on bottom or top M.2 slot).



- A graphics device, if not already provided onboard, to optionally use or install the live image with a graphics configuration.

  Alernatively the serial console can be used instead, and VNC can also be used during the installation to get more installation option (custom partitionning)

  On VX305x, a specific order code and a specific graphic module are required: VX3058SA881151000 or VX3052SA88015100 and MODP-GM2-SM750 . The module provides an HDMI port.

4.2 Firmware Requirements

The version of the BIOS firmware must be at least:

- On VX305x: 16016

This version is displayed in the BIOS Setup.

5 / Using and Installing the Live Image

5.1 Live Image Overview

Even if the image may be burnt on a media DVD, it is more recommended to use a USB stick (thanks to the hybrid format of the image):

- On Linux PC:

```
# umount /dev/sd<your_device_id> # make sure not mounted
# dd if=<your_image.iso> of=/dev/sd<your_device_id> bs=1M
```

- On Windows PC:

  We suggest to download and use "**win32diskimager**"

Writing to the USB stick destroys all data already present on it !

Once booted from the resulting live media, the system is ready to use with all available board features.

To boot from the live media, it is necessary to enter the BIOS boot menu or setup to select the live media as boto device.

As with standard Live images:

- There is an autologin as user "**liveuser**", and no password for user "**root**"
- When burning a USB stick with the methods above, the USB stick is non-persistent which means that every change made to the OS is lost at reboot.
- There is a non-persistent read/write overlay to store the modifications, but the OS freezes without any notification when this overlay is full.
- The "**df**" command gives wrong values on the remaining space, and the only way to know (in 512-byte blocks) is to run:

```
# dmsetup status live-rw
```

A persistent USB stick can be made using this live image and a tool such as "**livecd-iso-to-disk**" or "**Live USB Creator**".

However such a stick has its free space decreasing at each boot, or when changing some files, and removing some files does not restore the related amount of space.

The remaining free space has to be checked with the command above. If running out of free space, the system hangs and booting from the stick is no more possible.

A persistent stick is handy, but is much less robust to a brutal power-off or reset than a non-persistent one.

Running from a live media is suitable for evaluation, but it is recommended to run the system from a storage device on which the software has been installed. The installation can be done from the live media.

5.2 Kontron Packages Sources and Binaries

When running from the live image or from a storage media installed from this image, the Kontron specific packages and files can be found under **/opt/kvx**+:

- For each package: a binary RPM, source SRPM, and source tarball (**.tar.gz**)
- The kickstart file used to build the live image using livecd-creator (can be used to rebuild custom live images)
- and a README.txt for more information

To access these packages and files in the live image without booting it, another system running Linux is required:

- Mount the image file:

  If the available image is a file, use the following command:

  ```
  [root@localhost ~]# mount -o loop KVX-Live_FC25_ID17XXX.iso /mnt2
  ```

  or if the image is available on a USB stick on **/dev/sdb**:

  ```
  [root@localhost ~]# mount /dev/sdb1 /mnt2
  ```

- Then:

  ```
  [root@localhost ~]# mount -o loop /mnt2/LiveOS/squashfs.img /mnt3
  [root@localhost ~]# mount -o loop /mnt3/LiveOS/ext3fs.img /mnt4
  ```

- The files are under **/mnt4/opt/kvx** and can be copied elsewhere
- When done, unmount the image:

  ```
  [root@localhost ~]# umount /mnt4 /mnt3 /mnt2
  ```

5.3 Installation on a Storage Device

The live image offers the possibility to install the software on a storage device.

A system installed from a live media has the same behavior as if the installation was made from an installation media. The only difference is that no package selection is required: all the packages present on the live media are installed, including the Kontron specific software.

If needed, some additional software can be installed later using the standard package installation tools (rpm, yum, dnf, ...).

5.3.1 Graphics Install

- Boot the system from the live media (use BIOS boot menu or setup to select the boot device).
- When the GRUB menu appears, move the cursor up to the first entry (live boot) and press "**e**" to edit the boot parameters.
- Move the cursor down to the "**linuxefi**" line, and remove the "**console=ttyS0,115200**" (mandatory to have the installer running from the graphics desktop, otherwise the installer does not show up).
- Then type <Ctrl>-<X> to boot.
- A graphics desktop should appear at the end of the boot.
- Double click on the Desktop Icon "**Install to Hard Drive**", and wait for the window to open.

  This may take up to a minute. Do not click again on the icon until the window installation pop-up appears.

- Select the language to be used during the installation process. This also selects the language of the system once installed to the hard drive, and the default time zone.
- Then click on "**Continue**".
- Now click on "**Installation Destination**" (or equivalent in the selected language).
- Then select the target disk by clicking on its name to have the check box on it.

  Make sure not to select the live media from which you have booted as destination disk.
- Then click just "**Done**" for a default partitionning, or set a specific custom partitionning (for example a minimal sized system with standard partitions) as explained below:
  - Select "**I will configure partitioning**".
  - Click on "**Done**".
  - Select "**Manual partitioning**", then select "**Standard Partition**"
  - Then click on "**+**" to add each partition in this order:

```
Mount point        Size
/boot/efi          128M
/boot              256M
/                  not specified (field left empty for maximum size)
```

  - Then click **DONE** and be patient (it takes some seconds).
  - With the partitioning above, a warning may be displayed because of the missing swap, so click once again on **DONE** to ignore.

- Optionally click on "**TIME & DATE**" and/or "**NETWORK & HOST NAME**".
- Then after all that, click on "**Begin Installation**".

Advertisements at the bottom of the page during the installation are provided by Fedora, not Kontron.

- Click on "**ROOT PASSWORD**" to set the root password, then click **DONE**. In case of warning because of a weak password, click again if you really want to use it.
- Optionally click on "**USER CREATION**" to create an additional user.
- At the end of installtion, click on "**Quit**".
- Then shutdown the system (now or later), power-off, remove the USB stick, and power-on.
- The system should automatically boot from the installed storage device (the BIOS boot list is updated by the installer that sets a "**Fedora**" entry first in the list).
- At first system login, the desktop panel setting popup choice appears to select between "**One Empty Panel**" or "**Use default config**".

  To get a functional default desktop, select "**Use default config**".

In case the BIOS fails to boot from it, enter the BIOS setup and set the right storage device on top of the boot list, then save and exit.

5.3.2 Install without Graphics

Without graphics, booting the live image from a DVD (or from a USB stick in BIOS legacy mode) is not recommended: this makes the BIOS load ISOLINUX instead of GRUB, but as its redirection to the serial line does not work, no interaction with the operator is possible. The only thing to do is to wait for the board to boot automatically after several seconds. The recommended method is to boot from a USB stick in UEFI mode to get a functional GRUB menu.

- Use a serial console connected to the front (first) serial line.

  Settings are 115200 bauds, 8 bits, no parity, 1 stop bit, no flow control.

  This console will provide access to the BIOS setup, GRUB menu, OS console, login, and shell prompt.
- Boot the system from the live media (use BIOS boot menu or setup to select the boot device)

Depending on the board type, the BIOS may need to be configured with the serial redirection enabled to get the BIOS on the serial line.

- When the GRUB menu appears, move the cursor up to the first entry (**`live boot`**) and press "**`e`**" to edit the boot parameters.
- Move the cursor down to the "**`linuxefi`**" line, and remove the "**`rhgb`**" and "**`quiet`**" (not mandatory but gives a more verbose boot that may be useful).
- Then type <Ctrl>-<X> to boot.
- A liveuser shell prompt should appear at the end of the boot.
- Become root with "**`su -`**" (no password).
- Then run the installer

```
[root@localhost-live ~]# liveinst
```

- After a few seconds, the installer asks to select between "**`Text mode`**" (using the serial line console) or "**`VNC`**".

  `Text mode` provides a limited set of installation options (no custom partitioning), however it is simple to use.

  `VNC` mode requires an additional step to configure a network interface, and a VNC client on another machine connected to the network.
- In any case, the installation process has the same steps than for a graphics install. Just follow the instructions that are displayed.

5.3.3 Updating Kontron Packages after Install

These packages contain some kernel modules that may have been embedded in the **`initramfs`** image the last time it was built.

If one of these packages has to be updated, follow the instructions below after the update:

Check if one of these modules are in the current **`initramfs`**:

```
# lsinitrd | egrep 'cpld|nct7802y|pcf8563_kvx|almavme|lynxfb'
```

If this is the case, rebuild the **`initramfs`** without these drivers (should not be needed there), or rebuild it with the updated ones with:

```
# dracut --force
```

5.4 Installing the Kontron Packages on a Different Linux Version

If the Linux version supplied with the live image is not suitable, it is possible to install the Kontron packages on the distribution of your choice.

Depending on your kernel version or OS, some modifications may be needed in the source code before rebuilding the packages.

5.4.1 Installation of the Linux Distribution

The installation should behave like on any other computer, following the installation instructions of the distribution.

For Fedora, other installation media can be downloaded at: https://getfedora.org

5.4.2 Prepare to rebuild the Kontron Packages

First, install the following packages to the Linux distribution to be able to rebuild the Kontron packages:

```
tar
gcc
nasm
kernel-devel
kernel-headers
rpm-build (for RPM based distributions)
systemd-devel
autoconf
automake
libtool
pciutils
pciutils-devel
xorg-x11-util-macros
xorg-x11-server-devel
libicns-utils
```

When installing kernel-devel and kernel-headers, make sure to install the same version as the running kernel version (given by "`uname -r`"). For that, it is recommended to update the kernel package to the latest (same) version and reboot the system.

To get the sources of the Kontron packages, see section 5.2 "Kontron Packages Sources and Binaries" above.

5.4.3 Build and Install the Kontron Packages

This will build and install the packages for the current running kernel.

- If the kernel is updated after this installation, the packages must be rebuilt and reinstalled.
- Building can be done as a regular user, but installing requires root privileges.
- To build for a kernel other than the running one, but for which the kernel-devel package is installed, see the README file in the sources of the packages.

On a distribution using RPM:

- Build an SRPM (optional):

```
$ rpmbuild -ts package.tar.gz
```

- Build an RPM:

```
$ rpmbuild -tb package.tar.gz
```

- Then install the RPM:

```
# rpm -ivh /path_to_rpm/package*.rpm
```

However, If some changes need to be done to the sources before building:

- Extract the tarball:

```
$ tar xvzf package.tar.gz
$ cd package
```

- Make the changes.
- Build SRPM and RPM:

```
$ make rpm
```

- Install the RPM:

```
# rpm -ivh /path_to_rpm/package*.rpm
```

For distributions not using RPM:

- Extract the tarball:

```
$ tar xvzf package.tar.gz
$ cd package
```

- Build:

```
$ make
```

- Install:

```
# make install
```

Then reboot after installation.

5.5 Build your Own Customized Live Image

To build or customize an up to date live image (example for Fedora 25):

```
  # cd /opt/kvx
  # dnf install livecd-tools
  # edit kvx_livecd.ks as needed to add some packages or customize the image using a %post script
  # livecd-creator --verbose --config=kvx_livecd.ks --fslabel=KVX-Live_FCxx_IDxxxxx --releasever=25
--tmpdir=./livecd/tmp  --cache=./livecd/yum-cache
```

Then you can burn it to a USB stick using "**dd**" (destructive method), or with:

```
  # livecd-iso-to-disk --format --efi --reset-mbr KVX-Live_FCxx_IDxxxxx.iso /dev/sdX
```

or for a persistent image (with read/write overlay):

```
  # livecd-iso-to-disk --format --efi --reset-mbr --overlay-size-mb xxxx KVX-Live_FCxx_
IDxxxxx.iso /dev/sdX
```

The kickstart file **kvx_livecd.ks** is the one used to build the live image and is suitable for Fedora 25. If building a live image for another release, it is recommended to download the standard kickstart files for that release (package is **fedora-kickstarts** for Fedora), then use the right kickstart file (depending on the desired distribution flavour), then apply the KVX specific changes that are marked with "**KVX**" in **kvx_livecd.ks** .

Also in the command above, specify the right release number at the argument "**releasever**". This release may be different from the one running on the development host, however they should not be too different to avoid incompatibilities between the tools running on the host and the release to build.

6 / System Configuration

Information related to some specific configuration items of the Fedora system are detailed In this chapter.

6.1 Network

Network configuration is done with the NetworkManager service, with DHCP as default configuration for all interfaces.

By default, the **kvx_bsp** package/service assigns specific names for the onboard ethernet interfaces: "**em0**" (front), "**em1**", "**em2**", ... This is done with a **udev** rule (**/etc/udev/rules.d/75-kvx_bsp.rules**)

To disable this "**emX**" naming for onboard interfaces and return to the default OS naming, add the following kernel parameter to the kernel command line:

```
kifnames=0
```

The default OS naming can also be disabled to go back to the legacy "**ethX**" names with **X** that is incremented for each onboard or addon interface, according to the PCI probe order. For this the following kernel parameters must be added to the kernel command line:

```
net.ifnames=0 biosdevname=0
```

See **/usr/share/doc/kvx_bsp/README** for more info

In any case, assigning a specific network configuration to a specific interface may require some attention on the following points:

- The network configuration should not be bound to the MAC address, otherwise if the installed disk is moved from a system to another one (needed for maintenance) the interface no more works without manual reconfiguration.
- The network configuration should be bound to the device name (enX, ethX, ...) instead.

So make sure to do the right choices when using a graphical interface to configure the network, and then have a look to the **ifcfg-*** files in **/etc/sysconfig/network-scripts/** and fix them if needed.

Here is an example using Fedora 23 and the Network Manager, with a static IP configuration:

```
[root@sls13 ~]# cat /etc/sysconfig/network-scripts/ifcfg-myconfig
#HWADDR=00:00:DE:52:CB:46
TYPE=Ethernet
BOOTPROTO=none
IPADDR=172.20.144.93
PREFIX=24
GATEWAY=172.20.144.46
DNS1=172.20.144.3
DOMAIN=ariane.local
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
#IPV6INIT=yes
#IPV6_AUTOCONF=yes
#IPV6_DEFROUTE=yes
#IPV6_PEERDNS=yes
#IPV6_PEERROUTES=yes
#IPV6_FAILURE_FATAL=no
NAME="Front Ethernet"
DEVICE=em0
UUID=c165c034-0499-4543-b829-1d1c4c78bdc0
ONBOOT=yes
[root@sls13 ~]#
```

First, the name of the **ifcfg-*** file is not important because all **ifcfg-*** are parsed by the Network Manager regardless their name (here **ifcfg-myconfig**).

The **HWADDR** entry has been commented out so that we are no more bound to this MAC address and can change the board without problem.

The **NAME** entry is the human readable name of the interface as displayed by the Natwork Manager and can be set to any string.

The **DEVICE** entry is the one that bind this configuration to the interface, and only this one. This is the most important one.

Depending on the local network configuration, if some problems occur when resolving fully qualified domain names (host.-domain), it may be necessary to edit **/etc/nsswitch** and replace:

```
hosts:      files mdns4_minimal [NOTFOUND=return] dns myhostname
```

with

```
hosts:      files dns
```

6.2 Boot Parameters

Some boards require some specific kernel parameters to work properly.

Every time the board type is changed, the **kvx_bsp** package/service automatically updates the **grub.cfg file** (**/boot/efi/EFI/fedora/grub.cfg**) so that a single additional reboot is required to have a fully functional system.

All these updates (if any) are logged and can be displayed using **journalctl**:

```
# journalctl -e _SYSTEMD_UNIT=kvx_bsp.service
```

or

```
# journalctl -e -u kvx_bsp
```

See **/usr/share/doc/kvx_bsp/README** for more info.

As the live image is a non persistent media on which each change is lost at reboot, it is necessary to the the right kernel parameters set by default to have all the features working properly. For this, the live image uses a default set of parameters that are suitable for all supported boards, even if not optimal on a specific board. If needed, these parameters can be edited under **GRUB** before booting.

6.3 OS Runlevel

Runlevels are no more managed by **/etc/inittab but by systemd**.

Regardless if the console is redirected to the serial line or not, the boot can be set to:

- Multi-user.target: runlevel 3 (without graphics login)
- Graphical.target: runlevel 5 (with graphics login enabled)

To get the current runlevel:

```
# systemctl get-default
```

To change the runlevel:

```
# systemctl set-default multi-user.target
```

or

```
# systemctl set-default graphical.target
```

To force a runlevel 5 boot, it is also possible to add "**5**" to the kernel command line.

6.4 Firewall

If the firewall must be disabled but has been enabled during the installation:

- To disable at next boot

```
systemctl disable firewall
```

- To disable right now

```
systemctl stop firewall
```

6.5 SELinux

SELinux stands for Security-Enhanced Linux. The Security-Enhanced Linux kernel enforces mandatory access control policies that confine user programs and system servers to the minimum amount of privilege they require to do their jobs.

If you experience some trouble running some services or have some permission issues, try to set the System Default Policy to Permissive instead of Enforcing by running the **system-config-selinux** tool, or from command line doing as follows:

- Disable on boot by editing **/etc/selinux/config** to set **SELINUX=permissive** instead of **SELINUX=enforcing**
- Disable now: **setenforce 0**

6.6 RTC Date Configuration

Depending on the board type, an additional ultra low power Real Time Clock (RTC2) may be used in addition to the PCH RTC to increase RTC battery lifetime.

The **kvx_bsp** package provides an **rtc2** service that enables this RTC (a new **/dev/rtc1** device is created) and set it to the system time on shutdown or reboot (otherwise any date update under OS is lost).

The related log messages can be displayed with:

```
# journalctl -e _SYSTEMD_UNIT=rtc2.service
```

or

```
# journalctl -e -u rtc2
```

See **/usr/share/doc/kvx_bsp/README** for more info.

6.7 RS485 Transmit Enable

On VM6050, VM6052, VM6054, and VX3040, the serial transmit buffer can be enabled/disable to allow RS485 operation.

This can be done with the following commands:

```
# cpldtool -f 7 SERIAL1_TXEN 1  # transmit on
# cpldtool -f 7 SERIAL1_TXEN 0  # transmit off
```

or

```
# cpldtool -f 7 SERIAL2_TXEN 1  # transmit on
# cpldtool -f 7 SERIAL2_TXEN 0  # transmit off
```

depending on the serial port to use.

On VX6080, VX305x, VX6090, VX6058, VX6064, only one transmit on/off is available under the name: **SERIAL_TXEN**.

6.8 Modules Parameters

Depending on the system configuration, some modules may require some different parameters when loaded, such as the watchdog timeout.

This can be done by adding a file to the **/etc/modprobe.d/** (along with the **kvx_bsp.conf** already managed by the **kvx_bsp** service), however if the modules are embedded in the **initramfs**, this config file should also be present in the **initramfs**.

Another way to do, that also works for **initramfs**, is to add a parameter to the kernel command line with the syntax **<modulename>.<valuename>=<value>**; for example:

```
cpld_wdt.timeout=40
```

7 / VXFabric™

Kontron VXFabric™ is an open infrastructure which implements efficient inter board communication at hardware speed. The architecture is compliant with the OpenVPX standard (VITA 65) which defines two main hardware topologies of the backplane: distributed and centralized topologies.

To get more information about this software, go to the www.kontron.com web site and enter the key word "vxfabric" into the search engine.

8 / Multinodes Diskless

Multinodes Diskless is a powerful set of Python scripts to help the configuration of diskless root filesystems and to boot distant nodes through the Ethernet network (PXE).

This software is not provided in the Live image. Contact Kontron for more information.

9 / KEHM Health Monitoring Demo

The live image provides a demo version of the Kontron open computer health monitoring software.

KEHM implements a data driven framework designed to be extendable beyond the base computer to cover all customized elements (hardware, software, infrastructure) and compute in a single pass a health sensor tree which top level OK, WARNING or ALARM value represents the complete customer solution. The framework is based on logical elements called health sensors, managed by a library.

Computer Health Sensor can take three possible values :

- ALERT (computer behavior can be completely wrong)
- WARNING (computer operates but may fail soon)
- SUCCESS (normal operation)

KEHM comes as a library and sample files. The library computes health sensor values from a customizable health sensors definition file, and provides this health values in the form of an object tree, with the complete solution health at the top, and all health sensors information in the health tree branches and leaves.

This demo version is fully functionnal on VM605x and VX305x with the sample sensor files provided (see files in **/usr/share/kehm**).

A VX6090 or VX6064 board is seen as two independant VX3058 boards.

A VX6058 is seen as a VX3058.

Other boards are currently not supported by this demo.

The **/opt/kvx/kehm/install.sh** script has the necessary steps to install a CBIT system (**kehmcbit**), and a minimal http service (kehmhttp, on port 8000).

This script can be run from command line, from the "**KEHM Demo Installation**" icon on liveuser desktop, or from the applications menu (under **System** submenu).

Once this script has run, the required services are started and the demo is ready for use.

The demo features a nice dashboard to inspect the health sensor xml output files using a html browser (at http://localhost:8000/kehm-RESULT.xml). The browser can be open with the "**KEHM Demo**" icon on liveuser desktop, or from the applications menu (under **System** submenu).

Please read the complete CMON-Line Release Notes included to learn more about the kehm technology and its numerous use cases. It can be open with the "**KEHM Release Notes**" icon on liveuser desktop, or from the applications menu (under both **System** and **Documentation** submenus).

10 / SM750 Graphical Chipset Support

The SM750 Silicon Motion chipset is a PCI Express display controller device.

The SM750 embedded graphics features include:

- 2D acceleration,
- 16 MB integrated video memory.
- The maximal supported resolution is: 1920x1080.

This device is fully supported by this Linux release .

This support is provided by the **sm750module** package.

This package contains the following software from Silicon Motion Inc. (SMI):

- Xorg modules (drivers used by the Xorg X server)
- Framebuffer kernel module (used by the low level console) with some additional configuration and service files provided by Kontron

The Xorg module requires an Xorg configuration file, but this file must be removed when using other graphics chips to have Xorg using the right driver.

This configuration file is automatically installed or removed by a service called "**sm750**" that is run at each boot:

- If an SM750 chip is detected, the symlink **/etc/X11/xorg.conf.d/50-sm750.conf** to **/usr/share/sm750module/sm750.xorg.conf** is created before the Xorg server starts.
- If no SM750 chip is detected, and that the **/etc/X11/xorg.conf.d/50-sm750.conf** is a symlink (not replaced by a custom file), it is removed.

On a systemd OS, to get the log for this service:

```
journalctl -e _SYSTEMD_UNIT=sm750.service
```

or

```
journalctl -e -u sm750
```

This service can be disabled, if needed, to stop any update of the **50-sm750.conf** configuration file:

```
systemctl disable sm750
```

For more information on the Xorg module configuration, refer to the Silicon Motion User's Manual by running the following command:

```
# man siliconmotion
```

A more detailed information can be found in the source files of the package (see text files inside .zip source files).

11 / KVX BSP Features

These features are provided by the **kvx_bsp** package.

11.1 Sensors

11.1.1 Sensors Overview

The **kvx_bsp** service automatically creates/updates the **/etc/sensors.d/kboard-sensors** symlink that points to a sensor file suitable for the detected Kontron board type.

To display sensors information, run the sensors command.

On VX305x board, the output should look like:

```
# sensors
ltc2945-i2c-0-6e

Adapter: SMBus I801 adapter at f000
 +12V Voltage : +11.90 V  (min =  +0.00 V, max = +102.38 V)
                          (lowest = +11.75 V, highest = +12.20 V)
 +12V Power   :  19.34 W  (lowest = 181.12 mW, highest =  42.96 W)
                          (max =   2.10 kW)
 +12V Current :  +1.62 A  (min =  +0.00 A, max =  +7.02 A)
                          (lowest =  +0.01 A, highest =  +3.62 A)

coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +42.0°C  (high = +82.0°C, crit = +104.0°C)
Core 0:        +42.0°C  (high = +82.0°C, crit = +104.0°C)
Core 1:        +42.0°C  (high = +82.0°C, crit = +104.0°C)
Core 2:        +42.0°C  (high = +82.0°C, crit = +104.0°C)
Core 3:        +43.0°C  (high = +82.0°C, crit = +104.0°C)
Core 4:        +43.0°C  (high = +82.0°C, crit = +104.0°C)
Core 5:        +43.0°C  (high = +82.0°C, crit = +104.0°C)
Core 6:        +43.0°C  (high = +82.0°C, crit = +104.0°C)
Core 7:        +43.0°C  (high = +82.0°C, crit = +104.0°C)

nct7802y-i2c-22-28
Adapter: I2C CPLD adapter
 3V3_SB Voltage :  +3.25 V  (min =  +3.14 V, max =  +3.46 V)
 VCORE  Voltage :  +1.81 V
 DDR4   Voltage :  +1.20 V  (min =  +1.14 V, max =  +1.26 V)
 +12V   Voltage : +11.92 V  (min = +11.40 V, max = +12.61 V)
 +5V    Voltage :  +5.07 V  (min =  +4.87 V, max =  +5.25 V)
 3V3AUX Present :  +3.30 V
 LTD temp       :  +34.0°C  (low  = -45.0°C, high = +85.0°C)
                            (crit = +95.0°C)
```

The "**3V3AUX Present**" info displayed above is not a voltage measurement but a presence status for this voltage on the VPX backplane. Possible values are only **0.00** or **3.30**.

In the output above, the "**max**" value for the LTC2945 "**+12V Voltage**" and "**+12V Power**" were not set by the BIOS and left to the maximum (default). They can be changed as explained below.

For the temperatures, the low, high and critical thresholds are displayed (if used). When the temperature of one of the core (coretemp sensors) goes beyond the critical threshold, an automatic reset of the board occurs.

For other sensors (not temperatures), the thresholds are named **min** and **max** instead of **low** and **high**.

When a sensor goes beyond one of the limits, an ALARM flag is displayed in the sensors output. So, in order to track down this kind of event, run the following command:

```
[root@]# sensors | grep ALARM
```

11.1.2 Sensors Threshold and Alarm Report

The thresholds are set by the BIOS to a suitable value. However, if needed, they can be changed under OS by specifying a new threshold in the **/etc/sensors.d/kboard-sensors** configuration file.

This is done by adding "**set <name> <value>**" lines, where **<name>** is one listed by the command:

```
    sensors -u
```

For example for an LM73 sensor, add the following lines to this sensor:

```
    set temp1_min -30
    set temp1_max 70
```

then restart or run :

```
    sensors -s
```

The thresholds can also be temporarily changed through the files under
/sys/devices/<path-to device>/hwmon/<path>/<file>

To get the list of files for each sensor:

```
    sensors -u
```

But the **/sys** values are not easy to use because they are raw values without the computation defined in **kboard-sensors**, and in milliDegrees, milliVolts, milliAmpers, or MicroWatts.

The ***_input** files give the raw value of the sensors.

An alarm processing can be implemented in polling mode according to the sensors output.

'**ALARM (MIN)**' is reported by '**sensors**' command when a value is lower than the minimum threshold.

'**ALARM (MAX)**' is reported by '**sensors**' command when a value is higher than the maximum threshold.

An interrupt mode is available, but only for the NCT7802 temperature, using the **nct7802y** module parameter **nct7802y_smbalert**.

By default this parameter is set to **0** meaning that the interrupt mode is not enabled.

If the interrupt mode is activated, the thresholds must be modified. It is necessary to fit the mode pre-configured by the BIOS. This mode sets the interrupt when the temperature is higher than the maximum threshold, and unsets the interrupt when the temperature is lower than the minimum threshold. In this case both threshold must have positive values.

To execute some commands on this interrupt, for example a shutdown, run the following script:

```
cat < /sys/devices/platform/cpld_i2c.6/i2c-22/22-0028/temp1_max_alarm_intr
echo "NCT7802 local temp went beyond the high limit"
shutdown -h now
```

11.2 kvx_board

This tool can be used to manage the supported VME/VPX board in the same chassis, through the backplane I2C bus.

The example below shows two VX6090 boards (dual sided):

```
# kvx_board -l
Slot 01.0: Board:VX6090 Power:ON LastRST:LOCAL-RESET RunState:ON
           BootState:OS-RUNNING Reset:OFF
           PowerProfile : requested:BIOS-SETUP current:BIOS-SETUP
           BITstatus : FAILED (POST)
Slot 01.1: Board:VX6090 Power:ON LastRST:SYSRESET RunState:ON
           BootState:BIOS-BOOT Reset:OFF
           PowerProfile : requested:BIOS-SETUP current:BIOS-SETUP
           BITstatus : OK
Slot 02.0: Board:VX6090 Power:ON LastRST:POWER-ON RunState:ON
           BootState:OS-BOOT Reset:OFF
           PowerProfile : requested:BIOS-SETUP current:BIOS-SETUP
           BITstatus : OK
Slot 02.1: Board:VX6090 Power:ON LastRST:WATCHDOG RunState:ON
           BootState:OS-RUNNING Reset:OFF
           PowerProfile : requested:BIOS-SETUP current:BIOS-SETUP
           BITstatus : OK

WARNING: at least one board above is reporting an error
```

The help of this command can be displayed with option **-h**:

```
# kvx_board -h

USAGE :  kvx_board <option> [<argument>] [<slotlist>]

Manage VPX/VME boards in the same backplane, using backplane I2C bus.

Options :
  -l            list all detected boards with their state.
  -r [0|1|p]    Reset (0), unreset (1), or do a reset pulse (p) on
                target board(s).
  -p [0|1]      Power-off (0), or power-on (1) the power supplies of
                target board(s).
  -w [0|1|2|3]  Request use of a power profile on target board(s).
                0:bios-setup, 1:down, 2:nominal, 3:up
                (not supported by all boards)

<slotlist> argument is required for -r, -p, or -w. If omitted with
-l, all boards are listed.

For a single sided board, a slot number can be any number from 1 to 21.
For a dual board, a .0 or .1 is appended for side A or side B/
A slotlist is a combination of slot numbers as follows :
  3 : slot 3 board, and both sides of board for dual boards.
  3.0 : slot 3 side A for a dual board.
  3.1 : slot 3 side B for a dual board.
  2-4 : slot 2 to 4 (both sides for dual boards).
```

```
  1-3,9,4.0,5.1 :
        slot 1 to 3, then 9 (including both sides for
        dual boards), then slot 4 side A and slot 5 side B.
  all : all detected boards.

For intervals (x-y), no eror is reported if a board is missing in the
interval (empty slot).
For a list (,), a missing board in the list returns an error.

All boards can be reached, including the one from which the command is
run. For dual sided boards, it is possible to control the other side of
the board through this command ... and also the local side (useful ?).

WARNING : keep in mind this command may reset or power-off the board
          from which it is run.

For VX6060, only side A is reachable, however both sides can be
reset using -r Ax (side A), or -r Bx (side B), or -r ABx (side A+B);
where x is 0, 1, or p. Using -r x does not work with this board.


The -l output contains the following information :
  Board:     board type
  Power:     internal board's PSUs state (STANDBY, ON)
  LastRST:   last reset cause (POWER-ON, WATCHDOG, SYSRESET, LOCAL-RESET)
  RunState:  reports if the board is running or not (OFF, ON).
             OFF if "Power:standby", or "Reset:on", or another reset
             source is active.
  BootState: state (level) at which the board is in its boot process
             (RESET, BIOS-BOOT, BIOS, PBIT, OS-BOOT, OS-RUNNING,
             COMPLETED, SHUTDOWN, REBOOT, USER-SPECIFIC)
  Reset:     state of board reset controlled through I2C
             (OFF, ON).
  PowerProfile : the requested power profile, and the current power
             profile at which the board is running, if supported.
             (BIOS-SETUP, DOWN, NOMINAL, UP)
  BitStatus: the status of the board. Can be OK, or FAILED with the
             reason (POST, RTC battery, PECI/Temp Alert, Onboard PSUs,
             Backplane Under-Voltage, Backplane Over-Voltage, PECI Crit,
             Therm Prot, Therm Trip, Catastrophic Error)
```

11.3 VPD Tool

NAME

vpdtool − display Kontron board VPDs (Vital Product Data)

SYNOPSIS

−a *boardtype*
− −boardtype
− −conffile or −f *file*
− −eclevel
− −help or −h
− −human or −H
− −keylist
− −macaddr
− −serialnumber
− −variant
−v variant

DESCRIPTION

vpdtool reads the VPDs (Vital Product Data) of many Kontron boards, including the following:

- ITC320/322 PENTXM2/4
- VM6050 VM6052 VM6054
- VM6250
- VX3020 VX3030 VX3035 VX3040 VX305x VX6058
- VX3230 VX3240
- VX6060 VX6070 VX6080 VM6090 VX6064

As it reads hardware ports via /dev/mem you need to be root to run it.

OPTIONS

−**a** *boardtype*

forces the architecture of the board to that given, e.g. VX3020

− −**conffile or** −**f** *file*

uses the given features definition file. This file says how to convert the vpd binary encodings into text descriptions.

− −**help or** −**h**

prints an option summary.

− −**human or** −**H**

displays features of the board in a human readable way.

− −**boardtype** − −**eclevel** − −**keylist** − −**macaddr** − −**serialnumber** − −**variant**

displays only the requested information. The options can be combined.

−**v variant**

force variant (will not be read from VPD). Can be used with -H
or --keylist to get the features or keylist for a given variant.

EXAMPLE

```
# vpdtool --human
VX3058 detected
Board type   : VX3058SA881
EC Level     : 14000A0
Serial Number: 1815281060016
Variant      : 00652000000205012000
MAC addresses  :
eth0: 0:0:de:52:cb:38
eth1: 0:0:de:52:cb:39
eth2: 0:0:de:52:cb:3a
eth3: 0:0:de:52:cb:3b

Features     :
PCB A
SA Class
Board pitch: 5HP / 1 inch
VX3058: 8-core CPU
Standard TDP Soc
Targetted TDP is 45W
SoC VCCIN tuning: 3-phase VCCIN
DDR PN is Micron MT40A512M8HX-093E:A (DDR4-2133, 4Gb, 0/95¿C, Mask A)
DDR: Dual bank memory configuration
DDR: One memory rank per bank (also called single die)
DDR: 4-Gbit devices
DDR: 4-Gbit devices
On-board PCIe switch PEX8725 equipped
Trusted Platform Module not equipped
ETH1: I210 controller equipped
ETH1: I210 EEPROM equipped
ETH1: On-board magnetic equipped
ETH1: 1000 BASE-T configuration
ETH2: I210 controller equipped
ETH2: I210 EEPROM equipped
ETH2: On-board magnetic equipped
ETH2: 1000 BASE-T configuration
P0 MUX1: P0.B7 & P0.A7 connected to CPLD GPIO3 & GPIO4
P0 MUX2: PCIE Common CLK connected from P0 to SOC
P1 MUX1: P1.E12 & P1.F12 connected to SoC USB2
SATA MUX: SoC SATA1 connected to P1 and SoC SATA4 connected to M2S1
DVI MUX: DVI connected to M2S1
PECI MUX: PCH PECI connected to CPU and IPMC => CPU IPMC
M2S1 (M.2 Mezzanine on Top Side): Connector equipped
M2S1 MUX: M2S1 connected to PCIe
M2S1 is MODP-GM2-SM750-xx
M2S2 (M.2 Mezzanine on Bottom Side): Connector equipped
M2S2 MUX: M2S2 connected to SATA
No predefined M2S2 mezzanine (a generic M2S2 may or may not be equipped).
FrontIO mezzanine connector equipped
FrontIO mezzanine: IDVI-VX305X-xxx
FrontIO mezzanine power supply: 5V
Front Connectors (Serial, USB, ETH) equipped
XMC connectors (J15, J16) not equipped
J16 Personality module: J16.F16 & J16.C16 not connected to V_5V0 and
XMC_PCISP not connected to J16
No predefined XMC mezzanine (a generic XMC may or may not be equipped).
No IPMI
SER2 MUX: Serial SER2 connected to SOC
Battery equipped in on-board battery holder
Connector for battery mezzanine not equipped
```

```
Supercap not equipped
Normal BOM (not extended temp. range)
Lead Free Process and Immersion Tin PCB Plating
```

FILES

/dev/mem
/dev/i2c/*

COPYRIGHT

Kontron

11.4 Sysvartool

This command is mainly used to get the report of the PBIT (Power Built In Test).

```
# sysvartool -A pbit -l
VX3058 detected
POSTs configured to run from command line:
       mem_data: PASSED
       mem_addr: PASSED
   mem_pattern1: PASSED
   mem_pattern2: PASSED
   mem_pattern3: PASSED
   mem_pattern4: PASSED
           core: PASSED
            rtc: PASSED
    ether_loop0: PASSED
    ether_loop1: PASSED
    ether_loop2: PASSED
    ether_loop3: PASSED
          hwmon: PASSED
sata0_controler: PASSED
sata1_controler: PASSED
            vpd: PASSED
ehci_controller: PASSED
xhci_controller: PASSED

PASSED  : 18
FAILED  : 0
NOT RUN : 0
TOTAL   : 18
```

```
sysvartool(1)              General Commands Manual              sysvartool(1)

NAME

        sysvartool - Kontron board system eeprom utility

SYNOPSIS

        sysvartool [-v] [-h] [-p] [-A area] [-l [variable]] [-a variable=value] [-d variable]
        [-f filename]


DESCRIPTION

        This  utility  displays or edits the contents of the system eeprom on kontron boards.
        The eeprom is partitioned into several areas, depending on the board.  These may
        include "uboot", "pbit", "vxworks", and  "linux".   By  default  the "user"  area is
        used.

        Each area has its own checksum, which is checked on read and updated on write.  An
        unitialised area will have aninvalid checksum.

        You need to be root to access the device.

OPTIONS

        The short and long version of each option is given below.

       -h --help
              shows the help message.
```

```
       -v --verbose
              more verbose output.

       -p --pbit
              shortcut of  -l -A pbit  to get the PBIT report.

       -A --area area
              select the area to display or edit.

       -l --list [variable]
              list all the variables and their value, or just the given variable.

       -a --add variable=value
              add or change the given variable with the new value.

       -d --delete variable
              delete the variable from the eeprom area.

       -f --file filename
              dump the contents of the area to the given file.
```

EXAMPLES

Display PBIT report.

```
    # sysvartool -p
    VX3058 detected
    POSTs configured to run from command line:
           mem_data: PASSED
           mem_addr: PASSED
        mem_bitflip: PASSED
        mem_addrpat: NOT RUN
       mem_addrpat2: NOT RUN
       mem_pattern1: PASSED (FAILED ONCE)
       mem_pattern2: PASSED
       mem_pattern3: PASSED
       mem_pattern4: PASSED
               core: PASSED
                tpm: PASSED
        pcie_vpx_sw: FAILED
          m2_bottom: PASSED
             m2_top: PASSED
             serial: PASSED
                rtc: PASSED
           sysflash: PASSED
               cpld: PASSED
               fram: PASSED
        ether_loop0: PASSED
        ether_loop1: PASSED
        ether_loop2: PASSED
        ether_loop3: PASSED
              hwmon: PASSED
    sata0_controler: PASSED
    sata1_controler: PASSED
                vpd: PASSED
             eeprom: PASSED
    ehci_controller: PASSED
```

```
            xhci_controller: PASSED
                     system: FAILED

            PASSED  : 27
            FAILED  : 2
            NOT RUN : 2
            TOTAL   : 31

        Add a variable to an unitialised user area.

            # sysvartool -a myvar=myvalue
            VX6060 detected
             area = 0, arch = 2
             ERROR: area checksum invalid
            Change successfully applied.

        List all variables in the user area.

            # sysvartool  -l
            VX6060 detected
             area = 0, arch = 2
            The user    section contains the following variables:
            myvar = myvalue

Kontron
sysvartool(1)
```

11.5 cpldtool

```
CPLDTOOL(1)
CPLDTOOL(1)

NAME

       cpldtool - Kontron board cPLD register utility

SYNOPSIS

       -a
       -d register
       -f register fieldname hexvalue ...
       -H
       -i register
       -s register hexvalue
       -v

DESCRIPTION

        This  utility interprets, displays, and sets flags in registers of the cPLD on Kontron
        boards including VX3230, VM6250, VX6060, VX3030,VM6050, VX3035, VX3040, VX6080,
        VM6052/VM6054, VX3058, VX6090, VX6064, VX6058 The cPLD is a small device that controls
        some  low-level aspects  of the board. See cpld(4).  You need to be root to access the
        device.  Do not change register values except as advised by Kontron.

OPTIONS

        Registers are specified using their name or hexadecimal offset, whereas values are
        assumed to be hexadecimal (whether prefixed  or  not by 0x)

       -a     displays the current value of all registers, with the value of all their fields.

       -d register
              displays the current value of the given register, with the value of all its
              fields.

       -H     to use with -a or -d to display all values in a human readable way instead of
              numerically.

       -i register
              displays information on the given register, listing each bitfield with its
              location in the register (mask).

       -s register hexvalue
              sets the given register to the given hexadecimal value.

       -f register fieldname hexvalue ...
              sets  the given register field by field.  Each field is specified by fieldname,
              as shown by option -i, and the hexadecimal value
              for that field.

       -v     more verbose output for the -a and -d options, detailing what some values mean.

EXAMPLES

       Display infos on register MEM_PROTECT at offset 0x9 :

       # cpldtool -i 9
       Reg 0x9 - MEM_PROTECT

       Field name: Boot_flash_select (mask:0x80)
```

```
       Explanation:
       Boot flash chip select configuration (from switch):
        0 : Boot on main BIOS
        1 : Boot on rescue BIOS
        ...
       Field name: USER_WP (mask:0x8)
       Explanation:
       USER level WP hardware protection:
        0 : No USER level WP correction
        1 : USER level WP active
        ...
       Field name: NVMRO (mask:0x1)
       Explanation:
       VPX NVMRO:
        0 : All non volatile memories are write enabled
        1 : All non volatile memories are write disabled

       Display register MEM_PROTECT :

       # cpldtool -d MEM_PROTECT
       Reg 0x9 - MEM_PROTECT = 0x00
            Boot_flash_select=0x0
            Boot_flash_toggle=0x0
            Both_boot_flash=0x0
            SSD_WP=0x0
            USER_WP=0x0
            SYS_WP=0x0
            VPD_WP=0x0
            NVMRO=0x0

       Display register MEM_PROTECT in a human readable way :

       # cpldtool -H -d MEM_PROTECT
       Reg 0x9 - MEM_PROTECT = 0x00
            Boot_flash_select = Boot on main BIOS
            Boot_flash_toggle = Boot flash is the one selected by Boot_flash_select
            Both_boot_flash = Only selected Boot flash is accessible/mapped
            SSD_WP = No protection
            USER_WP = No USER level WP protection
            SYS_WP = No SYS level WP protection
            VPD_WP = No VPD level WP protection
            NVMRO = NVMRO signal is low (unprotected)

       Set the USER_WP bitfield in register MEM_PROTECT :

       # cpldtool -f MEM_PROTECT USER_WP 1
```

SEE ALSO

cpld(4)

COPYRIGHT

Kontron

Example of output on a VX3058 board:

```
# cpldtool -a -H
VX3058 detected
Reg 0x0 - CPLD_ID = 0x04
              CPLD_Debug = standard version
              CPLD_Version = 0x4
Reg 0x1 - PCB_ID = 0x46
              Board_ID = VX3058 (3U Board VPX)
Reg 0x2 - FIRM_POST = 0x00
              PBIT_FAIL = No test failed
              PBIT_RUN = PBIT has not run
Reg 0x3 - PWON_STATUS = 0x60
              DPOST = 0x0
              BOARD_CLASS = RC board class
              POST_RTC = POST OK
Reg 0x4 - PWR_RST_CONFIG = 0x02
              PWRON_MODE = Power-on when VPX power present
              PEXRST_Dis = PLD_PEX_RST# is PLD_PLTRST# if bit1 = 1; otherwise is 0
              Alert_inhib = System Shutdown if THRMTRIP#, CATERR# or PROCHOT# Alerts occur
              PEXRST_Ctl = None : PLD_PLTRST# is propagated to PLD_PEX_RST# if enabled by
              bit 6.
Reg 0x7 - SERIAL_LINES_CTL = 0x04
              SERIAL_MODE = RS232
              SERIAL_TXEN = Transmit enabled
              SERIAL_TERM = 0x0

Reg 0x8 - PCI_MODE = 0x80
              XMC_PRESENT = XMC NOT installed
              XMC_SLOT_PRESENT = XMC slot is not present (Although XMC connectors J15 and
              J16 may be equipped).


Reg 0x9 - MEM_PROTECT = 0x00
              Boot_flash_select = Boot on main BIOS
              Boot_flash_toggle = Boot flash is the one selected by Boot_flash_select
              Both_boot_flash = Only selected Boot flash is accessible/mapped
              USER_WP = No USER level WP protection
              SYS_WP = No SYS level WP protection
              VPD_WP = No VPD level WP protection
              NVMRO = NVMRO signal is low (unprotected)
Reg 0xb - M2SLOTS_CONFIGURATION = 0x33
              M2_SLOT2_SLP = Active State
              M2_SLOT2_CFG = Module not present
              M2_SLOT1_SLP = Active State
              M2_SLOT1_CFG = Module not present
Reg 0xc - BOARD_CONFIGURATION = 0x90
              ETH2_LANSW_SEL = Routed to P2 VPX Backplane
              ETH1_LANSW_SEL = Routed to RJ45 Front panel connector
              BIOS_FAILSAFE = BIOS starts in normal mode
              DVI_FRONT_SEL = Routed to P2 VPX Rear Panel DVI port
              TPM_PRESENT = Not present
              IPMI_CONFIG = IPMI option not present
Reg 0xd - DRAM_CONFIG = 0xD4
              DRAM_CHANNEL = Dual CHANNEL(bank)
              DRAM_RANK = Single RANK
              DRAM_DENSITY = 4 Gb
              DRAM_SPEED = 2133 GT/s is supported
```

```
Reg 0xe - DIP_SWITCH_STATE = 0x00
              DBG_MD = normal operation (SW1[2] switch off)
              PCIESW_FAILSAFE = normal mode (switch off)
              PROCHOT = normal operation (switch off)
              FACTORY_MODE = normal operation
Reg 0xf - SERIRQ_CONTROL = 0xE0
              SMB_MSK = interrupt is masked
              PECI_ALERT_MSKK = interrupt is masked
              TIP_MSK = interrupt is masked
              GPIOs_INT = No interrupt occured
              SMB_INT = No interrupt occured
              ALERT_INT = No interrupt occured
              TIP_INT = No interrupt occured
              WDG_INT = No interrupt occured
Reg 0x55 - WDG_VALUE = 0x0F
              WDG_Value = 0xF
Reg 0x56 - WDG_CONTROL = 0x02
              WDG_Timeout = watchdog has not expired
              WDG_Clk_Period = Period is 1 Second
              WDG_Lock = Watchdog timer not locked
              WDG_Mode = Reset Mode
              WDG_Enb = Watchdog timer not enabled
Reg 0x57 - WDG_COUNT = 0x0F
              WDG_Count = 0xF

Reg 0x58 - WDG_STARTUP = 0x05
              WDG_On_At_Startup = watchdog not enabled at startup
              WDG_Restart_Count = 0x5


Reg 0x5b - ALERT_STATUS = 0x00
              CATERR = No error
              THERMTRIP = No error
              VRHOT = No error
              PROCHOT = No error
              PECI_ALERT = No alert
              FIVR_FAULT = No error
              PCIE_ERR = 0 No error
              PCH_TEMP_ALERT = No alert
Reg 0x6a - GEO_ADD = 0x3E
              SYSCON = SYSTEM controller
              Error_Status = parity is valid
              GAP = 0x1
              GA = slot #1
Reg 0x6b - GA_GPIO = 0x3E
              CUR_SYSCON = SYSCON# at 0
              Parity_Status = GAGPIO parity is valid
              GAPGPIO = 0x1
              GA_GPIO = 0x1E
Reg 0x70 - VPX_CONTROL = 0xD4
              GDiscrete1_Ctl = VPX GDISCRETE1# signal not driven (high-Z)
              GDISCRETE_Status = GDISCRETE# Signal is high
              MSKR2LOC = Reset propagated
              VPX_RST = Reset not asserted
              VPX2LOC = Reset propagated
              LOC2VPX = Reset propagated
```

```
Reg 0x71 - PCIe_SWITCH_VPX = 0x10
              VPX_REFCLK = Board receives VPX REF clock
              VPX_AUXCLK = Board receives VPX AUX clock
              GDISCRETE_DEFAULT = 0x1
              PEX_SW_MODE = Transparent mode
              PEX_I2C_CFG_EN = disabled (default)
Reg 0x72 - I2C_BOARD_STATUS = 0xF5
              Power_Status = Power ON
              Reset_Source = Local Reset
              Reset_Status = PWOK and reset unasserted
              Boot_Status = OS-RUNNING : written by OS at the end of boot
Reg 0x73 - I2C_BOARD_CONTROL = 0x91
              Board_Id = VX305x/VX6058 family
              Reset = No Reset
              Power_OnOff = Power On
Reg 0x74 - I2C_ERROR_STATUS = 0x00
              Alert = no alert
              POST_Error = no error
              POST_RTC = POST OK
Reg 0x75 - I2C_PORT80 = 0x00
              Port_80 = 0x0
Reg 0x76 - I2C_FAILCODE = 0x00
              PEX_EEPROM_CRC_ERROR = no error
Reg 0x78 - I2C_MISC = 0x00
              Force_rescue_BIOS = not forced
              Force_EFI_Shell = not forced
              Power_CUR = BIOS setup and/or onboard switches
              Power_REQ = BIOS setup and/or onboard switches
```

11.6 FMRAM Tool

The **fmram** tool gives a transparent access to the FMRAM I2C device made of several banks. This device can be used to store some user data in a very fast way (no write wait state) and as often as needed (unlimited write cycles), unlike a standard EEPROM.

```
# fmram -h
```

Usage **fmram [options]**

a tool to read or write the FerroMagnetic RAM

Options are:

-h : this help

-r <value> : read at offset <value> (default 0)

-w <value> :write at offset <value> (default 0)

-s <value> : data size to read or write (default 4)

-f <filename> : file name used to store (optional) or read (mandatory) data

Example:

fmram -w 0x10 -s 0x11 -f data_file: store 17 bytes read from data_file to the ferromagnetic RAM at offset 16

Note a device above 64 KB is split into multiple 64 KB banks at consecutive I2C addresses. This tool is automatically selecting the target bank according to the offset so that this is transparent to the user, including when a read or a write is crossing a bank boundary.

11.7 Misc Tools

Some other tools are provided for debug and maintenance:

- **cpld_update**: to update the CPLD using an JEDEC image file
- **i2c_transfer**: low level raw I2C tool for debug and test purposes
- **io**: to access memory
- **port**: to access I/O ports
- **switch_3905_cfg**: to configure a VX3905 switch

To get help on these commands, run them with option **-h**.

An **/usr/local/bin/komliveinstall.sh** script is also provided as example to do a non-interactive installation of the live image to a storage media.

It uses "**liveinst**" in console mode, but as **liveinst** does not support kickstarts, this script workarounds the problem by providing automated input from console as an operator would do.

The only thing to supply the storage device name or model name (for example "**sda**", or "**GLS85LS1032A**").

The installation is in English-US (language and keyboard), unless option **-fr** is used (French). See header of script for more information.

11.8 CPLD

NAME

cpld – Kontron board facilities, cpld_i2c, cpld_leds, cpld-gpio, cpld-wdt

DESCRIPTION

This man page describes some of the facilities made available by the cPLD (complex programmable logic device) on several Kontron boards. Not all boards provide all the facilities.

The cPLD is a small device that controls some low – level aspects of the board, including power up sequencing, reset, gpios, i2c buses including one for communication to the backplane, timer, watchdog, leds, and many configuration and control features.

The device is visible through 1 – byte registers in i/o port space from addresses 0x0800 to 0x08FF. The facilities are implemented in several drivers. The driver sources are provided.

cpld
The cpld driver is needed by the other drivers. It provides the geographical id (slot number) of the board in the file /**proc**/**geo_id**.

Some other basic features can be accessed with the cpldtool(1) utility, or by direct i/o on the appropriate port. For example, use dd on /dev/cpld, /dev/port or the port command as user root (beware: inappropriate i/o on ports may crash the system). As an example port 0x800 holds the version number of the cpld:

```
[root]# port 0x800
@0x800 = 0x06
```

cpld_i2c
The cpld_i2c driver implements some i2c buses made available through the standard OS i2c interface.

cpld_leds
The cpld_leds driver provides standard OS led class devices in /**sys**/**class**/**leds**/. They have filenames of the form *devicename:colour*.

Each cpu has several user settable leds, with devices named led2,led3,...

They implement 3 brightness levels by writing a value to *devicename:colour*/brightness

- 0 on
- 1 slow blink
- 2 fast blink

This switches each LED to user mode. Write 0 to *devicename*:OFF/brightness to switch an LED off, or any other value to exit from user mode.

cpld – gpio
See the separate man page for the driver providing gpios implemented by the cpld.

cpld – wdt
See the separate man page for the driver providing a watchdog implemented by the cpld.

SEE ALSO
cpld – gpio(4)
cpld – wdt(4)
cpldtool(1)

FILES
/proc/geo_id
/dev/cpld
/dev/port
/dev/i2c-22, /dev/i2c-23, /dev/i2c-24

COPYRIGHT

Kontron

11.9 CPLD-WDT

CPLD-WDT(4)
CPLD-WDT(4)

NAME

cpld-wdt - Kontron board cpld_wdt watchdog driver

DESCRIPTION

This man page describes how to use the watchdog implemented by the cpld on various Kontron boards.

The principle of a watchdog is to automatically provoke some action after a given time passes without the watchdog being prodded by some process. This would indicate that the process is no longer working correctly. The cpld_wdt watchdog actions are to do nothing, to reset the board, to generate an interrupt that can wake up some other process, or to reboot. The prodding is done by writing to the watchdog device, which restarts the timeout.

The cpld_wdt module implements the standard Linux watchdog API, detailed in file **Documentation**/**watchdog**/**watchdog-api.txt** in the kernel source rpm. The supported features are described below.

Loading Module
There can be many devices and modules providing the watchdog API. Ensure only the cpld_wdt module is being used. For example, the standard iTCO_wdt watchdog module is usually blacklisted by a file in /**etc**/**modprobe.d**/ to avoid it being loaded.

The cpld_wdt module has to be explicitly loaded, for example by creating a systemd file with a name ending in ".conf" in /**etc**/**modules-load.d**/ listing the modules to load, one per line, for example:

```
cpld_wdt
```

Module Options

To provide initial options to the module, create a file with a name ending in ".conf" in /**etc**/**modprobe.d**/ holding a line such as:

```
options cpld_wdt timeout=50 trigger_mode=1 nowayout=1 lock=1
```

The module options, shown by the command "modinfo cpld_wdt", include

timeout

the integer timeout in seconds, from 0 to 510 with a resolution of 2 seconds. The default is 30. The actual timeout is rounded to the next odd value, with an additional uncertaintly of -0/+1 second.

trigger_mode

the action to do when the timeout expires. An integer value

- 0 to simply countdown with no action,
- 1, the default, to reset the board,
- 2 to generate an interrupt that can be used to wake up a read on the device, or to reboot if no read is pending.
- to power-cycle the board (not supported by all boards),

nowayout

an integer 1 if there is no way to stop the watchdog. The default is 0, which stops the timer if the magic character "V" is written just before the device is closed.

lock

an integer 1 to lock the timeout and mode (no more changeable until next board reset), and prevent the watchdog from being stopped by software. The default is 0 (not locked)

Usage from Scripts

The watchdog is implemented with the standard device /**dev**/**watchdog**. This file accepts ioctls to use the watchdog, but it can also be used without if ioctls are not easy, from example from shell scripts.

Reading this file to get some status info, or writing 'T' to it to clear the watchdog timeout status are Kontron specific.

To avoid the need to be root, simply change the ownership or permissions of /dev/watchdog.

--- Reset or power-cycle mode example :

Load the module with the required configuration : reset or power-cycle mode, with a timeout of 10 seconds:

```
# rmmod cpld_wdt
```

then

```
# modprobe cpld_wdt timeout=10 trigger_mode=1 nowayout=0  # for reset
```

or

```
# modprobe cpld_wdt timeout=10 trigger_mode=3 nowayout=0  # for power-cycle
```

Start the watchdog by writing to the device:

```
# echo > /dev/watchdog
```

Ping the watchdog faster than every 10 seconds:

```
# while sleep 5; do echo > /dev/watchdog; done
```

After a while stop the loop and wait for the board reset / power-cycle. This sets the timeout status bit that also sets a red LED on the board's front panel.

Once the board has restarted, check the watchdog timeout status :

```
# dd if=/dev/watchdog count=1 2>/dev/null
```

A 'T' character should be returned to report a timeout (otherwise no 'T' is returned). A 'R' character should also be returned to report the last board reset that was due to the watchdog (otherwise no 'R' is returned).

The same command also returns a 'E' if the watchdog is currently enabled (otherwise no 'E' is returned).

A "dd" is used here to only do a single read(), because a "cat" would loop forever calling read() until no character is returned which only occurs without timeout status set and with watchdog disabled.

To clear this timeout status (and also switch the red LED off), write a 'T' to the device but this write() also start/restarts the watchdog, so if it must stay off, also add a 'V' to disable it on close :

```
# echo TV > /dev/watchdog
```

--- Interrupt mode example :

Load the module with the required configuration : interrupt mode, with a timeout of 10 seconds:

```
# rmmod cpld_wdt
```

```
# modprobe cpld_wdt timeout=10 trigger_mode=2 nowayout=0
```

Start the watchdog, or refresh it if already started, by writing to the device:

```
# echo > /dev/watchdog
```

Waiting the timeout can be done by a blocking read that blocks until timeout:

```
# cat /dev/watchdog
```

WARNING : if no read is pending when the timeout occurs, the OS reboots.

The timeout status is immediately cleared on timeout, and the watchdog restarted. To prevent an OS reboot, another read must be done or the watchdog must be stopped with :

echo V > /dev/watchdog

--- Timer mode example :

Load the module with the required configuration : timer mode, with a timeout of 10 seconds:

```
# rmmod cpld_wdt
# modprobe cpld_wdt timeout=10 trigger_mode=0 nowayout=0
```

Start the watchdog, or refresh it if already started, by writing to the device:

```
# echo > /dev/watchdog
```

Poll the watchdog for timeout :

```
# cat /dev/watchdog
```

This returns nothing until timeout. After timeout, this command returns a single character 'T' (timeout) and this clears the timeout status. The watchdog is automatically disabled on timeout.

Watchdog Ioctl API

The following standard ioctls are supported.

WDIOC_GETSUPPORT

```
#include <linux/watchdog.h>
struct watchdog_info ident;
ioctl(fd, WDIOC_GETSUPPORT, &ident);
```

returns in the structure the fields

identity

the driver identification "cPLD WDT"

firmware_version

is always 1

options

describes the supported features, namely **WDIOF_KEEPALIVEPING**, **WDIOF_SETTIMEOUT**, **WDIOF_MAGICCLOSE**, and **WDIOF_CARDRESET**.

WDIOC_GETSTATUS, WDIOC_GETBOOTSTATUS

```
int status, bstatus;
ioctl(fd, WDIOC_GETSTATUS, &status);
ioctl(fd, WDIOC_GETBOOTSTATUS, &bstatus);
```

Returns watchdog status info, and boot related watchdog status info.

For both ioctls, the **WDIOF_KEEPALIVEPING** bit is set if a watchdog ping has been done (by **WDIOC_KEEPALIVE** or by a write()) ,since last call to these ioctls.

The **WDIOF_CARDRESET** bit is returned by **DIOC_GETSTATUS** if the timeout status flag is set, and returned by **DIOC_GETBOOTSTATUS** if the last board reset was due to the watchdog. Clearing the timeout status flag does not clear the bit in **DIOC_GETBOOTSTATUS**; only another non-watchdog reset can do that. Also a non-watchdpg reset does not clear the bit in **DIOC_GETSTATUS**; only software can do that by clearing the timeout status.

WDIOC_SETTIMEOUT

```
int timeout = ...;
ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
```

sets the timeout in seconds. The actual timeout is rounded to the next odd value, with an additional uncertaintly of -0/+1 second.

WDIOC_GETTIMEOUT

```
ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
```

returns the current timeout setting in the argument (not the dynamically changing counter value).

WDIOC_SETOPTIONS

```
int options = ...;
ioctl(fd, WDIOC_SETOPTIONS, &options);
```

configures the given options. These are **WDIOS_DISABLECARD** to disable the watchdog, and **WDIOS_ENABLECARD** to enable the watchdog.

A Kontron specific option is also implemented to clear the watchdog timeout status : WDIOS_KCLEARTMO = 0x8000

WDIOC_KEEPALIVE

```
ioctl(fd, WDIOC_KEEPALIVE, 0);
```

refresh (ping) the watchdog, enabling it if not yet enabled, and restarting the countdown timer.

WDIOC_GETTIMELEFT

```
int val;
ioctl(fd, WDIOC_GETTIMELEFT, &val);
```

get the time left before watchdog expiration (current countdown). If the watchdog is disabled, -1 is returned, which can beused to know if the watchdog is currently enabled or not.

SEE ALSO

cpld(4)
cpldtool(1)

FILES

/dev/watchdog
Documentation/watchdog/watchdog – api.txt

COPYRIGHT

Kontron

11.10 LEDs

The driver **leds_cpld** handles the front panel LEDs for user mode.

The driver **leds_cpld** creates a list of special files and classes in **/sys**:

```
[root]# ls /sys/class/leds/
led3:amber  led3:OFF  led4:amber  led4:OFF  led5:amber  led5:OFF
led3:green  led3:red  led4:green  led4:red  led5:green  led5:red
```

For each LED, there are three different colors available: green, red, amber that can be set by addressing the related file.

For each color (which are exclusive), there are four different modes:

- ON (echo 0, see following example)
- slow blinking (echo 1, see following example)
- fast blinking (echo 2, see following example)
- OFF (echo 1, see following example)

Example, to set these different modes on the LED 2 in amber:

```
[root]# echo 0 > led2:amber/brightness The LED 2 is ON in AMBER
[root]# echo 1 > led2:amber/brightness The LED 2 blinks low in AMBER
[root]# echo 2 > led2:amber/brightness The LED 2 blinks fast in AMBER
[root]# echo 0 > led2:OFF/brightness The LED 2 is OFF
```

When setting led2 or led3, both led2 and led3 are set to user mode.

To exit from this mode, set led<n>:OFF/brightness to a non null value; for example

```
[root]# echo 1 > led2:OFF/brightness
```

- VX305x: On the front panel, the name of the led1 is L1, led2 is L2 and led3 is L3.

The led1 (L1 and L4) is not manageable at user level, so no special file for led1 is available.

11.11 GPIOs

NAME

cpld-gpio - Kontron board gpio-cpld driver gpios

DESCRIPTION

This man page describes how to use the gpios implemented by the cpld on various Kontron boards.

The cpld gpio driver provides an API using files in the /sys filesystem. Reads and writes to the files are directly handled by the driver, provoking reads and writes of hardware registers. Reads normally return a short string terminated with a newline. Further reads need to seek back to the start of the file or re-open it. Writes should normally be of a single string, optionally terminated with a newline. Writing values other than those described below are undefined, but usually result in a write error with errno EINVAL Invalid argument.

Usage from Program
Operations can be easily done from a shell script. When used from a program, open the files for simultaneous read and write, except for "intr_stat" (read only). Read in one go into a buffer big enough to accept the whole reply, including newline (i.e 11 characters). Use lseek(2) to rewind the file descriptor before each read. Write each value using a single system call. There is no need to lseek before writes.

Permissions
By default the files are owned by root and only writeable by the owner. If necessary, these ownerships and permissions can be changed (each time the driver is loaded), for example by a udev rule.

Sysfs Files
There is one directory per gpio, starting with gpio1, under the class directory:

/sys/class/cpld-gpio/

The number of gpios depends on the board. Each directory has the following files. The string values described are those that can be written, or will be read.

direction holds the value "in" (incoming signal) or "out" (outgoing). The default is "in". For example:

```
[root]# echo out >/sys/class/cpld-gpio/gpio4/direction
[root]# cat /sys/class/cpld-gpio/gpio4/direction
out
```

value holds the value "0" (low signal) or "1" (high). On read return the current value of the gpio. On write sets the value of an outgoing gpio. For example:

```
[root]# echo 0 > /sys/class/cpld-gpio/gpio4/value
[root]# cat /sys/class/cpld-gpio/gpio3/value
1
```

polarity holds the value "hi" (interrupt when signal active on high level or rising edge), or "lo" (active on low level or falling edge). The default is "hi".

mode holds the value "edge" (interrupt on rising or falling edge) or "level" (on high or low level). The default is "edge".

toggle holds the value "on" (interrupt on any state change) or "off" (interrupt only on the configured polarity and mode). When on it overrides the polarity and mode settings. The default is "off".

interrupt holds the value "on" (enable interrupts) or "off" (disable). When an interrupt occurs in level mode, this value is automatically reset to "off" to avoid a permanent interrupt. The default is "off".

intr_stat is a read-only count of the number of interrupts received on the gpio. This is an unsigned 32 bit decimal value which can be up to 10 digits long, plus a newline. This file is read-only, and writing to it will result in a write error of EPERM Permission denied, or EIO Input/output error. For example:

```
$ cat /sys/class/cpld-gpio/gpio5/intr_stat
12
```

value_it on read waits for an interrupt on the gpio, and then returns a string of length zero. If interrupts have not been enabled, the file acts like the value file. The file acts like the value file for all writes.

To get an interrupt you must set the direction, polarity, mode, toggle and interrupt files appropriately, and then read value_it until it returns, or poll the intr_stat value until it changes.

Dual Purpose GPIO
On some boards, gpio2 is a dual purpose gpio which can also be used to generate a Maskable Reset. It is configured using cpld registers that can be set using cpldtool, or low level port accesses.

To check if the board provides this feature run the command:

```
[root]# cpldtool -a|grep MSKR2LOC
```

If MSKR2LOC is not found, gpio2 is not dual purpose. To configure gpio2 as a standard gpio which does not issue a local reset, on a VPX board:

```
[root]# cpldtool -f VPX_CONTROL MSKR2LOC 0
```

on a VME board:

```
[root]# cpldtool -f VME_CONTROL MSKR2LOC 0
```

or on either type of board specify the register by number instead:

```
[root]# cpldtool -f 0x70 MSKR2LOC 0
```

Use the same command with value 1 to restore the configuration. To do the same using low level port accesses, read the register and calculate the new value before writing it back. For example:

```
[root]# port 0x870
@0x870 = 0x1d
[root]# port 0x870 0xd
@0x870 <- 0x0d
[root]# port 0x870
@0x870 = 0x0d
```

SEE ALSO

cpld(4)
cpldtool(1)

FILES

/sys/devices/platform/cpld_gpios/cpld-gpio/gpio[1-8]

COPYRIGHT

Kontron

11.12 I2C

11.12.1 Master Interfaces

The **cpld_i2c** driver provides three I2C buses.

- **/dev/i2c-22**: for local onboard devices (NTC7802, EEPROMs, FMRAM, ...). This bus has a dual speed clock with default speeds at 1 MHz and 100 KHz. The clock that is used for a transfer depends on the target device.
- **/dev/i2c-23**: backplane bus that is only functional when the board is at physical VPX slot #1 or is VME system controller, for an optional communication with the chassis. The default speed is 100 KHz.
- **/dev/i2c-24**: backplane bus that is bussed between all boards in the same backplane for out-of-band communication between boards and an optional chassis monitoring board. Supports multimaster operation and clock stretching. The default speed is 100 KHz.

These busses support the **I2C_FUNC_I2C** feature, so the **I2C_RDWR ioctl** can be used to do combined raw I2C transfers (as done by the **i2c_transfer** tool).

For some boards, the multimaster operation requires a minimal CPLD version. Without at least this version, make sure to have only one master on the bus at a time, otherwise an unpredictable result may occur. The minimum CPLD versions are:

- VX6060: none: multimaster NOT supported.
- VX3030: 0x11
- VM6050: 0xE
- VX3035: 0x9
- VX3040: 0x7
- Other: none: multimaster supported by all versions

The CPLD version can be displayed with :

```
# port 0x800
```

or

```
# cpldtool -d 0
```

By default, devices **/dev/i2c-23** and **/dev/i2c-24** are only present if the board is at physical VPX slot #1 or is VME system controller. This is a legacy behaviour for older boards on which the multimaster operation was not supported. To enable then for other slots, the "cpld" module must be loaded with the parameter "**force_i2c_extend=1**". This can be done by adding this option to a file under **/etc/modprobe.d/**, or by adding **cpld.force_i2c_extend=1** to the kernel command line.

However even if the device **/dev/i2c-23** is present for all slots with this option, it is only functional at physical VPX slot #1 or when VME system controller.

11.12.2 Slave interface

In addition to the master interfaces listed above, each board has a slave interface on the backplane that can be used to access some of its internal cPLD registers from the master **/dev/i2c-24** interface of any board. This access can be used to reset, power-cycle, or get the status of any board from any other board or from a chassis management board. This slave interface is running from the board's standby power so is reachable even if the board's internal PSUs are off (due to standby mode or hardware fault).

Any board can also be reached from itself, but this can be done more efficiently through the LPC interface (using **cpld-tool**). However this is very useful for a dual sided board to reach side B from side A or side A from side B.

The "**kvx_board**" command provides control and monitoring of the boards through these registers.

These registers are fully managed by the Kontron's Chassis Monitoring Board (CMB) that can be optionally present in a chassis. This board is usually powered from a standby power and can access the boards in the backplane even if only standby power is present.

The slave address (7-bit) of each board is: **0x17** + **<physical slot number>**. The register to access is selected by sending a 1 byte "register offset" from **0x0** to **0x6**. On dual-sided boards, use **0x0** to **0x6** to access cPLD A, and **0x10** to **0x16** to access cPLD B.

Then the data byte to write to the register is sent, or a restart followed by a byte read is done.

kvx_board is the recommanded command to manage the boards, however here are some basic examples using the low level **i2c_transfer** tool:

- Reset board at slot #2 (address **0x19**):

```
# i2c_transfer /dev/i2c-24 0x19 w 1 1 0x02 # reset on
# i2c_transfer /dev/i2c-24 0x19 w 1 1 0x00 # reset off
```

  for dual boards, the commands above are acting on side A; for side B use:

```
# i2c_transfer /dev/i2c-24 0x19 w 11 1 0x02 # reset on
# i2c_transfer /dev/i2c-24 0x19 w 11 1 0x00 # reset off
```

- Power-off then power-on the board at slot #3 (address **0x1A**):

```
# i2c_transfer /dev/i2c-24 0x1A w 1 1 0x00 # off
# i2c_transfer /dev/i2c-24 0x1A w 1 1 0x01 # on
```

  for dual boards, the commands above are acting on side A; for side B use:

```
# i2c_transfer /dev/i2c-24 0x1A w 11 1 0x00 # off
# i2c_transfer /dev/i2c-24 0x1A w 11 1 0x01 # on
```

- Write and read then scratchpad register of board at slot #4 (address 0x1B, register 5) :

```
# i2c_transfer /dev/i2c-24 0x1B w 5 1 0xA5 # write A5 byte
# i2c_transfer /dev/i2c-24 0x1B r 5 1 1    # read back one byte
```

  This register is a general purpose register, supported only since VX3040. It can also be reached locally on LPC at IO port 0x877.

- Read the last value written to debug port 80 of board at slot #2 (address 0x19, register 3):

```
# i2c_transfer /dev/i2c-24 0x19 r 3 1 1
```

  This port 80 sniffer feature is useful for debugging a BIOS failing to start.

Contact Kontron for more information on this topic.

12 / VME interface (VME boards only)

On VME boards (VMxxxx), a PCI to VME bridge called "Alma" is implemented. It is managed by the **vmetoolkit** package containing:

- **almadriver**: the Alma PCI to VME bridge driver
- **almavmechan**: the utility to manage ingoing and outgoing VME channels
- Several test tools: Dma / Dma2 / MasterSgl, GEN_intr / RECV_intr, mbm3k, probeVmeBoard
- The source code of these tools as examples under **/usr/share/vmetoolkit/examples**
- Man pages (**almavme**, **almavmechan**, **mbm3k**)

On VPX boards (VXxxxx), the **almavme** driver is not loaded and the related software is useless.

12.1 ALMAVME

NAME

almavme – Alma VME Linux toolkit driver API

SYNOPSIS

Kernel API (from a kernel driver)

```
#include <almavme.h>

int alma_pci_channel_alloc(char *name, u32 vmeaddr, unsigned long pciaddr, u32 size, u32 flag);
int alma_pci_channel_free(int id);
unsigned long alma_vme_channel_alloc(char *name, u32 vmeaddr, u32 size, u32 flag);
int alma_vme_channel_free(unsigned long address);
u32 alma_cpuadr_to_vmeadr ”(unsigned long address);

int alma_user_dma(u32 vmeaddr,unsigned long pciaddr,u32 size,u32 flags);
int alma_user_dma_mcast(u32 vmeaddr, unsigned long pciaddr,u32 size,u32 flags,u32 select);

int alma_get_vme(void);
int alma_free_vme(void);
int alma_ctrl(int command, int config);

int request_vmeirq(unsigned int vector, void *handler);
int free_vmeirq(unsigned int vector);
void alma_vmeintcontrol(unsigned int mask, unsigned int ienable);
int disable_vmeirq(u32 level);
int enable_vmeirq(u32 level);
int alma_vmeinterrupt(unsigned char level, unsigned char vector);
int alma_phys_mem_alloc(struct alma_phys_mem_object *phys_mem);
int alma_phys_mem_free(struct alma_phys_mem_object *phys_mem);
```

User Space API (from an application)

```
#include <linux/almavme.h>
int fd = open(”/dev/almavme”, O_RDWR);
if(fd<0){
perror(”Error opening the almavme device”);
exit(1);
}
alma_ioctl_arg_t alma_ioctl;
alma_ioctl.intr.level = 1;
alma_ioctl.intr.vector = 2;
```

```
int error = ioctl(fd, VMEIOCTL_VMEINTR_GEN, &alma_ioctl);
close(fd);
```

See the User Space sections below.

DESCRIPTION

This describes the kernel service calls available for VME device drivers or application development on Kontron platforms using the almavme driver for all boards equipped with Alma2e and Alma2f.

Alma2f provides a highly integrated single chip solution to interface a VME64 bus with 2eSST protocol (two-edge Source-Synchroneous Transfer) and a 32-bit 66 MHz PCI Bus. All bridge features are programmable from the PCI bus or the VME bus. See the Alma2f User Manual CI.DT.A00.

In the following, functions are described from the kernel point of view, and examples show how to call them from a user application.

User Space Ioctls
All ioctls take a command argument, and an appropriate struct as third argument. As **alma_ioctl_arg_t** is a union of most of the possible struct types, it can be used as a suitable third argument for most of the ioctls. The descriptions below assume the following usage:

```
alma_ioctl_arg_t A;
A... = ...;
ioctl(fd,command,&A);
```

VME Access
All accesses to or from the VMEbus go through the PCI32 bus and the PCI-to-VME bridge hardware. No specific windows (slave or master) are opened at load time by the almavme device driver (except for the slave window to access Alma register space (A16) that is inherited from the firmware setup on VMPCx and PN3 boards). See the ”Mapping Service Calls” section below on how to open slave or master windows.

Mapping Service Calls
alma_pci_channel_alloc

```
int alma_pci_channel_alloc(char *name, u32 vmeaddr, unsigned long pciaddr,u32 size,u32 flag)
```

This allocates a slave channel from the VME to the PCI (i.e. incoming accesses to the board). Up to 15 channels can be opened, 8 of them support 2ESST (see the 2ESST section). The parameters of the routine describe the VME-PCI channel:

name is the name of this channel. It can be up to 12 bytes long, including the final null char. It is only used to label this channel in the list of channels printed by the almavmechan(1) utility.

vmeaddr is the VME physical address where the board will respond on the VME. It must be aligned to a 1MB boundary and also to the rounded-up channel size.

pciaddr is the physical address generated on the board when the PCI/VME bridge responds to the vmeaddr address on the VME bus. The address can be in PCI-IO space, PCI-MEM space or DRAM space (the flag field defines which one is used). It must be aligned to a 1 MB boundary and to the rounded-up channel size.

size is the size of the channel. It must be a multiple of 1 MB and it is rounded up to a power of 2.

flag contains the options for the channel. Options are ’ored’ from the following:

MEFG_WRTPOST access is write posted (default NO)

VMEFG_READAHEAD access is read ahead (default NO)

VMEFG_LEBE_*xx* defines the little/big endian conversion mode (default is address coherency)

***** The pci space which is one of:

VMEFG_PCIIO access PCI-IO

VMEFG_PCIMEM access PCI-MEM

VMEFG_DRAM access DRAM (default).

***** The AM code, which is decoded on the VMEbus for this channel. The default is **VMEFG_AM_A32SDATA** (0x0D). An AM code is specified as either:

VMEFG_AM_*xxx* flags that predefine standard AM codes, or

VMEFG_AM_DIRECT ored with multiple AM selecting bitmasks. The bitmask 0x0FF00000 defines which AM[5..3] are valid and the bitmask 0x000FF000 defines which AM[2..0] are valid. Each acceptable value of AM[2..0], i.e. the eight values 0 to 7, is selected by setting a bit in the bitmask, (1<<12) for 0, (1<<13) for 1, etc. To simplify, the **VMEFG_AMSLV_*xxx*** flags can be used when several AM codes must be decoded. This is below the defined set of available AMs:
A32 USER MBLT (AM 0x08) fg = 0x00201013
A32 USER DATA (AM 0x09) fg = 0x00202013
A32 USER PROG (AM 0x0A) fg = 0x00204013
A32 USER BLT (AM 0x0B) fg = 0x00208013
A32 SUPER USER MBLT (AM 0x0C) fg = 0x00210013
A32 SUPER USER DATA (AM 0x0D) fg = 0x00220013
A32 SUPER PROG (AM 0x0E) fg = 0x00240013
A32 SUPER BLT (AM 0x0F) fg = 0x00280013
A24 USER MBLT (AM 0x38) fg = 0x08001013
A24 USER DATA (AM 0x39) fg = 0x08002013
A24 USER PROG (AM 0x3A) fg = 0x08004013
A24 USER BLT (AM 0x3B) fg = 0x08008013
A24 SUPER USER MBLT (AM 0x3C) fg = 0x08010013
A24 SUPER USER DATA (AM 0x3D) fg = 0x08020013
A24 SUPER USER PROG (AM 0x3E) fg = 0x08040013
A24 SUPER USER BLT (AM 0x3F) fg = 0x08080013
A16 USER (AM 0x29) fg = 0x02002013
A16 SUPER (AM 0x2D) fg = 0x02020013
A16 LCK (AM 0x2C) fg = 0x02010013

A16 USER, A16 SUPER, A16 LCK can not be used to initialize a VME to PCI memory window due to the current driver window size definition that must be higher or equal to 1MB. A24 USER MBLT, A24 SUPER USER MBLT are identify as AM that present some transfer error. For these cases please contact the Kontron support.

This routine returns an identifier from 0 to 15 on success and -1 on failure. It can be called from user space with the ioctl() **VMEIOCTL_PCI_CHAN_ALLOC.**

alma_pci_channel_free

int alma_pci_channel_free(int id)

This is used by the VME driver to free a VME-PCI channel allocated by alma_pci_channel_alloc(). The parameter id is the channel identifier returned by alma_pci_channel_alloc(). It returns 0 on success, -1 on failure. It can be called from user space with the ioctl **VMEIOCTL_PCI_CHAN_FREE.**

alma_vme_channel_alloc

unsigned long alma_vme_channel_alloc(char *name, u32 vmeaddr, u32 size, u32 flag)

This allocates an outgoing master channel from PCI to VME. The parameters of the routine describe the desired PCI-VME channel:

name is the name of this channel. It can be up to 12 bytes long, including the final null char. It is only used to label this channel in the list of channels printed by the alma_vmechan utility.

vmeaddr is the VME physical address generated by the board. It must be aligned to 8 MB.

size is the size of the channel. It must be a multiple of 8 MB, and less than 256 MB.

flag contains the options for the channel. Options are 'ored' from the following:

VMEFG_WRTPOST access is write posted (default NO). It must be used with a BLT or MBLT channel.

VMEFG_READAHEAD access is read ahead (default NO). It must be used with a BLT or MBLT channel.

VMEFG_LEBE_*xx* defines the little/big endian conversion mode (default address coherency).

VMEFG_PCIIO access is from PCI-IO to VME (default is access from PCI-MEM to VME). There is no difference between PCI-MEM and PCI-IO.

AM code. One of the predefined **VMEFG_AM_***xxx* flags. The default is **VMEFG_AM_A32SDATA** (0x0D AM code).

This routine returns the CPU physical address on success and -1 on failure. The CPU physical address to be used by the Operating System or a user program can be mapped using the routine mmap(). This routine can be called from user space with the ioctl **VMEIOCTL_VME_CHAN_ALLOC.**

alma_vme_channel_free

int alma_vme_channel_free(unsigned long address)

This is used by the VME driver to free a PCI-VME channel allocated by alma_vme_channel_alloc().

address is the pointer returned by alma_vme_channel_alloc().

This routine returns 0 on success and SYSERR on failure. It can be called from user space with the ioctl **VMEIOCTL_VME_CHAN_FREE.**

alma_cpuadr_to_vmeadr

u32 alma_cpuadr_to_vmeadr(unsigned long address)

This returns the VME address that can be used to get at the given cpu physical address, or -1 if not visible from the VME.

User Space Mapping Service

ioctl(fd, **VMEIOCTL_PCI_CHAN_ALLOC**, &A)

does alma_pci_channel_alloc(A.pcichan.name, A.pcichan.vmeaddr, A.pcichan.pciaddr, A.pcichan.-size, A.pcichan.flag)

ioctl(fd, **VMEIOCTL_PCI_CHAN_FREE**, &A)

calls alma_pci_channel_free(A.index)

ioctl(fd, **VMEIOCTL_VME_CHAN_ALLOC**, &A)

returns alma_vme_channel_alloc(A.vmechan.name, A.vmechan.vmeaddr, A.vmechan.size, A.vmechan.flag)

ioctl(fd, **VMEIOCTL_VME_CHAN_FREE**, &A)

calls alma_vme_channel_free(A.addr)

For example (from MasterSgl.c).

```
char *devName = "mychan";
alma_ioctl.vmechan.name = devName; // Name
alma_ioctl.vmechan.vmeaddr = aBaseAddr; // VME Physical Address
alma_ioctl.vmechan.size = PCIVME_BLKSIZE; // As small as possible (8 MB)
```

```
alma_ioctl.vmechan.flag = VMEFG_AM_A32UDATA;
physAddr = ioctl(fd, VMEIOCTL_VME_CHAN_ALLOC, &alma_ioctl);
if(physAddr==-1){
  printf("vmeOpen: VME channel open / VMEIOCTL_VME_CHAN_ALLOC fail:
%s\n",devName);
  return (ERROR);
}
virtualAddr = (unsigned long)mmap(0, PCIVME_BLKSIZE, PROT_READ |
PROT_WRITE,
               MAP_SHARED, fd, (off_t)(physAddr&0xFFFFFFFF));
if(virtualAddr==-1){
  printf("vmeOpen: VME channel open: mmap fail: %s\n",devName);
  ioctl(fd, VMEIOCTL_VME_CHAN_FREE, &physAddr);
}
```

DMA Operations

The VME/PCI bridge includes two physical DMA channels that can copy data between PCI and VME buses. DMA bounce buffers are allocated if necessary, i.e. if the flags say the source or destination area is a virtual address.

Note that by default each DMA transfer is 4 kB (a page). Contact Kontron for other possibilities.

alma_user_dma

int alma_user_dma(u32 vmeaddr,unsigned long pciaddr,u32 size,u32 flags)

This executes a dma between the given vmeaddr and a pci, cpu physical or user virtual address, depending on the flags. It returns 0 on success.

vmeaddr is the VME physical address.

pciaddr is by default the DRAM logical address and DMA buffers are allocated. If the flag VMEFG_DMA_PADDR is set, it is the cpu physical DRAM address and DMA buffers are not allocated, but the dart or iommu is programmed and the address converted to a pci bus physical address. If the flag VMEFG_DMA_PCIADDR is set, it is the pci bus physical DRAM address and DMA buffers are not allocated, and no change is made to the address, no dart or iommu entries are made.

size is the size of the transfer in bytes. It is divided by 4 kB to get the number of DMA transfers.

flags contains the options for the channel. Options are 'ored' from the following:

VMEFG_DMA_VMEPCI the DMA is VME read and PCI write (default is PCI read VME write)

VMEFG_LEBE_*xx* define the little/big endian conversion mode (default is address coherency)

AM code. One of the predefined **VMEFG_AM_*xxx*** flags can be used. The default is VMEFG_AM_A32SDATA (0x0D AM code).

VMEFG_DMA_CHBKSIZ changes the number of VME cycles in each block (default is 8). Note that the maximum number of cycles in a block is 256 and that the number of cycles are numbered starting from 0 (0 = 1 cycle, 1 = 2 cycles, etc.). The number of cycles uses the bitmask 0x0000FF00, so the blocksize has to be shifted left 8 and then 'ored' with a flag. For example, to set 256 VME cycles in each block, 'or' ((256-1)<<8)|VMEFG_DMA_CHBKSIZ with the other flags. Usually performance is improved by increasing the blocksize.

This does as alma_user_dma() but uses 2ESST broadcast cycles. Flags should include VMEFG_AM_2ESST and VMEFG_XAM_A64BDCST. The select mask has a bit set for the geographic id of each remote board that should accept the broadcast (little-endian bits 1..21). See the section on 2ESST below.

User Space DMA Operations

ioctl(fd, **VMEIOCTL_DMA_START**, &A)

calls alma_user_dma(A.dma.vmeaddr, A.dma.data, A.dma.size, A.dma.flag).

ioctl(fd, **VMEIOCTL_DMA_START_MCAST**, &A)

calls alma_user_dma_mcast(A.mcast.vmeaddr, A.mcast.data, A.mcast.size, A.mcast.flag, A.mcast.select). This feature is not available on Alma2f.

For example:

```
dma_arg.vmeaddr = strtoul(argv[1], NULL, 0); // VME address
size = strtoul(argv[2], NULL, 0);
dma_arg.dmaaddr = malloc(size); // DMA buffer
if (dma_arg.dmaaddr == NULL) {
  fprintf(stderr," Error allocating DMA buffer\n");
  exit(1);
}
dma_arg.size = size; // size
if (argc <= 3) // Optional channel
  dma_arg.flag = VMEFG_AM_A32SDATA|VMEFG_DMA_VMEPCI;
else
  dma_arg.flag = strtoul(argv[3],NULL,0);
printf(" VME Addr: 0x%x, DMA buf: 0x%p, Size: 0x%x, Flags: 0x%x\n",
                dma_arg.vmeaddr,dma_arg.dmaaddr, dma_arg.size, dma_arg.-
flag);
err = ioctl(fd, VMEIOCTL_DMA_START, &dma_arg);
if (err != -1) {
  unsigned int *ptr = (unsigned int *) dma_arg.dmaaddr;
  for (i = 0; i < dma_arg.size/28; i++) {
    printf("[0x%p] %x %x %x %x %x %x %x\n",
    ptr, ptr[0], ptr[1], ptr[2],
    ptr[3], ptr[4], ptr[5], ptr[6]);
    ptr += 7;
  }
}
```

Contiguous Memory Allocation

The following routines allocate and free the physically contiguous memory needed for dma.

alma_phys_mem_alloc

int alma_phys_mem_alloc(struct alma_phys_mem_object *phys_mem)

This allocates a contiguous physical memory area. In the phys_mem parameter, only phys_mem->size needs to be set to the requested size, which can be up to 64 Mbytes. It returns 0 on success, -1 on failure. It can be called from user space with the ioctl **VMEIOCTL_PHYSICAL_MEM_ALLOC.**

alma_phys_mem_free

int alma_phys_mem_free(struct alma_phys_mem_object *phys_mem)

This frees a previously allocated contiguous physical memory area. In the phys_mem parameter, only phys_mem->cpu_phys_addr needs to be set to the address of the physical memory area. It returns 0 on success, -1 on failure. It can be called from user space with the ioctl **VMEIOCTL_PHYSICAL_MEM_FREE.**

User Space Contiguous Memory Allocation

ioctl(fd, **VMEIOCTL_PHYSICAL_MEM_ALLOC**, struct alma_phys_mem_object *phys_mem)

calls alma_phys_mem_alloc(phys_mem).

ioctl(fd, **VMEIOCTL_PHYSICAL_MEM_FREE**, struct alma_phys_mem_object *phys_mem)

calls alma_phys_mem_free(phys_mem).

For example (taken from the almavmechan tool):

```
mem_obj.size = ...;
rc = ioctl(fd,VMEIOCTL_PHYSICAL_MEM_ALLOC,&mem_obj);
printf("0x%x\n",mem_obj.cpu_phys_addr);
mem_obj.cpu_phys_addr = ...;
rc = ioctl(fd,VMEIOCTL_PHYSICAL_MEM_FREE,&mem_obj);
```

Locking

The VME bus can be explicitly locked before one or more accesses are done. It must always be paired with a free. These routines return 0 on success. On the VM605x board, the VME bus should be locked before issuing a simgle VME access concurrently with VME DMA accesses.

alma_get_vme

int alma_get_vme(void)

alma_free_vme

int alma_free_vme(void)

User Space Locking

ioctl(fd, **VMEIOCTL_GET_VME**, 0)

calls alma_get_vme()

ioctl(fd, **VMEIOCTL_FREE_VME**, 0)

calls alma_free_vme().

For example:

```
void fcop_withcapturevmebus(u32 *dst, u32 *src, u32 size){
  int ret = ioctl(fd,VMEIOCTL_GET_VME,0); // Capture the VME Bus
  if (ret==0) {
    fcop(dst,src,size); // Do the job
    ret = ioctl(fd,VMEIOCTL_FREE_VME,0); // Release the VME Bus
  }
}
```

VME/PCI Bridge Configuration

Some parameters can be changed in the VME/PCI bridge, such as request level, release mode, and arbitration.

alma_ctrl

int alma_ctrl(int command, int config)

If command is **ACTL_GET_CONFIG** the config is ignored, and the current configuration is returned. If command is **ACTL_SET_CONFIG,** the given config becomes the new VME/PCI bridge configuration, and 0 is returned. The config format is the following (bit 0 is the LSB):

request level is coded in the 3 LSB: ALMACFG_REQLEVELx for level x (from 0 to 3 inclusive). The default value is level 3.

fair mode is coded in bit 3 (ALMACFG_FAIRMODE). If active, the VME/PCI bridge does not request the VME if another VME device requests it on the same level.

bus release mode is coded in bits 4..6 (ALMACFG_BUSREL_MSK). Four modes are supported: ALMACFG_BUSREL_ROR (release on request), ALMACFG_BUSREL_RWD (release when done), ALMACFG_BUSREL_ROC (release on clear) and ALMACFG_BUSREL_RNE (release never). The default mode is ALMACFG_BUSREL_ROR.

fair mode timeout is coded in bit 7 (ALMACFG_FAIRTIMEOUT).

system controller is coded in bit 17 (ALMACFG_SYS_CTRL). It is READ ONLY. If active, the VME/PCI bridge is the VME system controller.

arbitration type is coded in bit 16 (ALMACFG_ROT_PRIO). If the VME/PCI bridge is the system controller, this bit sets the arbitration type: 0 fixed priority, 1 rotating priority.

timeout is coded in bits 8..15 (ALMACFG_TIMEOUTMSK). If the VME/PCI bridge is the system controller, these bits are the timeout value in microseconds for data transfer on the VMEbus. For example, if the timeout is 120 and if there is no response to a VME request after 120 microseconds, the VME/PCI bridge will generate a bus error on the VMEbus.

VME speed is coded in bit 20 and 21 with **ALMACFG_SPEED_***XXXX.* There are three modes: slow, medium and fast. It programs the device to different speeds for VME 2ESST cycles.

This routine can be called from user space with the ioctls **VMEIOCTL_SET_ALMACFG**

and **VMEIOCTL_GET_ALMACFG.** It is possible to get the VMEbus and release the VMEbus by software; alma_get_vme() gets the VMEbus and alma_free_vme() releases the VME bus. These routines can be called from user space with the ioctls **VMEIOCTL_GET_VME** and **VMEIOCTL_FREE_VME.**

User Space VME/PCI Bridge Configuration

ioctl(fd, **VMEIOCTL_SET_ALMACFG**, &A)

calls alma_ctrl(ACTL_SET_CONFIG,A.almacfg)

ioctl(fd, **VMEIOCTL_GET_ALMACFG**, 0)

returns alma_ctrl(ACTL_GET_CONFIG,0)

Read() and Write() Primitives

The VME driver read() and write() primitives offer direct user access to VMEbus space, using the AM code specified by the VMEIOCTL_SET_RWAM ioctl command. The DMA is used to do reads and writes without remapping of VME buffers in kernel or user space. /**dev**/**almavme** and any /**dev**/**vme***X* may be used to read/write on the VME. The usual lseek(2) system call can be used to change the vmeaddr.

User Space Read() and Write() Configuration

ioctl(fd, **VMEIOCTL_SET_RWAM**, &A)

sets the AM code to be used by the driver's read() and write() entry points from A.defam.

ioctl(fd, **VMEIOCTL_GET_RWAM**, &A)

returns in A.defam the current AM code used by the driver's read() and write() entry points.

For example:

```
vmeaddr = strtoul(argv[1],NULL,0); // VME address
size = strtoul(argv[2], NULL, 0);
buffer = malloc(size); // DMA buffer
if (buffer == NULL) {
   fprintf(stderr," Error allocating DMA buffer\n");
   exit(1);
}
if (argc <= 3) // Optional channel
```

```
    flags = VMEFG_AM_A32SDATA;
 else
    flags = strtoul(argv[3],NULL,0);
 printf(" VME Addr: 0x%x, DMA buf: 0x%p, Size: 0x%x, Flags: 0x%x\n",
                  vmeaddr, buffer, size, flags);
 err = ioctl(fd, VMEIOCTL_SET_RWAM, &flags); // Set the AM
 err = lseek(fd, vmeaddr, SEEK_SET); // Set vmeaddr offset
 err = read(fd, buffer, size); // Read data into buffer
 if (err != -1) {
   unsigned int *ptr = (unsigned int *) buffer;
   for (i = 0; i < size/28; i++) {
     printf("[0x%p] %x %x %x %x %x %x %x\n",
       ptr, ptr[0], ptr[1], ptr[2],
       ptr[3], ptr[4], ptr[5], ptr[6]);
     ptr += 7;
   }
 }
```

Interrupt Service Calls

The VME driver associates interrupt vectors with handlers by calling the routines request_vmeirq() and free_vmeirq(). The VME irq level may be masked/unmasked using alma_vmeintcontrol().

request_vmeirq

int request_vmeirq(unsigned int vector, void *handler)

Assign a handler to a specific VME vector. If the vector is between 8 and 15, it is for an addressed interrupt (see below). Otherwise, it is a VME irq vector from 0 to 255.

free_vmeirq

int free_vmeirq(unsigned int vector)

Release the specified vector.

alma_vmeintcontrol

void alma_vmeintcontrol(unsigned int mask, unsigned int ienable)

This masks or unmasks one or several interrupt levels.

mask is a 16 bit bitfield where each bit is associated with an interrupt level except bit 0. Bit n corresponds to the interrupt level n on the VMEbus.

ienable is 0 to mask and non-zero to unmask all the interrupt levels set in mask.

disable_vmeirq

int disable_vmeirq(u32 level)

This deprecated routine disables a single VME interrupt level, from 1 to 7.

enable_vmeirq

int enable_vmeirq(u32 level)

This deprecated routine enables a single VME interrupt level, from 1 to 7.

alma_vmeinterrupt

int alma_vmeinterrupt(unsigned char level, unsigned char vector)

This sends an interrupt on the VMEbus. Vector must be a multiple of 8. The interrupt vector generated is (vector+level).

User Space Interrupt Service

ioctl(fd, **VMEIOCTL_VMEINTR_GEN**, &A)

calls alma_vmeinterrupt(A.intr.level,A.intr.vector)

ioctl(fd, **VMEIOCTL_VMEINTR_CTRL**, &A)

calls alma_vmeintcontrol(A.intrctl.mask,A.intrctl.enable)

ioctl(fd, **VMEIOCTL_VMEIOINTSET**, &A)

connects interrupt A.vector to the VME driver, and returns an identifier that can be used with VMEIOCTL_INTRWAIT and VMEIOCTL_VMEIOINTCLR.

ioctl(fd, **VMEIOCTL_VMEIOINTCLR**, &A)

disconnects interrupt A.index from the VME driver.

ioctl(fd, **VMEIOCTL_INTRWAIT**, &A)

waits for an interrupt connected by VMEIOCTL_VMEIOINTSET. A.index is the identifier returned by VMEIOCTL_VMEIOINTSET.

For example, generating an interrupt:

```
lvl = strtol(argv[1], NULL, 0); // VME IRQ level
vect = strtol(argv[2], NULL, 0); // VME vector
printf("Level = 0x%lx, Vector = 0x%lx\n", lvl, vect);
alma_ioctl.intr.level = (u32) (lvl&0xFFFFFFFF);
alma_ioctl.intr.vector = (u32) (vect&0xFFFFFFFF);
err = ioctl(fd, VMEIOCTL_VMEINTR_GEN, &alma_ioctl);
```

For example, waiting for an interrupt:

```
lvl = strtol(argv[1], NULL, 0);
vect = strtol(argv[2], NULL, 0);
alma_arg.vector = (vect&0xFF); // Vector
printf("Init handling for interrupt level %d and vector 0x%x (0x%x)\n",
       lvl, vect, alma_arg.vector);
fflush(stdout);
Id = ioctl(fd, VMEIOCTL_VMEIOINTSET, &alma_arg);
// Enable the interrupt lvl
alma_arg.intrctl.mask = (1 << lvl); // VME IRQ level
alma_arg.intrctl.enable = 1; // Enable the IRQ
err = ioctl(fd, VMEIOCTL_VMEINTR_CTRL, &alma_arg);
printf("Wait for interrupt level %d (vector 0x%x) id=%d...\n", lvl, vect,
Id);
fflush(stdout);
alma_arg.index = Id; // Id in the user vector table
err = ioctl(fd, VMEIOCTL_INTRWAIT, &alma_arg);
printf("RECEIVED\n");
fflush(stdout);
// Disable the interrupt lvl
alma_arg.intrctl.mask = (1 << lvl); // VME IRQ level
alma_arg.intrctl.enable = 0; // Disable the IRQ
err = ioctl(fd, VMEIOCTL_VMEINTR_CTRL, &alma_arg);
printf("Disable VME IRQ level %d\n", lvl);
alma_arg.index = Id; // Id in the user vector table
err = ioctl(fd, VMEIOCTL_VMEIOINTCLR, &alma_arg);
```

Addressed Interrupts

There is another way to transmit an interrupt across the VMEbus using addressed interrupts. A byte write into a special register of the VME/PCI bridge will generate an interrupt on the board. A driver can choose to discriminate between eight different sources of addressed interrupts. The VME/PCI bridge registers can be accessed from the VMEbus in A16 mode. The A16 base address is setup by firmware (VMEID) on a 256 byte boundary, and extends for 256 bytes. To generate an addressed interrupt, write a byte at address (ALMAbase + 0xEE). The byte should have a value of (1<<x), where x is from 0 to 7 inclusive, chosen by the programmer. To receive addressed interrupts use request_vmeirq().

For example:

```
vmea16_addr = strtol(argv[3], NULL, 0); // VME A16 addr
printf("Adressed VME interrupt %ld remote A16:0x%lx\n", lvl-8,
vmea16_addr);
alma_ioctl.vmechan.name = "GENintr"; // Name
alma_ioctl.vmechan.vmeaddr =  0; // VME Physical Address
alma_ioctl.vmechan.size = PCIVME_BLKSIZE; // As small as possible (8 MB)
alma_ioctl.vmechan.flag = VMEFG_AM_A16S;
physaddr = ioctl(fd, VMEIOCTL_VME_CHAN_ALLOC, &alma_ioctl);
vaddr = (unsigned long) mmap(0, vmea16_addr + 0x100 /*size of A16
window*/,
  PROT_READ | PROT_WRITE, MAP_SHARED, fd, (off_t)(physaddr&0xFFFFFFFF));
if(vaddr==-1){
   printf("A16 VME channel open fail\n");
   ioctl(fd, VMEIOCTL_VME_CHAN_FREE, &physaddr);
   exit(1);
}
printf("VME_addr:0x%lx paddr:0x%lx vaddr:0x%lx\n", vmea16_addr, physaddr,
vaddr);
write_addr = (char*) (vaddr + vmea16_addr + 0xee);
write_val = (char)(1<<(lvl-8));
printf("WRITING 0x%x at 0x%p\n", write_val, write_addr);
*write_addr = write_val;
sync();
printf("READING 0x%x at 0x%p\n", *write_addr, write_addr);
sync();
munmap((void *)vaddr, 0x100); // size of A16 window
ioctl(fd, VMEIOCTL_VME_CHAN_FREE, &physaddr);
```

Further IOCTL() Commands
The following commands are also available:

ioctl(fd, **VMEIOCTL_VME_CHAN_GETIO**, struct pcivme_window *p)

copies the PCI-IO to VME channel descriptions into the third arg. It is N structures of type struct pcivme_window, where N is the number of PCI-IO to VME channels opened.

ioctl(fd, **VMEIOCTL_VME_CHAN_GETMEM**, struct pcivme_window *p)

copies the PCI-MEM to VME channel descriptions into the third arg. It is N structures of type struct pcivme_window, where N is the number of PCI-MEM to VME channels opened.

ioctl(fd, **VMEIOCTL_VME_CHAN_GETNB**, &A)

returns in A.vmechannb.mem the number of PCI-MEM to VME channels opened and in A.vmechannb.io the number of PCI-IO to VME channels opened.

ioctl(fd, **VMEIOCTL_PCI_CHAN_GET**, struct vmepci_window *p)

copies the VME to PCI channel descriptions into the third arg. It is 7 structures of type struct vmepci_window.

ioctl(fd, **VMEIOCTL_PCI_CHAN_GETEXT**, struct vmepci_window *p)

copies the VME to PCI channel descriptions into the third arg. It is MAXVMEPCIWIN structures of type struct vmepci_window.

ioctl(fd, **VMEIOCTL_SET_BOARDID**, int *ga)

set the GA (Geographical Address) for opening an A16S VME window pointing the the Alma bridge registers. Note that if the 'ga' value is equal to 0xff, the current GA of the board is not modified. Furthermore if 'ga' equal to -1 than this window is closed.

Sysfs Files

The driver provides files under the sysfs "vme" class directory, `/sys/class/vme/almavme.` These can be read or written.

/sys/class/vme/almavme/vme_bus_error
A read of this file returns as a string the number of VME bus errors seen by the driver. Any write will clear the counter.

2ESST
The Alma PCI/VME bridge implements 2ESST (Two Edge Source Synchronous Vme Transfer). This is an extension to the VME64 standard to allow the VME bus to operate at much greater speeds of nominally 160, 267 and 320 Mbytes/second, between boards supporting the extension.

2ESST transfers use a new address modifier (AM) code, and so are ignored by boards that do not support them. The supervisory/non-privileged and program/data sub-modes are replaced by a new extended address modifier (XAM). The data transmitter uses both edges of the clock to strobe data and does not wait for any acknowedgement (two edge source synchronous). 64 bits (8 bytes) of data are clocked on each edge. Transmissions must use an even number of edges, so the minumum unit of transfer is 16 bytes, though it is possible to flag the last cycle as ignored. It is not optimal to transfer only 16 bytes of data. A maximum of 128 beats of 2 edges of 8 bytes, ie 2048 bytes is allowed per transfer.

The initiator specifies in a three-part setup phase the maximum transfer rate. The slave can terminate the transfer with an error if it is unable to perform at this rate. It can also suspend the transfer if it does not currently have the resources, and the transfer will be retried later. The address transferred during setup phase does not specify the low 4 bits, so addresses are always aligned on 16 bytes.

There are physical limitations imposed by the distance between boards, the length of the backplane, and the effect of other boards in the rack, including a broadcast selecting many boards or not, so 2ESST transfers might work in one direction, and not in the reverse direction between the same two boards. For example, the maximum speed of 320 Mbytes/second might only be possible in a rack of 6 slots, with only 4 boards in place.

The Alma bridge provides 8 extra VME slave channels, for incoming VME accesses, allocated by the usual alma_pci_channel_alloc() routine when the VMEFG_AM_2ESST flag is used. These can be used for 2ESST accesses or ordinary accesses. (The first 8 VME channels cannot be used for 2ESST).

The maximum 2ESST rate that a channel will accept can be specified. Higher rates will cause the transfer to fail. The XAM must also be given and must match that used for the dma transfer.

2ESST transfers can only be generated by dma using the usual alma_queue_dmalist() routine, or alma_user_dma() its broadcast version alma_user_dma_mcast(). They must be targeted to a 2ESST window with the same AM, XAM, and the same or a higher rate.

The rate chosen for the dma can be throttled globally to a lower value by ioctl VMEIOCTL_SET_ALMACFG or kernel service alma_ctrl(), or the almavmechan(1) option -setalmastate.

Addresses for 2ESST dma must be aligned on 16 bytes.

The 2ESST protocol also provides for a broadcast transfer, where one board can do a single write that is received simultaneously by several boards. The boards to receive the broadcast are selected during the address phase by their geographic id. Broadcast AM therefore need to provide a bitmask selecting the remote boards that are to receive the write. Each board has a geographic id which is the slot num-

ber in which it is positioned. The bitmask is an ”or” of 1<<n where n is the geographic id of a targeted board.

Select mask decoding by the slave is not currently supported by the Alma device, and it will accept all broadcasts, even if it is not selected. One workaround is to setup several separate slave A64 windows targetting the same physical memory, and only enable the appropriate ones for a given select mask.

The almavmechan(1) command has an option –dmabdcst to generate 2ESST broadcasts.

EXAMPLES

Refer also to the example files Dma.c, Dma2.c, GEN_intr.c, RECV_intr.c, MasterSgl.c in /usr/share/vmetoolkit/examples.

RESTRICTIONS

Currently, the driver and ioctls do not provide meaningful errno values on error.

SEE ALSO

almavmechan(1), mbm3k(1)
Alma2f User Manual CI.DT.A00

FILES

/dev/almavme
/dev/vme*X*

12.2 almavmechan

NAME

almavmechan – VME bus utility for almavme driver

SYNOPSIS

–vmememalloc *name vmeaddr size flag*
–vmeioalloc *name vmeaddr size flags*
–vmefree *cpu_addr*
–pcialloc *name vmeadr cpuadr size fg*
–pcifree *id*
–printall
–vmeioprint
–vmememprint
–pciprint
–vmeintrgen *level vector*
–vmeintrctrl *mask on/off*
–vmeiointset *vector*
–vmeiointclr *vector*
–vmeintwait *id*
–setamrw *value*
–vmeedit
–getalmastate
–setalmastate *val*
–editalmastate
–dmastart *vmeadr dramadr size fg*
–dmabdcst *vmeadr pwbadr size fg sel*
–dmawait *id*
–physmemalloc *size*
–physmemfree *phys_addr*

DESCRIPTION

almavmechan can create and delete PCI to VME or VME to PCI channels using the ALMA PCI to VME bridge. It prints the current channels opened. It can also read and modify the bridge's state. This utility uses the ioctl's of the VME bus driver described in almavme(4).

OPTIONS

–vmememalloc *name vmeaddr size flag*

allocate a PCIMEM to VME channel

–vmeioalloc *name vmeaddr size flags*

allocate a PCIIO to VME channel

–vmefree *cpu_addr*

free a PCI to VME channel

–pcialloc *name vmeadr cpuadr size fg*

allocate a VME to PCI channel. *fg* parameter defines the VME standard AM as ALMA bridge register format. The following table shows the *fg* values for the different VME standard AM value:

A32 USER MBLT (AM 0x08) fg = 0x00201013
A32 USER DATA (AM 0x09) fg = 0x00202013
A32 USER PROG (AM 0x0A) fg = 0x00204013

A32 USER BLT (AM 0x0B) fg = 0x00208013
A32 SUPER USER MBLT (AM 0x0C) fg = 0x00210013
A32 SUPER USER DATA (AM 0x0D) fg = 0x00220013
A32 SUPER PROG (AM 0x0E) fg = 0x00240013
A32 SUPER BLT (AM 0x0F) fg = 0x00280013
A24 USER MBLT (AM 0x38) fg = 0x08001013
A24 USER DATA (AM 0x39) fg = 0x08002013
A24 USER PROG (AM 0x3A) fg = 0x08004013
A24 USER BLT (AM 0x3B) fg = 0x08008013
A24 SUPER USER MBLT (AM 0x3C) fg = 0x08010013
A24 SUPER USER DATA (AM 0x3D) fg = 0x08020013
A24 SUPER USER PROG (AM 0x3E) fg = 0x08040013
A24 SUPER USER BLT (AM 0x3F) fg = 0x08080013
A16 USER (AM 0x29) fg = 0x02002013
A16 SUPER (AM 0x2D) fg = 0x02020013
A16 LCK (AM 0x2C) fg = 0x02010013

A16 USER, A16 SUPER, A16 LCK can not be used to initialize a VME to PCI memory window due to the current driver window size definition that must be higher or equal to 1MB.
A24 USER MBLT, A24 SUPER USER MBLT are identify as AM that present some transfer error.
For these cases please contact the Kontron support.

–pcifree *id*

free a VME to PCI channel

–printall

print all allocated channels.

–vmeioprint

print all PCIIO to VME channels.

–vmememprint

print all PCIMEM to VME channels.

–pciprint

print all VME to PCI channels.

–vmeintrgen *level vector*

send a VME interrupt.

–vmeintrctrl *mask on/off*

mask or unmask VME interrupts.

–vmeiointset *vector*

connect a VME interrupt.

–vmeiointclr *vector*

disconnect a VME interrupt.

–vmeintwait *id*

wait for a VME interrupt.

–setamrw *value*

set AM (address modifier) to use for accesses through standard read() and write() routines.

–vmeedit

use the VME data editor to do direct VME bus read/writes (with VME addresses).

–getalmastate

print the current Alma state.

–setalmastate *val*

set the Alma state.

–editalmastate

change the Alma state interactively.

–dmastart *vmeadr dramadr size fg*

start a DMA between VME and PCI.

–dmabdcst *vmeadr pwbadr size fg sel*

start a broadcast DMA.

–dmawait *id*

wait for the end of a DMA.

–physmemalloc *size*

allocate a contiguous physical memory area.

–physmemfree *phys_addr*

free a contiguous physical memory area.

EXAMPLES

To create a PCI–MEM to VME channel at VME address 0x10000000, size of 8 MB with read ahead and write posting modes:

```
# almavmechan –vmememalloc GREAT 0x10000000 0x8000000 0x3
  0xd2000000
```

The result 0xd2000000 is the CPU physical address needed to access VME address 0x10000000.

To print all opened channels and the state of the ALMA bridge:

```
# almavmechan –printall


------------------------------------------------------------------------
                      No PCIIO to VME CHANNEL
------------------------------------------------------------------------
------------------------------------------------------------------------
                      PCIMEM to VME CHANNELS
------------------------------------------------------------------------
  Name       CPUaddr     PCIaddr     VMEaddr  size(Mb)     AM     Conv  WP  RH
------------------------------------------------------------------------
GREAT         0xd2000000 0x52000000 0x10000000  0128     A32SDATA ADDR  Yes
Yes
------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------
                      No VME to PCI CHANNEL
------------------------------------------------------------------------------
Alma state:
        System controller: fixed priority.
                           timeout 100
        Fair mode timeout.
        Not Fair mode.
        VME bus release mode :  ROR
        Request level : BR3
        Dead lock logic ON.
        2ESST rate  FAST
```

SEE ALSO

almavme(4), mbm3k(1)

12.3 VME IRQ Response Time Improvement

By default the Linux kernel uses the C-State mode in the IDLE loop.

This feature impacts the performance of waking up a IDLE CPU.

The VME IRQ response time can have some relationship with the IDLE CPU mode in the Linux kernel.

To avoid a lack of VME IRQ response time, the '`idle=poll`' can be given to the linux kernel command line.

13 / RC Boards

13.1 How to Manage the Lack of RTC Battery

Some hardware constraints may prevent the use of the RTC's battery on the boards. This implies to workaround or disable some standard behaviour of Fedora services. The very first one is "**fsck**" which is run at boot time. **fsck** checks that the date of the last mount of the checked partition is prior to the current date. If not, the boot is stopped in a maintenance mode. This could happen after with the power off and without the RTC's battery. To workaround this, simply create a file named **/etc/e2fsck.conf** with the following content:

```
[options]
broken_system_clock=1
```

The same file has to be added to the **initramfs** too. To do this, simply run the command:

```
[root]# dracut --force -I /etc/e2fsck.conf
```

Without the right date for the system, some other services and functions may present problems. For example, the command "**make**" warns if there is some previous build detected in the future and this could lead to bad recompilation. If the network is available, it is recommended to setup the service **ntpd** to adjust the system date with a date server.

13.2 External Devices Connection

One impact of the RC board is the missing of front panel.

As a consequence, devices requiring access to the system such as:

- Serial line console.
- USB mouse
- USB keyboard
- Display monitor (through the HDMI port)
- Ethernet

will be plugged in from the rear side through a Rear Transition Module (RTM) Paddle board.

For example, the Kontron PB-VX3-4xx is a 3U VPX Rear Transition Module (compliant with the definition of the Rear Transition Module on VPX Standard -VITA 46.10) provides rear I/O peripherals connectivity for Kontron VX305x Single Board Computers. It is required to perform a graphical linux installation stage on a VX305x-RC board.

- USB mouse and USB keyboard through a USB HUB.
- The display monitor (through the HDMI port).

After turning on the system, you should notice all devices have been recognized correctly allowing to take control of the graphical environment.

13.3 RC Specifications

The RC version of the boards is designed to work in different environmental constraints and for different levels of temperature and power dissipation.

This may imply specific setups (cpu frequencies, hardware parameters,...) to guarantee the specified behavior.

They are described in the “Hardware User's Guide” of each board. Please check this document for your board and for the required environment.

14 / Errata

14.1 VME Timeout set by default in the Driver at Infinite - CRP 4286 (VME Boards only)

Description: The VME timeout is defaulty set at infinite. The module parameter '**alma_vmetim**' that corresponds to VME timeout is set at 0. It means infinite VME timeout. The Inifnite VME timeout driver default is to preserve the Kontron Legacy VME. This default parameter value (at 0) can hang the VME bus in some specific use cases.

Workaround: To avoid VME bus issue, we advice the user to append these different **alma_vme** module parameters to the Linux bootline as follows:
alma_vme.alma_vmetim=255 alma_vme.alma_pcitim=1

14.2 VME A24 USER_MBLT and SUPER_USER_MBLT in DMA Mode Accesses does not work - CRP 4287 (VME Boards only)

Description: All access to VME A24 USER_MBLT (AM=0x38) and SUPER_USER_MBLT (AM=0x3c) in DMA mode failed

Workaround: None

14.3 VME IRQ Response Time - CRP 4288 (VME Boards only)

Description: VME IRQ response time can have less performance as expected. This problem is mainly due to CPU idle mode on Intel Architecture.

Workaround: Refer to section 12.3 “12.3” page 67.

14.4 Board restarting after a shutdown/halt - KDP #26779/#26008 (VX305x and VX6090 only)

Description: When the system is shutdown by the user (**Shutdown** from desktop menu, or command "**halt**", or command "**shutdown -h now**"), the system initiates a shutdown but ends with a board reset and restart. This is a CPLD issue, not a Linux issue. See Hardware Release Notes for a possible fix.

Workaround: Use command "**shutdown -H now**" (**-H** instead of **-h**).

14.5 Incorrect front Ethernet interface name - KDP #32175 (VX6060/VX3030/VX3035 only)

Description: With KVX Live ID17228 the front Ethernet interface is named **enp0s25** instead of **em0**, because of a missing entry for this chip in **/usr/share/kvx_bsp/getethname.sh**.

Will be fixed in future release.

Workaround: Edit **/usr/share/kvx_bsp/getethname.sh** to have:

```
VX6060 | VX3030 | VX3035)
    IF_LIST="
8086:10ea.0 1 em0 # 82577LM on VX6060/VX3030 front panel
8086:1502.0 1 em0 # 82579LM on VX3035 front panel
8086:1510.0 1 em1
8086:1510.1 1 em2
8086:1510.2 1 em3
8086:1510.3 1 em4
8086:150f.0 1 em1
8086:150f.1 1 em2
8086:150f.2 1 em3
8086:150f.3 1 em4
8086:150e.0 1 em1
8086:150e.1 1 em2
8086:150e.2 1 em3
8086:150e.3 1 em4
"
    ;;
```

14.6 SM750 driver blinking mouse cursor on graphics activity - KDP #32176

Description: With KVX Live ID17228 and an SM750 graphics module, the mouse coursor is blinking when some activity occurs on the screen.

A fix is planned in next release (updated SiliconMotion driver > v4.1.14).

Workaround: None

14.7 SM750 driver 2D graphics acceleration disabled - KDP #32177

Description: With KVX Live ID17228 and an SM750 graphics module, the EXA acceleration (2D) is disabled (see **/usr/share/sm750module/sm750.xorg.conf**).

With the acceleration enabled the graphics is very slow (scrolling up in a terminal is so slow that the scrollbar does not follow the mouse and is several seconds late).

Workaround: None

About Kontron

Kontron, a global leader in embedded computing technology and trusted advisor in IoT, works closely with its customers, allowing them to focus on their core competencies by offering a complete and integrated portfolio of hardware, software and services designed to help them make the most of their applications.

With a significant percentage of employees in research and development, Kontron creates many of the standards that drive the world's embedded computing platforms; bringing to life numerous technologies and applications that touch millions of lives. The result is an accelerated time-to-market, reduced total-cost-of-ownership, product longevity and the best possible overall application with leading-edge, highest reliability embedded technology

Kontron is a listed company. Its shares are traded in the Prime Standard segment of the Frankfurt Stock Exchange and on other exchanges under the symbol "KBC".
For more information, please visit: **www.kontron.com**

CORPORATE OFFICES

FRANCE

150, rue Marcelin Berthelot
ZI de Toulon-Est - BP 244
83078 Toulon Cedex 9 - France
Tel: +33 4 98 16 34 00
Fax: +33 4 98 16 34 01
sales.KFR@kontron.com

HEAD OFFICE

Lise-Meitner-Str. 3-5
86156 Augsburg
Germany
Tel.: + 49 821 4086-0
Fax: + 49 821 4086-111
info@kontron.com

NORTH AMERICA

9477 Waples Street, Suite 150
San Diego, CA 92121
USA
Tel.: + 1 888 294 4558
Fax: + 1 858 677 0898
info@us.kontron.com

ASIA PACIFIC

1~2F, 10 Bldg, Nº 8 Liangshuihe 2nd Str.
Economical & Techno. Develop. Zone,
Beijing, 100176, P.R. China
Tel.: + 86 10 63751188
Fax: + 86 10 83682438
info@kontron.cn