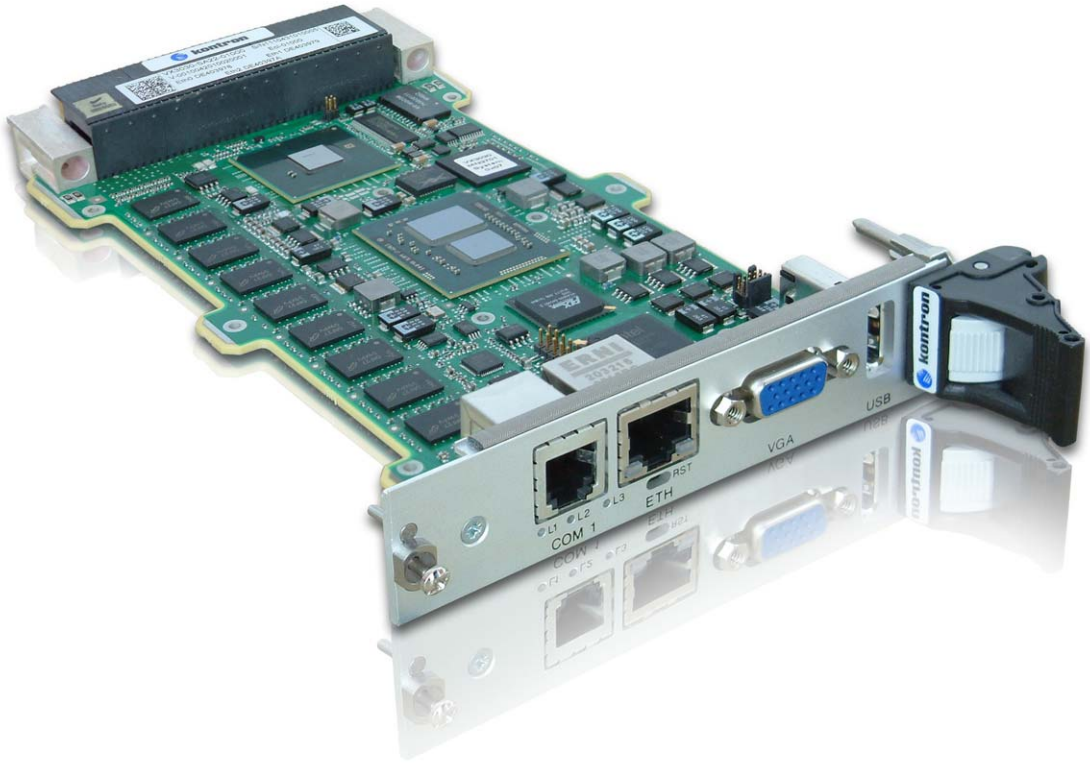
» VX3030 «



Release Notes VxWorks 6.8 on VX3030

SD.DT.G04-0e - March 2012

Revision History

Publication Title:	VxWorks 6.8 on VX3030	
Doc. ID:	SD.DT.G04-0e	
Rev.	Brief Description of Changes	Date of Issue
0e	Initial Version	March 2012

Proprietary Note

This document contains information proprietary to Kontron. It may not be copied or transmitted by any means, disclosed to others, or stored in any retrieval system or media without the prior written consent of Kontron or one of its authorized agents.

The information contained in this document is, to the best of our knowledge, entirely correct. However, Kontron cannot accept liability for any inaccuracies or the consequences thereof, or for any liability arising from the use or application of any circuit, product, or example shown in this document.

Kontron reserves the right to change, modify, or improve this document or the product described herein, as seen fit by Kontron without further notice.

Trademarks

This document may include names, company logos and trademarks, which are registered trademarks and, therefore, proprietary to their respective owners.

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

Conventions

This guide uses several types of notice: Note, Caution, ESD.

Note: this notice calls attention to important features or instructions.


Caution: this notice alert you to system damage, loss of data, or risk of personal injury.


ESD: This banner indicates an Electrostatic Sensitive Device.

All numbers are expressed in decimal, except addresses and memory or register data, which are expressed in hexadecimal. The prefix `0x' shows a hexadecimal number, following the `C' programming language convention.

The multipliers `k', `M' and `G' have their conventional scientific and engineering meanings of $*10^3$, $*10^6$ and $*10^9$ respectively. The only exception to this is in the description of the size of memory areas, when `K', `M' and `G' mean $*2^{10}$, $*2^{20}$ and $*2^{30}$ respectively.

When describing transfer rates, `k' `M' and `G' mean $*10^3$, $*10^6$ and $*10^9$ *not $*2^{10}$ $*2^{20}$ and $*2^{30}$.*

In PowerPC terminology, multiple bit fields are numbered from 0 to n, where 0 is the MSB and n is the LSB. PCI and CompactPCI terminology follows the more familiar convention that bit 0 is the LSB and n is the MSB.

Signal names ending with an asterisk (*) or a hash (#) denote active low signals; all other signals are active high.

Signal names follow the PICMG 2.0 R3.0 CompactPCI Specification and the PCI Local Bus 2.3 Specification.

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


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


Caution, Electric Shock!
Before installing a not hot-swappable Kontron product into a system always ensure that your mains power is switched off. This applies also to the installation of piggybacks. Serious electrical shock hazards can exist during all installation, repair and maintenance operations with this product. Therefore, always unplug the power cable and any other cables which provide external voltages before performing work.

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 board 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 board.

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

In performing all necessary installation and application operations, please follow only 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 board, please re-pack it as nearly as possible in the manner in which it was delivered.

Special care is necessary when handling or unpacking the product. Please consult the special handling and unpacking instruction on the previous page of this manual.

vx3030

NAME

vx3030 - Kontron VX3030

INTRODUCTION

This document is the VX3030 BSP User's Guide. It gives the installation procedure for the VX3030 vxWorks 6.8 BSP complemented with the vxWorks 6.8/Workbench 3.2 patch level required to support this VX3030 BSP.

This file also contains information on the board BIOS configuration, BSP general information, used device drivers and finally limitations and BSP evolutions compared to the previous versions.

The document contains the following first level chapters:

VxWorks REQUIREMENTS

To support VX3030 BSP, VxWorks 6.8 must be first updated to 6.8.2 and the following WindRiver patches (delivered with the BSP) must be also installed :

VxWorks6.8PatchIbexPeak_AHCIICH_WIND00269093_20110427.zip
VxWorks6.8PatchOsageIOApicDrive_20110822_022114.zip
VxWorks6.8PatchWIND00250729_FS_20110221_221123.zip
VxWorks6.8PatchWIND00264220_20110420_010822.zip

BIOS REQUIREMENT

The BIOS ID of the VX3030 must be greater or equal to ID11328

First of all restore the default values. To proceed, enter in the BIOS by pressing F2 or DEL during the boot process of the VX3030.

* F3 -> Optimized Defaults

After this is done and the Optimized default BIOS parameters entered then do the following:

* Chipset -> South Bridge Configuration -> SATA Configuration -> SATA Controler(s) [Enabled]
* Chipset -> South Bridge Configuration -> SATA Configuration -> SATA Mode Selection; [AHCI]
* Advanced -> CPU Configuration -> Hyper-Threading [Disabled]
* Kontron -> CPU Configuration [Select the maximum CPU frequency or the CPU frequency desired] or let the default value : 2000Mhz

Once the setting is done save the parameters for the next boot :

* Save & exit -> Save changes and Reset

JUMPERS & SWITCHS

Before using a board with VxWorks, verify that the board runs in factory configuration by using vendor-supplied ROMs and jumper settings and by checking the RS-232 connection. Refer to VX3030 3U VPX SBC User's Guide for a detailed description of the VX3030 board.

BSP INSTALLATION GUIDE

1. Install vxWorks 6.8 / Workbench 3.2

Get and install the DVDROM of the vxWorks 6.8 release. DVD-R147824.1-1-00 : VxWorks-6.8 and VxWorks Platform-3.8

Remark: In the "Installer" Window, when asked to "Choose on line update setting",
Remove selection for the 2 following items:
- check for and apply latest updates for this installer
- check online for the latest Wind River product updates

2. Install Update Pack 2

Install the WindRiver Update pack 2 for vxWorks 6.8.

The update pack 2 vxWorks 6.8 from WindRiver is present in the BSP DVDROM in the following folder :

updatePack2

This patch comes from

DVD-R147824.1-4-01 : Update pack 2

Uncompress the archive file deliverd by WindRiver (see the WINDRIVER support web site : support.windriver.com) that corresponds to DVD-R147824.1-4-01. Go to the DVD-R147824.1-4-01

folder and launch the appropriate shell script for the host machine.

For Linux :

```
./setup_linux
```

For Windows launch :

```
setup.exe
```

For Solaris :

```
./setup_solaris
```

Install the update pack 2 release

3. Install the VX3030 BSP

The BSP of the VX3030 is located in VXW68-BSP-VX3030-xxxxx folder. Where xxxxx identify the BSP ID.

Files present at the top of the VXW68-BSP-VX3030-xxxxx folder :

bootrom.bin: bootrom to put in a SATA or USB disk
bootrom.pxe: bootrom that is able to be launched through PXE protocol
INSTALL.txt: File explaining what the current release support
vximages : VxWorks images folder compiled for UP,SMP mode with GNU and DIAB compiler
VXW68-BSP : contains the vxWorks BSP files and docs
updatePack2: Folder that contains the update pack 2 for vxWorks 6.8 from WindRiver
grub4dos : Folder that contains useful file for grub4dos
vx3030.pdf : Describes the BSP release for vxWorks 6.8 for vx3030
VXW68-BSP-VX3030-xxxxxx.zip: BSP zip file ID xxxxxx to be used for installation

First **set the environment variables** by typing under the WindRiver Installation folder:

Linux OS :

```
./wrenv.linux -p vxworks-6.8
```

Windows OS :

```
wrenv.exe -p vxworks-6.8
```

Go to the DVDROM directory.

Unzip the file **VXW68-BSP-VX3030-xxxxx.zip** in the home vxWorks 6.8 directory as follow :

under **linux OS** :

```
cd $WIND_HOME
unzip /mnt/cdrom/VXW68-BSP-VX3030-xxxxx.zip -d .
```

under **Windows OS** (if **WIND_HOME**=C:\WindRiver and D: is the DVDROM) :

unzip the D:\VXW68-BSP-VX3030-xxxxx.zip in C:\WindRiver

During the unzipping process, in case the following message appears :

```
./vxworks-6.8/target/h/make/rules.bsp? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
```

Type A to replace existing files.

4. Install all the patches (located in the BSP os.patches folder)

Copy patches from :

Linux OS : $**WIND_HOME**/vxworks-6.8/target/config/VX3030/os.patches/*.* to $**WIND_HOME**/updates

Windows OS : %**WIND_HOME**%/vxworks-6.8/target/config/VX3030/os.patches/ to %**WIND_HOME**%/updates

Unzip all files in **updates** folder

Launch the WindRiver Installer :

Linux OS : $**WIND_HOME**/maintenance/wrInstaller/x86-linux2/wrInstaller

Windows OS : go the the %**WIND_HOME**%/maintenance/wrInstaller/x86-win32 and launch wrInstaller.exe by clicking on the icon.

Do not update the Installer itself so skip the first step if an Installer update is proposed

Select "patch" in the menu to install all the patches (4 patches must be installed).

5. Recompile libs

Once the patches are installed, the platform needs to be regenerated for the target.

Rename/Recompile an existing file :

-- Under Linux OS --

```
mv $WIND_HOME/vxworks-6.8/target/h/hwif/vxbus/vxbMethodDecl.h
 $WIND_HOME/vxworks-6.8/target/h/hwif/vxbus/vxbMethodDecl.h.orig
```

```
cd $WIND_HOME/vxworks-6.8/target/src/hwif/methods

make vxbMethodDecl.h
```

The following commands must be performed after each BSP installation to add all new drivers registration:

```
mv $WIND_HOME/vxworks-6.8/target/config/all/vxbUsrCmdLine.c
$WIND_HOME/vxworks-6.8/target/config/all/vxbUsrCmdLine.c.orig

cd $WIND_HOME/vxworks-6.8/target/config/comps/src/hwif

make vxbUsrCmdLine.c
```

-- Under Windows OS --

```
mv %WIND_HOME%/vxworks-6.8/target/h/hwif/vxbus/vxbMethodDecl.h
%WIND_HOME%/vxworks-6.8/target/h/hwif/vxbus/vxbMethodDecl.h.orig

cd %WIND_HOME%/vxworks-6.8/target/src/hwif/methods

make vxbMethodDecl.h
```

The following commands must be performed after each BSP installation to add all new drivers registration:

```
mv %WIND_HOME%/vxworks-6.8/target/config/all/vxbUsrCmdLine.c
%WIND_HOME%/vxworks-6.8/target/config/all/vxbUsrCmdLine.c.orig

cd %WIND_HOME%/vxworks-6.8/target/config/comps/src/hwif

make vxbUsrCmdLine.c
```

Go to :

-- Under **Linux OS** --

```
cd $WIND_HOME/vxworks-6.8/target/src

make CPU=PENTIUM4 TOOL=gnu rclean
make CPU=PENTIUM4 TOOL=diab rclean
make CPU=PENTIUM4 TOOL=gnu VXBUILD=SMP rclean
make CPU=PENTIUM4 TOOL=diab VXBUILD=SMP rclean
make CPU=PENTIUM4 TOOL=gnu
make CPU=PENTIUM4 TOOL=diab
make CPU=PENTIUM4 TOOL=gnu VXBUILD=SMP
make CPU=PENTIUM4 TOOL=diab VXBUILD=SMP
```

-- Under **Windows OS** --

```
cd %WIND_HOME%/vxworks-6.8/target/src

make CPU=PENTIUM4 TOOL=gnu rclean
```

```
make CPU=PENTIUM4 TOOL=diab rclean
make CPU=PENTIUM4 TOOL=gnu VXBUILD=SMP rclean
make CPU=PENTIUM4 TOOL=diab VXBUILD=SMP rclean
make CPU=PENTIUM4 TOOL=gnu
make CPU=PENTIUM4 TOOL=diab
make CPU=PENTIUM4 TOOL=gnu VXBUILD=SMP
make CPU=PENTIUM4 TOOL=diab VXBUILD=SMP
```

6. Compile VX3030 kernel

The installation procedure is ended. The VX3030 kernel can then be compiled by command or the Workbench Project can be launched to configure and compile a new image kernel.

To compile by command, do not forget the environment variable initialization:

Linux OS : ./wrenv.linux -p vxworks-6.8 **Windows OS** : wrenv.exe -p vxworks-6.8

```
cd vxworks-6.8/target/config/vx3030
make vxWorks
```

or for standalone kernel (advised)

```
make vxWorks.st
```

or if the SMP (Symmetric Multiprocessing Support) feature is expected

```
make vxWorks VXBUILD=SMP
```

BOOTING VxWorks ON VX3030

When the board is powered on, the system BIOS code loads and executes the bootstrap loader. The bootstrap loader is written in 8088 16-bit assembly language. The BIOS obtains the bootstrap loader from the boot sector, which may be in one of several locations: an USB 2.0 key, a hard disk, or some other alternatives. When the BIOS finds the bootstrap loader, it transfers execution to it. The bootstrap loader finds the bootrom.sys file, loads it into memory, and transfers execution to romInit.

1. Booting VxWorks via PXE

It is possible to boot VxWorks via the network using PXE (the Preboot eXecution Environment). PXE support is provided by the BIOS. The advantage of using PXE is that no floppy or hard disk is required (this is important given that support for floppy disk dives is becoming increasingly rare with newer PC systems).

Configure BIOS Setup for PXE boot:

* Chipset -> South Bridge Configuration -> Ibexpeak options -> PXE ROM: [Enabled]

F4: Save Changes and Exit, then go to BIOS setup again and set boot device (PXE: Boot -> Boot Option #1: [IBA GE Slot ...])

In addition to the regular bootrom.bin loader, a bootrom.pxe loader is provided for PXE booting VxWorks. The bootrom.pxe loader is virtually identical to the normal bootrom.bin, except that it has an extra block of padding prepended to it. When booting from disk, the vxld.bin loader is normally loaded by the BIOS at the address 0x7C00. It then in turn loads the bootrom at the address 0x8000 and starts running. By contrast, when booting with PXE, the bootrom is loaded directly at the address 0x7C00 by the PXE ROM. A leading block of 1024 NOP instructions is therefore used to force the **romInit( )** entry point to fall at the address 0x8000, just as it would do if it had been loaded from a disk. Once loaded and running, bootrom.pxe is able to load VxWorks images via the network just as the normal bootrom.bin does.

PXE booting requires the use of a DHCP and TFTP server. When the PXE ROM starts up, it will use DHCP to discover an IP address and the path for the bootrom file. Generally, the PXE ROM assumes that the DHCP and TFTP servers will be running on the same machine. The way the DHCP and TFTP servers are configured will depend on what software is used. DHCP and TFTP supports are available for most *NIX/Linux/*BSD systems, as well as for Microsoft Windows (R). The following sample of setup instructions explains how to configure the open source ISC DHCP server running on a FreeBSD system.

Assume that your client machine is on the 10.0.0.0/32 network, and that your DHCP/TFTP server's IP address is 10.0.0.1. When the PXE client is started up, the PXE ROM should display the station address of the ethernet port. We will assume the address is 00:00:E8:01:02:03, and that we want to assign the target an IP address of 10.0.0.3. Given this, we need to create a dhcpd.conf file as follows:

```
# dhcpd.conf
#
# Sample PXE boot configuration file for ISC dhcpd
#

ddns-update-style none;

subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.3 10.0.0.3;
  deny unknown-clients;
  host vxWorks_pentium_target {
    hardware ethernet 00:00:e8:01:02:03;
  }
  option routers 10.0.0.1;
  option broadcast-address 10.0.0.255;
  filename "/bootrom.pxe";
  default-lease-time 600;
  max-lease-time 7200;
}
```

Start the DHCP server with the following command, as superuser:

```
# dhcpd -cf /path/to/dhcpd.conf
```

Next, edit the /etc/inetd.conf file to enable the TFTP service, and insure inetd is started. The enable inetd, edit the /etc/rc.conf file and add a line that says **inetd_enable="YES"** then run the following commands:

```
# sh /etc/rc.d/inetd stop
# sh /etc/rc.d/inetd start
```

Finally, create a /tftpboot directory, and place a copy of the bootrom.pxe loader here.

Once this setup has been done, reboot the PXE client system. It should display an output similar to the following:

```
CLIENT MAC ADDR: 00 00 E8 01 02 03  GUID: xxxxxxxx xxxx xxxx xxxx xxxxxxxxxxxx
CLIENT IP: 10.0.0.3  MASK: 255.255.255.0  DHCP IP: 10.0.0.1
GATEWAY IP: 10.0.0.1
TFTP...
```

Right after this output appears, the VxWorks bootrom should start up and the bootrom startup banner should be displayed. At this point, you can enter the desired VxWorks boot parameters and use the bootrom to load a VxWorks image.

Important notes:

- PXE is a 16-bit real mode environment. Consequently, boot applications downloaded via PXE are limited in that they must be able to start up in 16-bit real mode, and they must be less than 640K in size. The VxWorks bootrom.bin image meets these requirements, however standalone vxWorks images might not.

- The PXE environment also provides a universal API for accessing the underlying ethernet device, however VxWorks has no support for this feature. This means that while you can download the VxWorks bootrom via any PXE-enabled ethernet device, a native driver for the ethernet device must be available in VxWorks in order for the bootrom to subsequently download a VxWorks image (or for the downloaded VxWorks image to be able to use the ethernet port).

- If no floppy drive is present in the system, the bootrom will not be able to save the bootline parameters anywhere (unless alternate non-volatile storage is provided).

2. Make bootrom for bootable disk (USB, SATA)

This section explains how to build a bootrom image including USB,PATA and/or SATA boot features.

If you want to boot vxWorks image from usb key, enable **INCLUDE_USB** in **config.h**, then change **DEFAULT_BOOT_LINE** to **DEFAULT_BOOT_LINE_USB**,

If you want to boot vxWorks image from CF card or PATA, enable **INCLUDE_DRV_STORAGE_INTEL_ICH** in **config.h**, then change **DEFAULT_BOOT_LINE** to **DEFAULT_BOOT_LINE_PATA** (CF device will be /ata01:1, primary PATA disk should be /ata00:1),

If you want to boot vxWorks image from SATA disk, enable **INCLUDE_DRV_STORAGE_INTEL_AHCI** in **config.h**, then change **DEFAULT_BOOT_LINE** to **DEFAULT_BOOT_LINE_SATA**

In command line (for example under Linux)
Open a VxWorks Development Shell, in the BSP directory
"$(**WIND_BASE**)/target/config/vx3030", run the command:

```
'make clean'
'make bootrom.bin BOOTAPP=y'
```

BOOTAPP flag is important for the bootrom. Do not forget it! (only under shell command)

The bootrom can be generated under the Workbench tool, no BOOTAPP flag is needed.

copy the bootrom image ("bootrom.bin") to the disk.
copy vxWorks image to disk if you want to load image from it.

3. Install and boot GRUB4DOS on bootable disk (USB, SATA)

This method describes the Grub installation on USB or SATA device from your own PC. **This is not the recommanded method.** The recommanded method is to format and install Grub using vxWorks. See following chapters 4 (for SATA) or 5 (for USB) for this.

1) Install GRUB4DOS on disk from your PC to start the vxWorks boot loader (for USB, SATA)

(NOTE: you can download the grub4dos from
http://sourceforge.net/projects/grub4dos/
and grub4das installer an installer for Windows from
http://download.gna.org/grubutil/grubinst-1.1-bin-w32-2008-01-01.zip)

a) INSTALLING from Windows host (Windows 7/XP was tested)

USB: Insert a USB key to the host PC and format it to the FAT16 file system
(NOTE: USB stick capacity must not exceed 2GB)
SATA/PATA hard disk: create a single primary partition not larger than 2GB and format it to the FAT16 file system

Run "Command Prompt" as Administrator and execute the command:

```
grubinst (hdN)
```

where (hdN) represents the disk drive (N is a number of the disk: 0,1,2,...).
For example, if a system has only one physical hard disk (not "C","D",etc.,)
these could be logical disks, not physical,
the USB key will be a 2nd physical disk in the system, and the correct command will be :

```
grubinst (hd1)
```

copy the "grldr" and "menu.lst" files to the USB key / SATA primary partition.

b) INSTALLING from Linux host
Download the grub4dos from
http://sourceforge.net/projects/grub4dos/
USB : Insert a USB key to the host PC and format it to the FAT16 file system
(NOTE: USB stick capacity must not exceed 2GB)

```
mkdosfs -v -F 16 -S 512 /dev/sdb
```

SATA/PATA hard disk: create a single primary partition not larger than 2GB and format it to the FAT16 file system

```
mkdosfs -v -F 16 -S 512 /dev/sdb1
```

Unzip the grub4dos package and install grub to the MBR by invoking:

```
bootlace.com /dev/sdXÂ
```

where **/dev/sdX** is the device, e.g. **/dev/sdb** (NOT SATA disk partition, e.g /dev/sdb1)

Mount the device and copy the "grldr" and "menu.lst" files to it.

2) Configure grub4dos

Create the "menu.lst" file (example of the content) :

```
timeout 30
default /default
title vxWorks Bootloader
root (hd0,0)
chainloader --force --load-segment=0x0800 --load-offset=0 --boot-cs=0x0800 --boot-ip=0
/bootrom.bin
boot
```

3) Setup BIOS to BOOT from the specified disk,
at the VGA console you should see something like:

```
 GRUB4DOS 0.4.4 2009-06-20, Memory 637K / 1012M, MenuEnd: 0x48AA5     0
------------------------------------------------------------------------
vxWorks boot loader




------------------------------------------------------------------------
```

4) Press enter and grub4dos will start vxWorks bootloader, you will see:

```
  Booting vxWorks boot loader

FileSystem type is fat, partition type 0x06
```

5) Finally check the vxWorks boot loader console (usually RS232 serial port 0). Then setup the boot loader (if necessary) to load your VxWorks image.

When the bootrom boots up, it can enumerate out the USB key, sometimes you need to plug out and plug it in again, because of some different vendors' USB keys compatibility issues. Enter following command in VxWorks Boot Shell,

```
[VxWorks Boot]: devs
```

You will see the following:

```
drv name
   0 /null
   1 /tyCo/0
   1 /tyCo/1
   6 host:
   3 /bd0

[VxWorks Boot]: ?
```

You will see some information as follows:

```
Device Formats:
  boot device: usb                 file name: /bdx/vxWorks

Boot Flags:
  0x02 -   load local system symbols
  0x04 -   don't autoboot
  0x08 -   quick autoboot (no countdown)
  0x80 -   use tftp to get boot image

Available Boot Devices:
  Filesystem Devices:  /bd0  bd0
  Enhanced Network Devices: gei0
[VxWorks Boot]:
```

The "/bd0" shows vxWorks bootrom found on the USB key.
Enter **c** command in the VxWorks Boot Shell to change the boot device to "usb" and the file name to "/bd0/vxWorks", save it. Finally, enter @ command to start the vxWorks image, and you'll see vxWorks shell.

if you use HDD instead of USB, you will see the device appearing as "/ata00:1" (PATA Master) or "/ahci00:1" (SATA)

if necessary, use the **c** command to change
the boot device to the appropriate device, file name. Save it, and use @
command to load vxWorks image, and you'll see vxWorks shell.

4. Install and boot with the MKBOOT vxWorks command on a bootable SATA disk

1) Boot vxWorks kernel including mkboot

mkboot command is available in the vxWorks BSP of VX3030. This command can be used to install a bootloader on SATA disk directly from VxWorks.
To do this, you must first build a bootrom correctly configured (see above) and the corresponding standalone kernel (make vxWorks.st). You must first boot this kernel by the way you prefer (PXE bootrom or USB bootrom). Once this kernel is booted then, you should have access to the target SATA disk device.

2) Format the disk and install boot loader under vxWorks Shell command

Execute the following command to prepare the bootable disk
(for example, with a SATA device and 4 partitions created, the SATA formating can also be done by calling **usrCreatePartitions( )** routine. This will have the same effect as the following command)

```
-> xbdCreatePartition "/ahci00:1", 4, 33, 33, 33
        Instantiating /ahci00 as rawFs, device = 0x1
        Instantiating /ahci00:1 as rawFs,  device = 0xc0001
        Instantiating /ahci00:2 as rawFs,  device = 0xd0001
        Instantiating /ahci00:3 as rawFs,  device = 0xe0001
        Instantiating /ahci00:4 as rawFs,  device = 0xf0001
        value = 0 = 0x0

-> dosFsVolFormat "/ahci00:1",0x02,0
        Formatting /ahci00:1 for DOSFS
   " ...
        OK.
        value = 0 = 0x0

-> dosFsVolFormat "/ahci00:2",0,0
        Formatting /ahci00:2 for DOSFS
        " ...
        OK.
        value = 0 = 0x0

-> dosFsVolFormat "/ahci00:3",0,0
        Formatting /ahci00:3 for DOSFS
     " ...
        OK.
        value = 0 = 0x0

-> dosFsVolFormat "/ahci00:4",0,0
     " ...
        OK.
        value = 0 = 0x0

-> copy "bootrom.bin","/ahci00:1/bootrom.sys"
```

```
                value = 0 = 0x0

        -> mkMbrAhci 0,0
                value = 0 = 0x0

        -> mkbootAhci 0,0,"bootrom.bin"
                num bytes written = 500728
                value = 0 = 0x0

        -> copy "vxWorks.st","/ahci00:1/vxWorks.st"
                value = 0 = 0x0

        Then reboot and Select SATA device from the BIOS Setup (Enter F2 to Setup)

        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ++++++++++++++++++++++++++++++++

                              VxWorks System Boot
Copyright 1984-2008  Wind River Systems, Inc.

        CPU: Kontron VX3030 SYMMETRIC IO
        Version: VxWorks 6.8
        BSP version: 2.0/11272
        Creation date: Sep 26 2011, 11:24:48

        Press any key to stop auto-boot...
        1

        [VxWorks Boot]: c
'.' = clear field;  '-' = go to previous field;  ^D = quit

        boot device          : gei0 ata=0,0
        processor number     : 0
        host name            : pcsstr2
        file name            : /tmp/vxWorks.st /ahci00:1/vxWorks.st
        inet on ethernet (e) : 172.20.161.25:0xffffff00
        inet on backplane (b):
        host inet (h)        : 172.20.161.184
        gateway inet (g)     :
        user (u)             : vxworks
        ftp password (pw) (blank = use rsh): vxworks
        flags (f)            : 0x8
        target name (tn)     : vx3030
        startup script (s)   :
        other (o)            : gei0

        [VxWorks Boot]: @
        boot device          : ata=0,0
        unit number          : 0
        processor number     : 1
        host name            : pcsstr2
        file name            : /ahci00:1/vxWorks.st
        inet on ethernet (e) : 172.20.161.25:0xffffff00
        host inet (h)        : 172.20.161.184
        user (u)             : vxworks
        ftp password (pw)    : vxworks
        flags (f)            : 0x8
```

```
target name (tn)      : vx3030
other (o)             : gei0

Loading /ahci00:1/vxWorks.st...1725728 + 383248

Starting at 0x408000...
Attached IPv4 interface to gei unit 0
Adding 6233 symbols for standalone.

]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks 6.8
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]      KERNEL: WIND version 2.13
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]      Copyright Wind River Systems, Inc., 1984-2009

CPU: Kontron VX3030 SYMMETRIC IO.  Processor #1.
Memory Size: 0x3eee0000.  BSP version 2.0/11269.
Created: Sep 26 2011, 11:19:16
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.

-> devs
        drv name
        0 /null
        1 /tyCo/0
        1 /tyCo/1
        7 pcsstr2:
        10 /vio
        3 /ahci00:1
        3 /ahci00:2
        3 /ahci00:3
        3 /ahci00:4
    value = 0 = 0x0
-> ll "/ahci00:1"

Listing Directory /ahci00:1:

-rwxrwxrwx  1 0       0              430432 Jan  1  1980 bootrom.sys
-rwxrwxrwx  1 0       0             2276470 Jan  1  1980 vxWorks.st
```

5. Install and boot with the MKBOOT vxWorks command on a bootable USB disk

1) Boot the vxWorks kernel including mkboot

The mkboot command is available in the vxWorks BSP of VX3030. This command can be used to install a bootloader on a USB disk directly from VxWorks.
To do this, you must first build a bootrom correctly configured (see above) and the corresponding standalone kernel (make vxWorks.st). You must first boot this kernel by the way you prefer (PXE bootrom or USB bootrom). Once this kernel is booted then you should have access to the target USB disk device.

2) Format the disk and install the boot loader under the vxWorks Shell command

Execute the following command to prepare the bootable disk
(for example, with a USB device and 2 partitions created, the USB formating can also be done calling **usrCreatePartitions( )** routine. This will have the same effect as the following command)

```
        -> xbdCreatePartition ("/bd0", 2, 94, 0, 0)
                Instantiating /bd0 as rawFs, device = 0x1
                Instantiating /bd0 as rawFs,  device = 0x50001
                Instantiating /bd0:2 as rawFs,  device = 0x60001
                value = 0 = 0x0

        -> dosFsVolFormat ("/bd0",0x12,0)
                Formatting /bd0 for DOSFS
           " ...
                OK.
                value = 0 = 0x0

        -> dosFsVolFormat ("/bd0:2",0x20,0)
                Formatting /bd0:2 for DOSFS
                Instantiating /bd0:2 as rawFs, device = 0x60001
            " ...
                OK.
                value = 0 = 0x0

        -> copy "bootrom.bin","/bd0/bootrom.sys"
                value = 0 = 0x0

        -> mkMbrUsb("/bd0")
                value = 0 = 0x0

        -> mkbootUsb ("/bd0","bootrom.bin")
                num bytes written = 500728
                value = 0 = 0x0

        -> copy "vxWorks.st","/bd0/vxWorks.st"
                value = 0 = 0x0

        Then reboot and Select USB device from the BIOS Setup (Enter F2 to Setup)

        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ++++++++++++++++++++++++++++++++++

                                VxWorks System Boot
Copyright 1984-2008  Wind River Systems, Inc.

        CPU: Kontron VX3030 SYMMETRIC IO
        Version: VxWorks 6.8
        BSP version: 2.0/11272
        Creation date: Sep 26 2011, 11:24:48

        Press any key to stop auto-boot...
        1

        [VxWorks Boot]: c
'.' = clear field;  '-' = go to previous field;  ^D = quit

        boot device          : gei0 usb
```

```
processor number     : 0
host name            : pcsstr2
file name            : /tmp/vxWorks.st /bd0/vxWorks.st
inet on ethernet (e) : 172.20.161.25:0xffffff00
inet on backplane (b):
host inet (h)        : 172.20.161.184
gateway inet (g)     :
user (u)             : vxworks
ftp password (pw) (blank = use rsh): vxworks
flags (f)            : 0x8
target name (tn)     : vx3030
startup script (s)   :
other (o)            : gei0

[VxWorks Boot]: @
boot device          : usb
unit number          : 0
processor number     : 0
host name            : pcsstr2
file name            : /bd0/vxWorks.st
inet on ethernet (e) : 172.20.161.25:0xffffff00
host inet (h)        : 172.20.161.184
user (u)             : vxworks
ftp password (pw)    : vxworks
flags (f)            : 0x8
target name (tn)     : vx3030
other (o)            : gei0

Booting from USB file - /bd0/vxWorks.st
Found Bulk Device with 1 Logical Units at node 103
Failed to open "/bd0/vxWorks.st".  Attempt 1 of 10.
1966896 + 211872
Starting at 0x408000...

Bulk Class Driver Successfully Initialized
Target Name: vxTarget
Found Bulk Device with 1 Logical Units at node 103

]]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks 6.8
]]]]]]]]]]]]]]]]]]]]]]]]]]]]      KERNEL: WIND version 2.13
]]]]]]]]]]]]]]]]]]]]]]]]]]]]     Copyright Wind River Systems, Inc., 1984-2009

CPU: Kontron VX3030 SYMMETRIC IO.  Processor #1.
Memory Size: 0x3eee0000.  BSP version 2.0/11269
Created: Sep 26 2011, 11:19:16
ED&R Policy Mode: Deployed
WDB Comm Type: WDB_COMM_END
WDB: Ready.

-> devs
        drv name
        0 /null
        1 /tyCo/0
        1 /tyCo/1
        7 pcsstr2:
        10 /vio
        3 /bd0
```

```
        3 /bd0:2
value = 0 = 0x0

-> ll "/bd0"

        Listing Directory /bd0:
        -rwxrwxrwx  1 0       0               507776 Jan  1  1980 bootrom.sys
        -rwxrwxrwx  1 0       0              2952421 Jan  1  1980 vxWorks
        value = 0 = 0x0
```

HARDWARE FEATURES SUPPORTED BY VX3030 BSP

List of the hardware features supported by this BSP

This section gives the hardware features list supported onto the board.

Hardware Interface	Driver/Component	Status	Tested
DRAM	1GB/2GB modif. in **conf.h**	SUPPORTED	Yes (Fixed Size in **config.h**)
DRAM	3GB	SUPPORTED	No
UART:0	**vxbNs16550Sio.c**	SUPPORTED	Yes (RS232 only)
UART:1	**vxbNs16550Sio.c**	SUPPORTED	Yes (RS232 only)
1GB-ETHERNET:0	**gei825xxVxbEnd.c** (modified)	SUPPORTED	Yes
1GB-ETHERNET:1,2	**gei825xxVxbEnd.c** (modified)	SUPPORTED	Yes
SATA	**intelAhciDrv.c**	SUPPORTED	Yes (AHCI Only)
GRAPHICS	**vxbM6845Vga.c**	SUPPORTED	Yes (PC Console Only)
TIMER	**vxbI8253Timer.c**	SUPPORTED	Yes
APIC-TIMER	**vxbLoApicTimer.c**	SUPPORTED	Yes
TIMESTAMP	**vxbIntelTimestamp.c**	SUPPORTED	Yes
RTC	**vxbMc146818Rtc.c**		
	vxbMc146818RtcK.c	SUPPORTED	Yes
USB2-HOST	USB EHCI driver	SUPPORTED	Yes
USB1-HOST	USB UHCI driver	SUPPORTED	No
SMBus	**vxbPchSMBus.c**	SUPPORTED	Yes
CPLD Watchdog	**vxCpld.c**	SUPPORTED	Yes
EEPROM (2 devices)	**vxI2cEeprom.c**	SUPPORTED	Yes
CPL LEDS	**vxLED.c**	SUPPORTED	Yes
CPLD GPIO	**vxGPIO.c**	SUPPORTED	Yes (IO and Interrupt)

Temp Sensor (1 devices)	**vxLm73.c**	SUPPORTED	Yes
Voltage Sensor	**ads7830.c**	SUPPORTED	Yes
FMVRAM	**vxI2cEeprom.c**	SUPPORTED	Yes
AUDIO	n/a	UNSUPPORTED	No
I2C Backplane num 0	**vxI2cEeprom.c**	SUPPORTED	Yes
I2C Backplane num 1	**vxI2cEeprom.c**	SUPPORTED	Yes
Boot on HDD SATA	SATA AHCI	SUPPORTED	Yes
Boot on SATA Flash module	SATA AHCI	SUPPORTED	Yes (Greenliant 32GB)
Boot on USB Mass Storage	USB EHCI	SUPPORTED	Yes
Boot on USB Flash module	USB EHCI	SUPPORTED	Yes (Initialize USB disk under vxWorks)
Backplane PCIe for XMC/PMC	PMC/XMC vendor's driver	SUPPORTED	Yes

Serial Configuration

SIO should be enabled by the BIOS baseboard. Default configuration is COM0:
Baud Rate: 115200
Data: 8 bit
Parity: None
Stop: 1 bit
Flow Control: None

Network Configuration

This BSP provides support for VxBus drivers, including network drivers. The list of VxBus network drivers available for PCIe devices includes the following:

```
INCLUDE_GEI825XX_VXB_END
INCLUDE_END
```

To permit command line builds and bootrom building, the **config.h** file must be modified to define the macros (as described above) corresponding to the desired driver.

The following END is supported by this BSP:

Driver	**END**	**END2**
GEI	**INCLUDE_GEI825XX_VXB_END**	UNSUPPORTED

KONTRON DEVICES DRIVERS

Following is a short description of the device drivers developped by Kontron and specific to VX3030 like GPIO, EEPROM, Watchdog, etc.

1. Manage 3rd party drivers

VxBus Drivers developed by Kontron are located in the following directory :

```
$WIND_BASE/target/3rdparty/kontron_vx3030
```

It includes the following drivers

```
'ads7830'     - Voltage sensors driver
'rtc146818'   - RTC driver
'vxCpld'      - CPLD driver
'vxI2cEeprom' - EEPROM driver (2 EEPROM and 1 FMRAM towards I2C CPLD)
'vxLm73'      - LM73 1 temperature sensors driver (towards I2C CPLD)
'pchsmbus'    - SMBUS driver (used for voltage sensors)
'vxGPIO'      - GPIO driver (towards CPLD)
'vxLED'       - LED driver  (towards CPLD)
```

and an includes directory for *.h include files

For details on any of these drivers please refer to their own documentation located in $(**WIND_HOME**)/docs/extensions/eclipse/plugins/com.windriver.ide.doc.bsp/vx3030_6.8

```
vxbAds7830.html
vxbVxCpldDrv.html
vxbLm73.html
vxbVxI2cEepromDrv.html
vxbMC146818RtcK.html
vxGPIO.html
vxbPchSMBus.html
vxLED.html
```

These drivers are accessible by the user in the VX3030 BSP through the two library files :

* **libkontron.a** (UP mode)
* **libkontron_smp.a** (SMP mode)

These drivers do not implement standard I/O control entries but use a WindRiver Method called driverControl. This method allows the User API to access directly to the user driver routine with a structure of function pointers. For example the driver ads7830 implements this feature. Note that other drivers implement the same feature. This is the declaration of the method in the driver :

```
LOCAL device_method_t vxbAds7830Drv_methods[] =
    {
    DEVMETHOD(driverControl, vxbAds7830FuncGet),
    { 0, 0}
    };
```

The routine vxbAds7830FuncGet will return to the user application a pointer to the following structure that can be retrieved in each .h driver file.

```
struct vxbAds7830Control
    {
    VXB_DEVICE_ID pInst;

    STATUS (*chanRead)
        (
        VXB_DEVICE_ID pInst,
        UINT8 chan,
        UINT8 *pVal
        );

    STATUS (*voltGet)
        (
                VXB_DEVICE_ID pInst,
                UINT8 chan,
                UINT8 **ppName,
                UINT32 *pVal
        );
     };
```

To point to the correct vxBus Driver and retrieve the driverControl Method, we recommend to use the following routine:

```
void *vxbGetCtrlInfo(char *instName, int unit)
```

This routine is present in the VX3030 BSP in the **bspUtil.c** file.

To summarize how the user can use the driverControl Method the vx3030 BSP contains example like the following one that is already present in the BSP and allow the user to get the temperature of the board. This code is present in the BSP in the file name **tempGet.c** :

```
#define NB_SENSORS      1
LOCAL BOOL drvLm73CtrlStatus[NB_SENSORS] = {FALSE};

LOCAL struct vxbLm73Control *drvLm73Ctrl[NB_SENSORS];

STATUS tempGet(char *name,int drvNum,float *temp)
{
        STATUS status;

        if(drvLm73CtrlStatus == FALSE)
                drvLm73Ctrl = vxbGetCtrlInfo(name,drvNum);

        if(drvLm73Ctrl == NULL || drvLm73Ctrl->pInst == NULL)
        {
                return -1;
        }
        else
        {
                drvLm73CtrlStatus = TRUE;
        }
```

```
        status = drvLm73Ctrl->tempGet(drvLm73Ctrl->pInst,temp);

        return status;
}
```

2. Voltage Sensors

ADS7830 Voltage sensors are available behind the Board SMbus. In the "examples" directory, **voltage.c** gives an example on how to get and test the voltage.

```
-> ads7830Test
0: Getting driver functionality : OK
1: Reading registers : OK
2: Getting voltage values :
Channel 0: 12V VPX 11892
Channel 1: 5V VPX 4886
Channel 2: 3V3 3300
Channel 3: 2V5 6U 2498
Channel 4: 3V3 VPX SB 3222
Channel 5: 1V05S 3U 1025
Channel 6: 1V05 3U IBEX 1044
Channel 7: 1V 3U 976 OK
3: Checking voltage ranges :
Channel 0: 12V VPX 11892 [min=11500,max=12500]
Channel 1: 5V VPX 4886 [min=4500,max=5500]
Channel 2: 3V3 3300 [min=3000,max=3500]
Channel 3: 2V5 6U 2498 [min=2300,max=2800]
Channel 4: 3V3 VPX SB 3222 [min=3000,max=3500]
Channel 5: 1V05S 3U 1025 [min=1000,max=1100]
Channel 6: 1V05 3U IBEX 1044 [min=1000,max=1100]
Channel 7: 1V 3U 986 [min=950,max=1050] OK

TOTAL TESTS = 4, failed = 0

value = 0 = 0x0
->
```

3. RTC Use

Internal IBEXPEAK PCH RTC is used to manage the current date. The RTC in the IBEXPEAK is the MC146818. A vxBus driver has been adapted to use it in RTC mode. The name of this vxBus driver is "mc146818RtcKontron". The standard "vxbRtcGet" and "vxbRtcSet" vxbus driver entries can be used. To use Kontron RTC vxBus driver for date and time in project then define the following include:

```
DRV_RTC_MC146818
```

See the **examples** directory to find an example code for rtcGet and rtcSet routines

```
-> rtcGet
Year:12, Month:2, Day:21, WDay:2
Hour:13, Min:59, Sec:22
value = 0 = 0x0
```

```
->
```

4. CPLD Watchdog

CPLD Watchdog is available through CPLD register access. It uses vxCpld vxBus driver method. Watchdog can be used in interrupt, timer or reset mode. An example on how to use vxCpld and watchdog is given in the "examples" directory in the file **cpldWdt.c**.

```
-> cpldWdt
0: Getting functionality : OK
1: Getting CPLD parameters : OK
Cpld version : 0xb (1)
Cpld geo id : 0x1d (1)
2: Checking CPLD parameters : OK
3: Activate watchdog (10 sec)  : OK
4: Trigger watchdog in 9 sec  : OK
5: Disable the watchdog in 10 sec : OK
6: Reset the controller (controller must be reset in 5 sec)  :
TOTAL TESTS = 6, FAILED = 0

value = 0 = 0x0
->

Then board Reset
```

5. Board Information (VPD)

Board information or Vital Product Data (VPD) like Serial Number can be accessed from VPD EEPROM. If necessary, the user can include and link application located in $**WIND_BASE**/target/3rdparty/kontron_vx3030/vxI2cEeprom/ with 3 files

```
vx6060_vpd.c
vxvariant.c
vxvpdtool.c
```

The VPD can be read with **vpddisplay( )**. This routine is included into the EEPROM vxBus **vxI2cEeprom** folder but it is not loaded by default into the kernel. To include the command **vpddisplay( )** you can invoke it somewhere into the code. For example into

```
extern STATUS vpddisplay(int dispflags);
void usrAppInit (void)
    {
        vpddisplay(0);
#ifdef  USER_APPL_INIT
        USER_APPL_INIT;
#endif
```

Then compiling will cause the **vpddisplay** command to be included into the kernel :

```
-> vpddisplay(0)
Board type   : VX3030-SA22-0N100
EC Level     : 01002
Serial Number: 1811271050062
Variant      : 03100440100a0001
Keylist      :
/PCB_A/SACLASS/MEZZSATA/BHQUAD/ARDLV2G/IBOMON/ETHSTD6U/1GB_DDR3_1067/DDR3-1067/IRTC/XDPON/STDCLK/
CK505REFOSC/STD_GND/SCAPON/HDDOFF/NOMEZZIO/2G5TS/P1PWRMAGOFF/PWRMAGOFF/FP422OFF/TMLVDSON/I2CSTD/
I2CA/1SLOT/STDVPXKEY/P80ON/VS2V12/NOSYNCAB/ITIN/BC/


value = 0 = 0x0
-> vpddisplay(1)
03100440100a0001
value = 0 = 0x0
-> vpddisplay(2)
01002
value = 0 = 0x0
->
```

6. EEPROMs/FMRAM Accesses

Two EEPROMs and one FMRAM are available on Kontron Core i7 VX3030 boards.

One EEPROM is VPD reserved (Vital Product Data) and the second is User reserved. The User reserved EEPROM can thus be used entirely by an application. The FMRAM is also totaly free to be used. All these devices are available through the vxBus vxI2CEeprom driver.

An example on how to access FMRAM is given with file **vxFramTool.c** with **target/3rdparty/kontron_vx3030/vxI2cEeprom/vxFramTool.c** routines:

```
STATUS vxprintfmram(int offset, int size)
STATUS vxwritefmram(int offset, int size, UINT8 buffer[])
```

These routines are part of the vxBus library but not linked to the kernel by default. The user needs to invoke it somewhere into the code to include them.

This file example is easy to adapt for EEPROM. Instead of calling eeprom->fmramRead or write method user can call any other appropriate method like ->userRead or ->userWrite to have access to the User EEPROM etc. See vxI2cEeprom driver documentation itself for a complete vxBus method.

Example of use, dump FMRAM from offset 0 size 40 bytes

```
-> vxprintfmram(0,40)
Raw FMRAM data:
0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0
0x 0 0x 0 0x 0 0x 0 0x20 0x20 0x20 0x38 0x34 0x36 0x39 0x33 0x30 0x38 0x38 0x36
0x20 0x20 0x20 0x20 0x20 0x20 0x31 0x36
value = 0 = 0x0
->
```

7. CPLD Temperature Sensors

There is a LM73 temperature sensors available on the VX3030 (with I2C CPLD). It can be read simply with the command **tempGetShow** included into the BSP in **tempGet.c** file.

tempGetShow must be invoked with the LM73 sensor number ID as single parameter.

```
STATUS tempGetShow(int num);

-> tempGetShow(0)
Temp = 51.50
value = 0 = 0x0
```

8. CPLD GPIO

There is 5 GPIOs available on VX3030. The second GPIO (GPIO1) is reserved for Reseting board when its direction is set to INPUT so it should be used only as OUTPUT. GPIO are managed with vxBus driver vxGPIO. See **vxbGPIO.h** file to find all GPIO mode that are programmables. See **vxbGPIO.c** for all vxBus method associated with GPIO

GPIO programmable vxBus methods and modes can be :

- GPIO getDirection/setDirection (**GPIO_DIRECTION_IN**, **GPIO_DIRECTION_OUT**)

- GPIO getValue/setValue (**GPIO_VALUE_LO**, **GPIO_VALUE_HI**)

- GPIO getIntMode/setIntMode: level sensitive or edge (**GPIO_INTMODE_LEVEL**, **GPIO_INTMODE_EDGE**)

- GPIO getIntEnabled/setIntEnabled

- GPIO getIntPolarity/setIntPolarity (**GPIO_POLARITY_LO**, **GPIO_POLARITY_HI**)

- GPIO getIntToggle/setIntToggle (**GPIO_TOGGLE_OFF**,**GPIO_TOGGLE_ON**)

When a GPIO is selected in Toggle Mode GPIO, the interrupt Mode and polarity are not taken into consideration. If enabled then an interruption will be generated if a GPIO changes its state.

A test example is given in the **examples** directory file **gpioTest.c**. The test can work only if the GPIOs are connected two by two and it will demonstrate how to program them.

9. CPLD LEDs

Two front side LEDs can be controlled by the vxLED vxBus driver. Leds colors and blink modes can be set. See **vxLED.h** for available modes. See **led.c** file in example directory for a LEDs test demonstration.

10. I2C Backplane

Two SMbus (100Kz) are available on VX3030 P0 VPX backplane. SMbus 0 is always accessible while SMbus 1 is only accessible for a Board System Controller. A program using CPLD I2C driver is given into BSP file **example/i2cBpAccess.c**. This program is adapted for backplane EEPROM device. The Routine **i2cBpAccessScan( )** provides a scan example of SMbus backplane while other routines demonstrate how to read write into a device (this is adapted for EEPROM device with a page size of 32)

```
-> ld < i2cBpAccess.o
value = 136237232 = 0x81ed0b0
-> i2cBpAccessScan

---------- I2C SCAN BUS EXT 0 ------------
8bits accesses devs found  : 0x32 0xa0
16bits accesses devs found : 0x32 0xa0
---------- I2C SCAN BUS EXT 1 ------------
8bits accesses devs found  :
16bits accesses devs found :
value = 0 = 0x0
-> basicTestEEPROM(0,0xA0)
** EEPROM 0xa0 on SMB 0 ***
Comparing data
value = 0 = 0x0
->
```

11. Get PBIT Result

If available PBIT (POwer-On Built In Tests) results are read from EEPROM (if PBIT have been ordered with the BIOS). They can be read with the command **pbitdisplay( )** available in the BSP. See file **vxpost.c** in 3rdParty

```
-> pbitdisplay
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
    pcie_vpx_sw: PASSED
         serial: PASSED
            rtc: PASSED
       sysflash: PASSED
           cpld: PASSED
   temp_sensors: PASSED
    temperature: PASSED
         fnvram: PASSED
    ether_loop0: PASSED
    ether_loop1: PASSED
    ether_loop2: PASSED
    ether_loop3: PASSED
    ether_loop4: PASSED
        voltage: FAILED
sata0_controler: PASSED
```

```
sata1_controler: PASSED
            vpd: PASSED
         eeprom: PASSED
usb1_controller: PASSED
usb2_controller: PASSED
         system: PASSED (FAILED ONCE)

PASSED  : 26
FAILED  : 1
NOT RUN : 0
TOTAL   : 27

value = 0 = 0x0
->
```

BSP INFORMATION

1. Identify BSP Release ID

To identify the version of this VX3030 BSP see **KRELEASE** file into vx3030 BSP directory.

2. SMP mode

The BSP supports SMP (Symmetric Multiprocessing) mode.

3. Make Targets

The make targets are listed as the names of object-format files. Other images not listed here may not be tested.

bootrom
bootrom.bin
bootrom.pxe
vxWorks (with **vxWorks.sym**)
vxWorks.st

4. BSP Bootloaders and Bootroms

Bootloader/Bootrom	**Status**
BIOS (pre-flashed)	SUPPORTED
bootrom	SUPPORTED
bootrom_uncmp	UNSUPPORTED
vxWorks	SUPPORTED
vxWorks_rom	UNSUPPORTED

vxWorks_romCompress	SUPPORTED
vxWorks_romResident	UNSUPPORTED

5. BSP Validated Tools

Tool	Connection/Type	Status
Workbench Debugger	ETHERNET/WDB	SUPPORTED
Workbench System Viewer	ETHERNET/WDB	SUPPORTED
Workbench MemScope	ETHERNET/WDB	UNVALIDATED
Workbench ProfileScope	ETHERNET/WDB	UNVALIDATED
Workbench StethoScope	ETHERNET/WDB	UNVALIDATED
Workbench CoverageScope	ETHERNET/WDB	UNVALIDATED
Workbench Core File Analysis	ETHERNET/WDB	UNVALIDATED
Workbench Sensor Points	ETHERNET/WDB	UNVALIDATED
Workbench OCD Debugger	JTAG	UNSUPPORTED
Workbench OCD Flash	JTAG	UNSUPPORTED

6. VxWorks on VGA, PC_CONSOLE mode

This function will use PC monitor (VGA) as a VxWorks console. To use these controller, the **INCLUDE_PC_CONSOLE** directive must be enabled in **config.h**. (this is to be done before Workbench project creation). USB keyboard is supported.

7. intelAhciDrv and intelAhciShow

To use this WindRiver SATA Hard disk driver as a boot device, the **INCLUDE_DRV_STORAGE_INTEL_AHCI** directive must be enabled in **config.h**, vxWorks image support is enabled by the'**INCLUDE_DRV_STORAGE_INTEL_AHCI**' component.

8. SMBus on PCH Controller

To use onboard sensors, the Intel SMbus of Platform Controller Hub (PCH) driver must be included,(define **DRV_SMBUS_INTEL_PCH**). This driver is provided by Kontron as a 3rdparty driver.

9. BSP Timer Routines

The BSP provides its own routines to help for time measurement.

gettimeofdayus : gives the current timestamp with microsecond precision
gettimeofdayns : gives the current timestamp with nansecond precision
gettimeofdayShow : shows the current timestamp in seconds and microseconds
gettimeofdayShow : shows the current timestamp in seconds and nanoseconds

To use these routines you must include the file **kontron.h** in your application. Then a time measurement can be done for example in microseconds with the following code:

```
#include <sys/times.h>
#include <kontron.h>

    struct timeval start;
    struct timeval end;
    int totalTime;

    gettimeofdayus(&start);

     ... What you need to measure ...

    gettimeofdayus(&end);

    totalTime = (int)(end.tv_sec*1000000LL+end.tv_usec - (start.tv_sec*1000000LL+start.tv_usec));
    printf ("Done in %d us\n", totalTime);
```

10. vxbI8253Timer

This WindRiver library contains a board-independent interface to manipulate the timer functions on Intel 8253 and compatible timer chip devices. Timer 0 is used as the system clock. According to different applications, timer 1,2 has different usages. (See **vxbI8253Timer.html** WindRiver documentation for more information)

11. MC146818 RTC clock driver

This library is the vxBus compliant driver which implements the functionality specific to the MC146818 real time clock Timer (define **DRV_RTC_MC146818**). It doesn't implement the standard MC146818 timer driver functionality but adds TC time functions

The user application can use the VxBus custom methods defined for this driver:

```
METHOD_DECL(vxbRtcGet)
METHOD_DECL(vxbRtcSet)
```

and the methods should be called as

```
struct tm t;

vxbDevMethodRun(DEVMETHOD_CALL(vxbRtcGet), &t);
vxbDevMethodRun(DEVMETHOD_CALL(vxbRtcSet), &t);
```

See also above chapter named **RTC Use**

12. vxbIntelTimestamp

This WindRiver library is the driver for the timestamp on Intel chipsets to use the timestamp feature, the macro **INCLUDE_TIMESTAMP** must be defined in **config.h**. See **vxbIntelTimestamp.html** for more

details.

13. USB Host Support

A standard USB EHCI / USB UHCI controller is supported in this BSP. Please refer to Wind River's USB documents.

14. Boot Devices

The supported boot devices are:

gei - Primary Ethernet (10baseT, 100baseTX, or 1000baseT/BX)
ata - ATA/IDE drive (SATA, PATA)
usb - USB disk

15. Boot Methods

The boot methods are affected by the boot parameters. If no password is specified, RSH (remote shell) protocol is used. If a password is specified, FTP protocol is used, or, if the flag is set to 0x80, TFTP protocol is used.

16. Software Board reset method

The standard WRS routine **reboot( )** or CTRL-X can be used to reboot on bootrom. **sysHardReset( )** command can be used to reboot the board and restart the board from BIOS. So the reset routines are

reboot - jump on bootrom
sysHardReset - reset board and restart BIOS
sysVpxReset - reset board and restart BIOS and possibly reset entire VPX rack

KNOWN PROBLEMS

1. Memory Autoscan

Memory auto scan is not supported. **LOCAL_MEM_AUTOSIZE** must be disabled and RAM size should be set in **config.h** e.g.
For 1GB RAM board configuration

```
#define  SYSTEM_RAM_SIZE  (0x3B19A000)
```

For 2GB RAM board configuration (default)

```
#define  SYSTEM_RAM_SIZE  (0x7B19A000)
```

This size limitation is due to the fact the ACPI table fills a part of the latest 80MB of memory. This memory configuration is done under the UEFI BIOS VX3030 ID12059

2. PCIE IRQ on Old BIOS

The PCI express IRQ in legacy mode not correctly routed by the BIOS ID11322 or lower release. Can impact some PMC or XMC connected to the PCI express backplane. Corrected since BIOS 11328

3. GPIO Level mode not supported

The GPIO LEVEL mode is not supported. This is a CPLD board problem. The recommanded GPIO mode is Edge mode.

4. Possible long boot delay when booting on USB device with separate symbol table

This is a WindRiver defect WIND00322792. When booting from usb device, with a separate symbol table and with a network to initialize, then the boot will take 1 minute 30s. This issue occurs only if the ethernet is not connected or if there is no traffic on the ethernet. The workaround is to always link the symbol table to the kernel (**INCLUDE_STANDALONE_SYM_TBL** must be defined).

BSP EVOLUTIONS

1. Compared to Previous Release

1.1 BSP VX3030

None, first release

1.2 VX3030 vxBus driver

None first release

BIBLIOGRAPHY

Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture

Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference

Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide

AP-485, Intel Processor Identification and the CPUID Instruction

PCI System Architecture, Fourth Edition, Addison-Wesley, 1999, ISBN 0-201-30974-2

SEE ALSO

VxWorks User's Guide: Getting Started, *VxWorks Programmer's Guide: Configuration*, *VxWorks Programmer's Guide: Architecture Supplement*, *vxworks Application Programmers Guide 6.8*

MAILING ADDRESS
Kontron Modular Computers S.A.S.
150 rue Marcelin Berthelot - BP 244
ZI TOULON EST
83078 TOULON CEDEX - France

TELEPHONE AND E-MAIL
+33 (0) 4 98 16 34 00
sales@kontron.com
support-kom-sa@kontron.com

For further information about other Kontron products, please visit our Internet web site: www.kontron.com.