ADB3 Driver Layer 1.4.27 for VxWorks Release Note

Introduction

This release note accompanies the ADB3 Driver Layer for VxWorks. The latest version of this driver can be found at:

https://support.alpha-data.com/pub/admxrcg3/vxworks

For support, send e-mail to support@alpha-data.com

The ADB3 Driver Layer for VxWorks is a custom layer that can be built into a VxWorks kernel image.

Operating systems supported

This release of the ADB3 Driver Layer for VxWorks supports the following operating systems:

- VxWorks 7

Hardware supported

This release of the ADB3 Driver Layer for VxWorks supports the following Alpha Data hardware:

- ADM-XRC-6TL
- ADM-XRC-6T1
- ADM-XRC-6T-DA1
- ADM-XRC-6TGE and ADM-XRC-6TGEL
- ADM-XRC-6T-ADV8
- ADPE-XRC-6T and ADPE-XRC-6T-L
- ADPE-XRC-6T-ADV
- ADM-XRC-7K1
- ADM-XRC-7V1
- ADM-VPX3-7V2
- ADM-PCIE-7V3
- ADM-PCIE-KU3
- ADM-XRC-KU1 and ADM-XRC-KU1A-T8
- ADM-PCIE-8V3
- ADM-PCIE-8K5
- ADM-AD01478
- ADM-AD01479

License agreement

Please refer to the files **license.rtf** or **license.txt** within this software package for the licensing terms that apply to this software. Please contact Alpha Data if alternative licensing terms are required.

Alpha Data reserves the right to use different licensing terms for future releases of this software.

Installation

Installation in a Windows VxWorks host

To install the ADB3 Driver Layer in a Windows VxWorks host, unzip **vxbAdb3-vxw7-1.4.27.zip** into the **%WIND_PKGS%** folder. If prompted about whether or not to allow merging of folders or overwriting files, respond with "allow". If done correctly, this should produce a directory **%WIND_PKGS%\custom\alphadata\vxbAdb3-1.4.27**.

After installing the ADB3 Driver Layer the next steps are typically:

- Open an existing VxWorks Source Build (VSB) project in Workbench, or create a new one, and include the ADB3 Driver Layer in it using the configurator.

  NOTE: In the configurator, the ADB3 Driver Layer can be found under the **alphadata** folder.
- Build the VSB project.
- Open an existing VxWorks Image project (VIP) in Workbench, or create a new one, that uses the aforementioned VSB project. Include the ADB3 VxBus Driver using the configurator.

  Optionally, include the ADB3 VxBus Driver Utilties using the configurator.

  NOTE: In the configurator, the ADB3 Driver can be found in the folder **Hardware -> Device drivers -> Other**.
- Build the VIP project to obtain a VxWorks kernel which includes the ADB3 VxBus Driver. The ADB3 VxBus Driver automatically starts during the latter stages of the kernel boot process.

Installation in a Linux VxWorks host

To install the driver in a Linux VxWorks host, extract **vxbAdb3-1.4.27.tar.gz** into the **$WIND_PKGS** directory. If done correctly, this should produce a directory **$WIND_PKGS/custom/alphadata/vxbAdb3-1.4.27**.

After installing the ADB3 Driver Layer the next steps are typically:

- Open an existing VxWorks Source Build (VSB) project in Workbench, or create a new one, and include the ADB3 Driver Layer in it using the configurator.

  NOTE: In the configurator, the ADB3 Driver Layer can be found under the **alphadata** folder.
- Build the VSB project.
- Open an existing VxWorks Image project (VIP) in Workbench, or create a new one, that uses the aforementioned VSB project. Include the ADB3 VxBus Driver using the configurator.

  NOTE: In the configurator, the ADB3 Driver can be found in the folder **Hardware -> Device drivers -> Other**.
- Build the VIP project to obtain a VxWorks kernel which includes the ADB3 VxBus Driver. The ADB3 VxBus Driver automatically starts during the latter stages of the kernel boot process.

Uninstallation

Uninstallation in a Windows VxWorks host

Uninstallation consists of deleting the layer from the **custom** layers folder of the VxWorks 7 installation. To do, manually delete the **%WIND_PKGS%\custom\alphadata** folder and its contents. Do not delete **%WIND_PKGS%\custom**, because this may contain layers from other vendors.

After deleting the layer, when a VSB project that depended on this layer is next opened, the layer should be automatically removed from the VSB project. It can then be rebuilt if desired, and any VxWorks image projects

(VIPs) that depend on the VSB project can also be rebuilt if desired.

Uninstallation in a Linux VxWorks host

Uninstallation consists of deleting the layer from the **custom** layers folder of the VxWorks 7 installation. To do, manually delete the **$WIND_PKGS/custom/alphadata** folder and its contents. Do not delete **$WIND_PKGS/custom**, because this may contain layers from other vendors.

After deleting the layer, when a VSB project that depended on this layer is next opened, the layer should be automatically removed from the VSB project. It can then be rebuilt if desired, and any VxWorks image projects (VIPs) that depend on the VSB project can also be rebuilt if desired.

VPD write-protection mechanism

The VPD write-protection mechanism described in the ADM-XRC Gen 3 SDK User Guide is exposed via the VxWorks kernel image configurator. If **VXBADB3_ENABLE_VPD_WRITE** is **TRUE**, the ADB3 Driver permits writes to VPD memory.

Note that for certain models, a board may additionally need to be operating in "Service Mode" in order to permit writes to VPD memory. Please refer to the User Manual for your reconfigurable computing board to determine whether or not "Service Mode" applies, and how to enter it.

VXBADB3_ENABLE_VPD_WRITE is the initial value of the global integer variable **adb3DrvEnableVpdWrite**. If necessary, this variable can be manipulated at runtime in order to enable or disable writes to VPD memory. This variable is checked every time an application attempts to write to the VPD memory, so changes to this variable take effect immediately rather than being sampled when the driver is started.

To set this value to 1 (thus enabling VPD writes at driver level) using the VxWorks shell, use:

```
-> adb3DrvEnableVpdWrite=(int)1
```

To set this value to 0 (thus disabling VPD writes at driver level) using the VxWorks shell, use:

```
-> adb3DrvEnableVpdWrite=(int)0
```

Common buffer support

The ADB3 Driver can allocate one or more "common buffers" at startup. The purpose of this feature is to support applications that use Direct Master data transfer, such as an ethernet-style I/O interface where the sizes and arrival times of packets of data are not known in advance by software running on the host. These buffers have the following characteristics:

- Persist until the driver is stopped.
- Guaranteed aligned to a specified power-of-2 address boundary.
- Allocated from the appropriate pool of memory in order to be contiguous and visible to bus-master devices.
- Can be mapped into the virtual address space of a user-mode process; see "ADMXRC3 API Specification 1.5.0" or later for details of the new ADMXRC3 API functions relating to common buffers.

This feature is exposed via the VxWorks kernel image configurator:

- **VXBADB3_NUM_COMMON_BUFFER** determines the number of common buffers allocated; the default is zero, meaning that no common buffers are allocated by default.
- **VXBADB3_COMMON_BUFFER_SIZE** determines the size in bytes of each common buffer; the default is 64 kiB (0x10000).
- **VXBADB3_COMMON_BUFFER_ALIGN** determines the power-of-2 address boundary to which each common buffer is guaranteed to be aligned; the default is 32 bytes.

Unlocking entire Flash memory as writeable

Beginning with release 1.4.27, the **VXBADB3_UNLOCK_ENTIRE_FLASH** driver parameter (default **FALSE**), which can be set via the Kernel Configuration of a VxWorks Image Project, allows writing to regions of on-board Flash memory that are otherwise read-only. If **TRUE**, the entirety of on-board Flash memory is made writeable. The purpose of this parameter is to enable in-system updates to the region of Flash memory that holds Bridge FPGA firmware for models that have a Bridge & Target FPGA architecture (e.g. ADM-XRC-KU1).

A new utility, **adxmrc3FlashMcs**, can be used to perform Bridge firmware updates in-system. This utility can write the contents of an Intel Hex (.mcs) file into Flash memory. Enter **adxmrc3FlashMcsProgram** in the kernel shell for help.

Known issues

Fixed-local addressing DMA transfers

The flag ADMXRC3_DMA_FIXEDLOCAL when used with the DMA functions in the ADMXRC3 currently has no effect for Gen 3 hardware.

Release history

Release 1.4.27

This release implements ADMXRC3 API Specification version 1.8.4.

Enhancements:

1. Added new boolean driver parameter **VXBADB3_UNLOCK_ENTIRE_FLASH** (default **FALSE**), which can be set via the Kernel Configuration of a VxWorks Image Project. If **TRUE**, the entirety of on-board Flash memory is made writeable. The purpose of this parameter is to enable in-system updates to the region of Flash memory that holds Bridge FPGA firmware for models that have a Bridge & Target FPGA architecture (e.g. ADM-XRC-KU1).
2. Added a mitigation for PCIe timeout errors that may occur on certain platforms when a bitstream is downloaded to a Target FPGA via the Bridge FPGA. To prevent the accumation of a backlog of PCIe writes, a PCIe read is now performed every few PCIe writes.

Release 1.4.26

This release implements ADMXRC3 API Specification version 1.8.4.

Enhancements:

1. Added support for ADM-01479.
2. Now uses 64-bit PCIe addressing by default, for models and hardware platforms that support it, which may eliminate the need for bounce-buffering, in which case a DMA performance increased might be observed.

Release 1.4.25

This release implements ADMXRC3 API Specification version 1.8.4.

Enhancements:

1. The new flags defined in ADMXRC3 API Specification version 1.8.4. are now recognized by the **ADMXRC3_Read** & **ADMXRC3_Write** API functions: **ADMXRC3_ACCESS_WIDTH_8**, **ADMXRC3_ACCESS_WIDTH_16**, **ADMXRC3_ACCESS_WIDTH_32** & **ADMXRC3_ACCESS_WIDTH_64**.
2. Added new driver parameters **PreferredReadWidth** & **PreferredWriteWidth**.

Release 1.4.24

This release implements ADMXRC3 API Specification version 1.8.3.

Enhancements:

1 Added support for ADM-AD01478.

Release 1.4.23

This release implements ADMXRC3 API Specification version 1.8.3.

Corrections:

1 Added a workaround for ADB3 Bridge bug that affects ALL models that prevents correct operation in system with Intel 3rd & 4th Generation Xeon CPUs and may crash the system when the ADB3 Driver starts. The workaround permits the driver to start without crashing the system and have sufficient functionality to permit the ADB3 Bridge firmware to be updated.

Release 1.4.21.1

This release implements ADMXRC3 API Specification version 1.8.3.

Corrections:

1 Fixed a compiler error seen when building the driver with the diab compiler, due to a type mismatch in a parameter of the function **adb3ForcePcieGen**.

Release 1.4.21

This release implements ADMXRC3 API Specification version 1.8.3.

Enhancements:

1 Added support for ADM-XRC-KU1A-T8.

2 The ADB3 Driver Utilities are now provided as an optional component that may be included in a VxWorks Image Project (VIP), to remove the need to create a separate Workbench project for them.

Release 1.4.20

Corrections:

1 The driver now starts before the usrToolsInit() function in the kernel is called, meaning that the driver should now be running before any user application that is built into the kernel and before any user-defined kernel shell startup script.

Release 1.4.19

This release implements ADMXRC3 API Specification version 1.8.3.

This is the first release of the ADB3 Driver Layer for VxWorks, and is broadly feature-equivalent to ADB3 Driver for VxWorks 1.4.19.

Address: 160 Dundee Street,
Edinburgh, EH11 1DQ, UK
Telephone: +44 131 558 2600
Email: sales@alpha-data.com
Website: http://www.alpha-data.com

Address: 10822 West Toller Drive, Suite 120,
Deer Creek Technology Center, Littleton, CO 80127
Telephone: (303) 954 8768
Email: sales@alpha-data.com
Website: http://www.alpha-data.com