ALPHA DATA

ADB3 Driver 1.4.5 for Wind River VxWorks Release Note

Introduction

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

ftp://ftp.alpha-data.com/pub/admxrcg3/vxworks

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

Operating systems supported

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

- Wind River VxWorks 5.5 and 6.x

Hardware supported

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

- ADM-XRC-6TL
- ADM-XRC-6T1
- ADM-XRC-6TGE
- ADM-XRC-6T-ADV8
- ADPe-XRC-6T and ADPe-XRC-6T-L
- ADM-XRC-II (legacy hardware)

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.

Building the driver

Prerequisites for building the driver are a Linux or Windows host machine with either Tornado 2.2 & Vxworks 5.5 or Workbench & VxWorks 6.x installed on it.

The driver is supplied in source code form so that it can be cross-built for a variety of CPU architectures and hardware platforms. To build the driver, follow the instructions in the appropriate subsection.

Cross-building the driver on a Windows host

To build the driver on a Windows host, follow these steps:

1. Unpack this package somewhere, for example

```
C:\MyTesting\adb3_drv-1.4.5
```

For convenience, the remainder of this document refers to this directory as %ROOT% (although it should be noted that no such environment variable is created nor referenced by the driver's build system).

2. Start a command prompt that is capable of performing command-line VxWorks builds. For VxWorks 6.x, use the "VxWorks Development Shell" shortcut. For VxWorks 5.5, it is necessary to start a normal command prompt, and then execute the **torVars.bat** batch file that can normally be found in

   ```
   C:\Tornado2.2\host\x86-win32\bin
   ```

3. In the command prompt, change directory to %ROOT% from step 1.
4. Execute MAKE with the appropriate options, as described in **"MAKE options"**. For example, to build a debug VxBus driver for a SMP Pentium 4 system, use

   ```
   make CONFIG=p4-vxbus-6.7 VSB=smp clean all
   ```

   This uses the predefined configuration **p4-vxbus-6.7**. Assuming the build is successful, the binaries are:

   ```
   %ROOT%\driver\monolithic\vxworks\bin\p4-vxbus-6.7\debug_smp\adb3Driver.out
   %ROOT%\api\modules\admxrc3\vxworks\bin\p4-vxbus-6.7\debug_smp\admxrc3Api.out
   ```

At this point, you are ready to proceed to starting the driver as described in **"Starting the driver"**.

Cross-building the driver on a Linux or UNIX host

To build the driver on a Linux or UNIX host, follow these steps:

TBA

MAKE options

The top-level Makefile for the ADB3 VxWorks driver accepts a number of options which are passed on the MAKE command line. These are:

- **CONFIG=<*configuration*>**
  Specifies a predefined configuration defined by the file

  ```
  %ROOT%\driver\monolithic\vxworks\rules.<configuration>
  ```

  The rules file may contain any of the following options; for an example, see

  ```
  %ROOT%\driver\monolithic\vxworks\rules.p4-vxbus-6.7
  ```

- **CPU=<*cpu*>**
  Specifies the CPU being targetted; for example PPC604 or PENTIUM4 (default). Must be appropriate for the TARGET option.
- **DEBUG=<*false*|*true*>**
  Specifies a release or debug (default) build.
- **EXTRA_CCOPTS=<*extra compiler options*>**
  Specifies extra C compiler options.
- **EXTRA_LDOPTS=<*extra linker options*>**
  Specifies extra linker options.
- **TARGET=<*target spec*>**
  Defines the target specification, which must be appropriate for the **CPU** option. Examples of valid target specifications for the DIAB toolchain are **-tPPC604FH:vxworks55** (PowerPC 604 VxWorks 5.5) and **-tPENTIUM4LH:vxworks67** (default, Pentium 4 VxWorks 6.7). Examples of valid target specifications for the GNU toolchain are **-mcpu=604** (PowerPC 604) and **-mtune=pentium4 -march=pentium4** (Pentium 4).

- **TOOLCHAIN=<*diab|gnu*>**
  Specifies the toolchain to be used to build the driver; legal values are **diab** (default) or **gnu**. If the **gnu** toolchain is selected, the following additional options must be specified (which can be in the rules file specified by the **CONFIG** option, for convenience):
  - **CC=<*compiler*>**
    Specifies the C compiler; must be appropriate for the **CPU** and **TARGET** options. For example, **ccppc** selects the PowerPC GNU compiler.
  - **LD=<*linker*>**
    Specifies the linker; must be appropriate for the **CPU** and **TARGET** options. For example, **ldppc** selects the PowerPC GNU linker.
  - **NM=<*object dumper*>**
    Specifies object dumper; must be appropriate for the **CPU** and **TARGET** options. For example, **nmppc** selects the PowerPC GNU object dump utility.
- **TYPE=<*legacy|vxbus*>**
  Specifies whether the driver should be built as a legacy driver or a VxBus driver (default).
- **VSB=<*variant*>**
  Specifies variant libraries, if required. If omitted, the normal libraries are used. The most common value for this option is **smp**.

When the **CONFIG** option is specified, the driver's build system reads a rules file that contains values for the other options. For example, the configuration **hcd5220-6.7** has a rules file

```
%ROOT%\driver\monolithic\vxworks\rules.hcd5220-6.7
```

This results in a VxBus driver targetting the Mercury HCD5220 single-board computer with SMP libraries. By way of illustration, this rules file contains:

```
CPU=PPC604
VSB=smp
TYPE=vxbus
ifeq ($(TOOLCHAIN),diab)
EXTRA_CCOPTS=-Xcode-absolute-far -Xdata-absolute-far -DHCD5220
TARGET=-tPPC604FH:vxworks67
else
ifeq ($(TOOLCHAIN),gnu)
EXTRA_CCOPTS=-mlongcall -DHCD5220
CC=ccppc
LD=ldppc
NM=nmppc
TARGET=-mcpu=604
else
$(error "TOOLCHAIN $(TOOLCHAIN) not recognized.")
endif
endif
```

If no **CONFIG** option is specified, the default configuration is **default**. The **rules.default** file contains:

```
CPU=PENTIUM4
TYPE=vxbus
ifeq ($(TOOLCHAIN),diab)
TARGET=-tPENTIUM4LH:vxworks67
else
ifeq ($(TOOLCHAIN),gnu)
CC=ccpentium
LD=ldpentium
NM=nmpentium
TARGET=-mtune=pentium4 -march=pentium4
else
$(error "TOOLCHAIN $(TOOLCHAIN) not recognized.")
endif
```

```
endif
```

It is possible that none of the predefined configurations supplied by Alpha Data is appropriate for your hardware platform. If that is the case, a new configuration can be created by using one of the existing rules files in

```
%ROOT%\driver\monolithic\vxworks
```

as a template, and modifying it appropriately.

Starting the driver

To start the driver in the target system, follow these steps:

1. Download the modules **adb3Driver.out** and **admxrc3Api.out** to the target system. This can be done using the **ld** command in the VxWorks shell or the target system's console. For example:

   ```
   -> ld <hostname:C:/MyTesting/adb3_drv-1.4.5/driver/monolithic/vxworks/bin/
   ppc604-5.5/debug/adb3Driver.out
   -> ld <hostname:C:/MyTesting/adb3_drv-1.4.5/api/modules/admxrc3/vxworks/bin/
   ppc604-5.5/debug/admxrc3Api.out
   ```

2. To start the driver, use the entry point **adb3DrvStart**:

   ```
   -> adb3DrvStart
   ```

   This entry point accepts two parameters:

   - **debugLevel** (int), default 0
     Verboseness of debug output sent to console using **logMsg**. The release version of a driver produces no output. In the debug version of the driver, a value of 0 results in minimal output and increasing values (up to 10) result in more output.
   - **bLegacyHardware** (int), default 0
     Nonzero to enable support for legacy hardware such as the ADM-XRC-II.

   For example, to start the driver with some extra debug output and support for legacy hardware, use:

   ```
   -> adb3DrvStart(2,1)
   ```

   A **debugLevel** value greater than zero in the debug version of the driver may greatly slow down execution of the driver, so 0 is recommended during normal usage.

Starting the driver with a **debugLevel** of 0 should result in output of the following form on the console:

```
-> adb3DrvStart(0,1)
0xf68b790 (tShell): adb3: dfDriverEntry: ADB3 Monolithic Driver, version=1.2.1.4
0xf68b790 (tShell): adb3: identifyPci9656: identified ADM-XRC-II
```

VPD write-protection mechanism

The VPD write-protection mechanism described in the ADM-XRC Gen 3 SDK User Guide is implemented as of release 1.1.1. To enable VPD writes, the global integer variable adb3DrvEnableVpdWrite must be set to 1 either programatically or using the VxWorks shell. This value 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) using the VxWorks shell, use:

```
-> adb3DrvEnableVpdWrite=1
```

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

```
-> adb3DrvEnableVpdWrite=0
```

Common buffer support

Beginning with release 1.4.4, the driver can create one or more "common buffers" at startup. The main purpose of this feature is supporting 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.

The driver parameter **PrimaryCommonBufferCount** determines the number of common buffers allocated; the default is zero, meaning that no common buffers are allocated by default. The parameter **PrimaryCommonBufferSizeLow** determines the size in bytes of each common buffer; the default is 64 kiB (0x10000). The parameter **PrimaryCommonBufferAlignment** determines the address boundary size to which each common buffer is guaranteed to be aligned; the default is 16 bytes (0x10).

Known issues

Fixed-local addressing DMA transfers

The flag ADMXRC3_DMA_FIXEDLOCAL currently has no effect for the ADM-XRC-6TL, ADM-XRC-6T1, ADM-XRC-6TGE and ADM-XRC-6T-ADV8 when used with the DMA functions in the ADMXRC3 API.

Compiler warnings during builds

The following compiler warnings may be generated when building the driver for certain configurations:

- "../../core/admxrc6tx_common.c", line 314: warning (dcc:1546): dangerous to take address of member of packed or swapped structure
  This warning may occur more than once in multiple source files, but can be ignored.
- "../../framework/vxworks/legacy_methods.c", line 154: warning (dcc:1500): function pciIntDisconnect2 has no prototype
  This warning can be ignored, and occurs because the **pciIntDisconnect2** function appears to be missing from the VxWorks 5.5 header files, although it exists and can be used in some VxWorks kernel images.

Release history

Release 1.4.5

This release implements ADMXRC3 API Specification version 1.5.0.

Enhancements:

1. Added common buffer functionality with the following ADMXRC3 API functions:
    - ADMXRC3_GetCommonBuffer
    - ADMXRC3_GetCommonBufferCount
    - ADMXRC3_MapCommonBuffer
    - ADMXRC3_UnmapCommonBuffer

2. Added support for the models ADPE-XRC-6T and ADPE-XRC-6T-L.

Corrections:

3. Fixed a crash that can occur when ADMXRC3_Unlock is called with an invalid value for the ADMXRC3_BUFFER_HANDLE parameter.
4. Fixed a crash when multiple queued DMA transfers are cancelled, by ADMXRC3_Cancel or by killing threads, within a small window of vulnerability around to the point at which the DMA transfer would complete normally were it not cancelled.
5. Fixed a race condition that could cause a crash every few hours of constant large DMA transfers in a typical SMP machine.
6. Corrected the scaling factors for sensors 1 to 10 for the models ADPE-XRC-6T and ADPE-XRC-6T-L.
7. Fixed a crash that can occur when attempting to do two or more DMA transfers on the same DMA channel.
8. Fixed a crash that can occur if the driver is somehow called by an ADMXRC3 API library from a different driver version, due to the handlers for ADMXRC3_GetSensorInfo and ADMXRC3_ReadSensor failing to properly validate arguments.
9. Fixed an issue specific to the ADM-XRC-6T-ADV8 where the driver emitted the debug message "*** avrInit: failed to get AVR uC firmware version".

Release 1.4.1

This release implements ADMXRC3 API Specification version 1.4.0.

Corrections:

1. Fixed a bug in the SI5338 clock synthesizer code for the ADM-XRC-6TGE that could corrupt memory when programming clock index 4. This clock generator is only available when the Si5338ExposeAllClocks driver parameter is nonzero; by default it is not available.
2. Support for ADM-XRC-6T-ADV8 is now feature-complete; added support for programming VPD and reading system monitor sensors via ADMXRC3 API.

Release 1.4.0

This release implements ADMXRC3 API Specification version 1.4.0.

Enhancements:

1. Added new API functions for performing DMA transfer to arbitrary PCI-E addresses: ADMXRC3_ReadDMABus, ADMXRC3_StartReadDMABus, ADMXRC3_StartWriteDMABus, ADMXRC3_WriteDMABus.
2. Added caching mechanism for Flash memory; reduces delays in execution of Flash API functions when performing many small write and/or erase operations.

Release 1.3.1

This release implements ADMXRC3 API Specification version 1.3.0.

New behavior:

1. Added support for the ADM-XRC-6TGE.
2. Added preliminary support for the ADM-XRC-6T-ADV8.
3. For the ADM-XRC-6TL, now recognizes "Extended" temperature range value (2) in VPD at offset 0x3E.
4. For the ADM-XRC-6T1, now recognizes "Extended" temperature range value (2) in VPD at offset 0x42.

Enhancements:

5. Now exposes (via the ADMXRC3 API) a programmable clock generator with index 0 on the ADM-XRC-6T1 when it has firmware 1.6 (PCI revision 0x06) or later.

Corrections:

6. ADMXRC3_GetClockFrequency now correctly returns the current clock frequency for a given clock generator. The driver now interrogates the hardware at startup to determine the current frequencies generated by each clock generator, so that ADMXRC3_GetClockFrequency can return the correct frequency even before any call to ADMXRC3_SetClockFrequency.
7. ADMXRC3_GetClockFrequency now correctly validates the pointer argument (3rd argument) passed to it, and returns ADMXRC3_NULL_POINTER if it is NULL.
8. Corrected the maximum frequency allowed for the clock generator with index 0 on the ADM-XRC-6TL. Previously, the driver incorrectly permitted frequencies up to 210 MHz to be requested, whereas 140 MHz is the correct maximum frequency.
9. Fixed ADMXRC3_EraseFlash failing to correctly validate the region specified to ensure that it is wholly within the unprotected region of a Flash memory bank.

Release 1.2.0

This release implements ADMXRC3 API Specification version 1.2.0.

New behavior:

1. For ADM-XRC-6TL and ADM-XRC-6T1, now recognizes "Extended" temperature range value (2) in VPD at offset 0x42.

Corrections:

2. Fixed a problem where, when a task closes a device handle that has ongoing non-blocking operations, the task may crash due to incorrect cleanup being performed by the driver.

Enhancements:

3. Added new API functions for performing DMA transfers with 64-bit local addresses:
   - **ADMXRC3_ReadDMAEx**
   - **ADMXRC3_ReadDMALockedEx**
   - **ADMXRC3_StartReadDMAEx**
   - **ADMXRC3_StartReadDMALockedEx**
   - **ADMXRC3_StartWriteDMAEx**
   - **ADMXRC3_StartWriteDMALockedEx**
   - **ADMXRC3_WriteDMAEx**
   - **ADMXRC3_WriteDMALockedEx**
4. Added support for new sensors in ADM-XRC-6TL and ADM-XRC-6T1 with firmware 1.4 or later. This provides additional sensors that show internal temperature and voltages in the PCI Express to OCP Bridge.

Release 1.1.2

Enhancements:

1. Improved the build system so that the GNU toolchain can be used for building the driver as well as the DIAB toolchain.
2. Added a rules file for building a VxBus driver for PowerPC PPC85XX and soft-floating point.

Corrections:

3. Fixed a bug where DMA transfers on the ADM-XRC-II (legacy model) do not work when the transfer is large enough to make the linked-list that describes the DMA transfer larger than a single element (may need to be greater than 16 MiB to trigger this condition).
4. Fixed a data corruption issue that can occur for the start and ending cache lines of a DMA transfer when the following conditions are met:
   - The platform does not implement hardware cache coherency for DMA transfers.
   - The DMA transfer does not start and end on a cache line boundary.
   - The DMA transfer direction is from the target FPGA to CPU memory.
5. The driver no longer uses the VxBus functions htole8/16/32/64 for endian-conversion when the driver is built as a VxBus driver because the htole64 function is not implemented by VxWorks.

Release 1.1.1

This is the first release of the ADB3 Driver for VxWorks.