ALPHA DATA

ADM-XRC LTS Driver 4.3.0b3 for Wind River VxWorks Release Note

Introduction

This release note accompanies the ADM-XRC LTS (Long-term Support) Driver for Wind River VxWorks. The latest version of this driver can be found at:

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

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

The ADM-XRC LTS Driver for Wind River VxWorks is built as a downloadable kernel module, to be downloaded to a CPU board after it has booted. There is an alternative package, ADM-XRC LTS Driver Component, which can be compiled into a VxWorks kernel image.

Operating systems supported

This release of the ADM-XRC LTS Driver for VxWorks supports the following operating systems:

- Wind River VxWorks 5.5 and 6.x

Hardware supported

This release of the ADM-XRC LTS Driver for VxWorks supports the following Alpha Data hardware:

- ADM-XRC / ADM-XRC-P
- ADM-XRC-II-L
- ADM-XRC-II
- ADM-XPL
- ADM-XP
- ADP-WRC-II
- ADP-DRC-II
- ADM-XRC-4LX / ADM-XRC-4SX
- ADCP-XRC-4LX
- ADPE-XRC-4FX
- ADM-XRC-4FX / ADM-XMC-4FX
- ADM-XRC-5LX
- ADM-XRC-5T1
- ADM-XRC-5T2 / ADM-XRC-5T2-ADV / ADM-XRC-5T2-ADV6 / ADM-XRC-5T2-ADV-CC1
- ADM-XRC-5LXA
- ADM-AMC-5A2
- ADM-XRC-5TZ
- ADC-BBP

- ADM-PCIE-6S1

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\admxrc_drv-4.3.0b3
   ```

   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 Windows 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, start a normal Windows 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 CPU=PENTIUM4 VSB=smp clean all
   ```

   In the above command, the options **DEBUG**, **TOOLCHAIN** and **TYPE** (VxBus driver vs. legacy driver) are not specified, so they default to **true**, **gnu** and **vxbus** respectively. Assuming the build is successful, the binaries are:

```
%ROOT%\driver\monolithic\vxworks\bin\vxbus_PENTIUM4gnu_debug_smp\admxrcDriver
.out
%ROOT%\api\modules\admxrc2\vxworks\bin\vxbus_PENTIUM4gnu_debug_smp\admxrc2Api
.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 ADM-XRC LTS Driver accepts a number of options which are passed on the MAKE command line. These are:

- **CPU=<*architecture*>**
  Specifies the architecture (default **PPC604**) for which the driver is to be built. In general, legal values for this can be seen by inspecting the folder **$(WIND_BASE)/target/h/tool/<*TOOLCHAIN*>**, where **TOOLCHAIN** is one of the supported toolchains (**diab**, **gnu** or **icc**). The files named **make.<CPU><TOOLCHAIN>** represent the allowed values for **CPU** and **TOOLCHAIN**. For example, for the file **make.ARMARCH7sfdiab**, **CPU=ARMARCH7** and **TOOLCHAIN=sfdiab**. See also the **TOOLCHAIN** option described below.
- **CTAG=<*configuration tag*>**
  This option, when specified, overrides the default output directory naming convention. When not specified, **CTAG** takes the value **<TYPE>_<CPU><TOOLCHAIN>_<*debug*|*release*>[_<VSB>]**.
- **CTAG_EXTRA=<*extra configuration tag*>**
  This option, when specified, is appended to **CTAG** in order to further distinguish configurations, if necessary. An underscore or other separator character is not automatically added, so the value of this option should normally begin with an underscore or hyphen.
- **DEBUG=<*false*|*true*>**
  When **true** (default), specifies a debug build in which optimizations are disabled. When **false**, specifies a release build in which normal optimizations are enabled.
- **QUIET=<*false*|*true*>**
  When **true**, the verbose display of build commands is suppressed, enabling warnings to be more easily seen during build. The default is **false**
- **SPECIAL=<*board*>**
  Specified to enable code paths for a single-board computer that requires special-case (non-generic) code in order for the driver to work correctly. Currently-supported boards are:
  - Mercury HCD5220
    Use **SPECIAL=HCD5220**
  - Motorola MVME5500
    Use **SPECIAL=MV5500**
- **TOOLCHAIN=<*gnu*|*sfgnu*|*diab*|*sfdiab*|*icc*>**
  Specifies the toolchain to be used to build the driver. This option is also used to specify whether or not the driver is built for soft-floating point (**sfgnu** or **sfdiab**).
- **TYPE=<*legacy*|*vxbus*>**
  Specifies whether the driver should be built as a legacy driver or a VxBus driver (default). If the build system detects a VxWorks 5.5 environment, **TYPE** is automatically set to **legacy**.
- **VSB=<*variant*>**
  Specifies VxWorks variant libraries, if required. If omitted, the normal libraries are used. This option **must** be specified when building the driver for a SMP system, or for a 64-bit system where the architecture also supports a 32-bit mode. Commonly-used values are **smp**, **lp64_smp** and **lp64**.

  Although VxWorks Source Builds (VSB) was a feature introduced in VxWorks 6.7, the driver makes a special case exception for VxWorks 6.6, which lacks VSB but permits SMP. In the case of VxWorks 6.6, the **VSB=smp** is permitted. This causes the driver's build system to pass -D_WRS_VX_SMP to the compiler.

Starting the driver

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

1. Download the modules **admxrcDriver.out** and **admxrc2Api.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/admxrc_drv-4.3.0b3/driver/admxrc_monolithic/vxwo
rks/bin/
legacy_PPC604gnu_debug/admxrcDriver.out
-> ld <hostname:C:/MyTesting/admxrc_drv-4.3.0b3/api/modules/admxrc2/vxworks/b
in/
```

```
legacy_PPC604gnu_debug/admxrc2Api.out
```

2 To start the driver, use the entry point **admxrcDrvStart**:

```
-> admxrcDrvStart
```

This entry point accepts one parameter:

- **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.

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

```
-> admxrcDrvStart(2)
```

A **debugLevel** value greater than zero 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:

```
-> admxrcDrvStart
0xdd7a390 (tShellRem232161468): admxrc(0): dfDriverEntry: ADM-XRC Monolithic Drive
r, version=4.3.0.8
0xdd7a390 (tShellRem232161468): admxrc(0): identifyDevice: PCI, vid=0x10b5 did=0x9
656 svid=0x4144 sdid=0x004d rev=0x02
0xdd7a390 (tShellRem232161468): admxrc(0): identifyAlphaData: identified ADM-XRC-4SX
value = 0 = 0x0
```

VPD write-protection mechanism

To enable writes to VPD memory (calls to **ADMXRC2_WriteConfig**), the configuration option **VXBADMXRC_ENABLE_VPD_WRITE** must be **TRUE**.

VXBADMXRC_ENABLE_VPD_WRITE is the initial value of the global integer variable **admxrcDrvEnableVpdWrite**. 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) using the VxWorks shell, use:

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

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

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

Known issues

Modifications to certain BSPs needed for interrupt delivery

Certain BSPs for single-board computers require a modification to an interrupt vector table in order for interrupts to be delivered to the ADM-XRC LTS Driver. If the driver behaves as if interrupts are not being delivered to it - for example, if DMA transfers hang, or the "ITest" example from the ADM-XRC SDK fails to work correctly - it may necessary to modify the file **hwConf.c** in your VxWorks kernel image project.

Although Alpha Data cannot in general provide precise instructions for doing this, for many BSPs the necessary steps are as follows:

1 Open **hwConf.c** in your VxWorks kernel image project in your favorite editor, and locate a table of this

form:

```
LOCAL const struct intrCtlrInputs loApicInputs[] = {
    { VXB_INTR_DYNAMIC, "yn", 0, 0 },
    { VXB_INTR_DYNAMIC, "gei", 0, 0 },
    ... other entries ...
#if defined (INCLUDE_HPET_MSI)
    { INT_NUM_IA_HPET_TIMER0, "iaHpetTimerDev", 0, 0 },
    ... other entries ...
#endif /* INCLUDE_HPET_MSI */
    ... other entries ...
};
```

2. If you have a single Alpha Data FPGA card in the target system, add the following entry to the end of the table:

   ```
   { VXB_INTR_DYNAMIC, "admxrc", 0, 0 }
   ```

   If you have more than one device, add as many entries as you have devices, incrementing the number in the 3rd position of each entry for each entry. For example, if there are 4 devices, add the following entries to the end of the table:

   ```
   { VXB_INTR_DYNAMIC, "admxrc", 0, 0 },
   { VXB_INTR_DYNAMIC, "admxrc", 1, 0 },
   { VXB_INTR_DYNAMIC, "admxrc", 2, 0 },
   { VXB_INTR_DYNAMIC, "admxrc", 3, 0 }
   ```

3. Rebuild your VxWorks kernel image. It should now be capable of delivering interrupts to the ADM-XRC LTS Driver.

vxbDmaBufLib symbols missing when downloading the ADM-XRC LTS Driver

If, when you download **admxrcDriver.out** to the target processor, you see messages of the form

```
Warning: module 0xffff80000023b1a0 holds reference to undefined symbol vxbDmaBufMe
mAlloc.
Warning: module 0xffff80000023b1a0 holds reference to undefined symbol vxbDmaBufMa
pCreate.
Warning: module 0xffff80000023b1a0 holds reference to undefined symbol vxbDmaBufTa
gCreate.
... other warnings ...
```

it indicates one of two things:

- The **INCLUDE_DMA_SYS** component is excluded from the kernel image. In that case, add it to the kernel image and rebuild. The ADM-XRC LTS Driver requires this component in order to be able to perform DMA transfers.
- The **INCLUDE_DMA_SYS** component is included in the kernel image but is being optimized out due to no kernel component using it. In that case, the workaround is to include a kernel component that is known to rely on **INCLUDE_DMA_SYS**, such as **INCLUDE_EHCI**. Then, rebuild the kernel image.

Compiler warnings during builds

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

- *../../framework/vxworks/vxbus_pci.c: In function 'getRawBusResources':*
  ../../framework/vxworks/vxbus_pci.c:137: warning: unused variable 'f'
  This warning arises from variables that are required in some VxWorks configurations and not in others, and can safely be ignored.
- *icc: command line warning #10006: ignoring unknown option '-Wbad-function-cast'*
  icc: command line warning #10006: ignoring unknown option '-Wno-sign-conversion'

This warning arises because the 64-bit build specs for the Intel Compiler for VxWorks set a couple of compiler options that are not supported for that particular version of the compiler. These warnings can safely 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 4.3.0b3

This is the first release of the ADM-XRC LTS Driver for VxWorks.