27 Renmar Av. Walpole, MA 02081 Web site: www.ueidaq.com E-mail: info@ueidaq.com



The High-Performance Alternative

UEIPAC VxWorks User Manual

January 2018 Edition

© Copyright 2018 United Electronic Industries, Inc. All rights reserved

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording, or otherwise without prior written permission.



Table of contents

1 Intro	duction	1
1.1 Co	nventions used in this manual	1
2 Conf	iguring and building a VxWorks kernel for UEIPAC	2
2.1 Ins	talling Software	2
2.1.1	Extract files	2
2.1.2	Configure GNU tools	2
2.1.3	Build external drivers	3
2.2 Bu	ilding a VxWorks kernel for UEIPAC	4
2.2.1	Create a VxWorks Image Project	4
2.2.2	Kernel Configuration	6
2.2.3	Build Kernel	13
2.3 Bo	oting VxWorks kernel on UEIPAC	14
2.3.1	Boot loader	14
2.3.2	Manual boot	14
2.3.3	Store kernel in flash	17
2.3.4	Automatic boot in VxWorks	18
2.4 Ad	ding pdnaserver to VxWorks image	19
2.4.1	Build pdnaserver Downloadable Kernel Module	19
2.4.2	Embed pdnaserver in your VxWorks kernel image	20
2.4.3	Running pdnaserver	21
3 Prog	ramming with PowerDNA API	22
3.1 Po	werDNA API	22
3.1.1	UEIPAC Compatible APIs	23
3.2 Bu	ilding PowerDNA library	26
3.2.1	Set-up environment	26
3.2.2	Install PowerDNA driver source and documentation	27
3.2.3	Build library using DIAB tools	27



3.2	2.4	Build library using GNU tools	28
3.3	Bui	lding an example as a kernel module	29
3.3	3.1	Creating workbench project	29
3.3	3.2	Running the example	35
3.3	3.3	Debugging the example	37
3.4	Tra	nsferring kernel module to flash drive	39
3.5	Loa	ding and running a kernel module	40



1 Introduction

This manual provides documentation for the UEIPAC VxWorks Board Support (BSP).

A UEIPAC consists of a CPU controlling multiple I/O devices. The CPU and I/O devices can come in different form factors: cubes, racks and military rugged racks.

The UEIPAC-VxWorks distribution contains a BSP to run a VxWorks kernel on the CPU, as well as drivers for the I/O devices to create applications.

1.1 Conventions used in this manual

When entering commands on the UEIPAC, note that the following conventions are used:

- commands prefixed with "=>" are entered in u-boot
- commands prefixed with "->" are entered in the VxWorks target C shell
- commands prefixed with "[vxWorks *]" are entered in the VxWorks target command mode shell



2 Configuring and building a VxWorks kernel for UEIPAC

2.1 Installing Software

2.1.1 Extract files

To extract files:

- 1. Open the VxWorks development Shell.
- 2. Copy the UEIPAC VxWorks archive:

%WIND_HOME%\vxworks-6.x\target

copy ueipac-vxworks-x.y.tgz %WIND_HOME%\vxworks-6.x\target

3. Extract archive:

3. Extract archive:

```
tar xvfz ueipac-vxworks-x.y.tgz
```

2.1.2 Configure GNU tools

WindRiver configures GCC to use the **-ansi** option by default. This disables the C++ comment style in C source code and is not compatible with the PowerDNA driver and library source code

Edit the file **\${WIND_HOME}/vxworks-6.x/target/h/tool/gnu/defs.gnu** and remove **-ansi** from the **CC_COMPILER** option.



2.1.3 Build external drivers

UEIPAC needs several external drivers that do not ship with VxWorks:

- PowerDNA: The driver for the PowerDNA/DNR I/O layers
- Bonding: A network driver that combines both network ports to implement "fault tolerance". Once the driver detects that a network port link status is down, it immediately switches traffic to the other port.

To build external drivers:

1. Change to the UEI driver directory:
cd %WIND_HOME%\vxworks-6.x\target\3rdparty\uei
2. Build the drivers:
make CPU=PPC32 TOOL=gnu
3. Copy files:

The kernel configuration tool does not automatically search for files in the %WIND_HOME%/vxworks-6.x/target/3rdparty directories. The CDF files from these directories must be manually copied to the following directory:

```
copy pdna/40pdna.cdf %WIND_HOME%\vxworks-6.x\target\config\comps\vxWorks
copy bonding/40bonding.cdf %WIND_HOME%\vxworks-
6.x\target\config\comps\vxWorks
```



2.2 Building a VxWorks kernel for UEIPAC

2.2.1 Create a VxWorks Image Project

To create a VxWorks image project:

- 1. Open WindRiver Workbench.
- 2. Select File/New/VxWorks Image Project.
- 3. Configure the project to be based on the UEIPAC834x BSP.

3		New VxWorks Imag	ge Project		_	×
Project S Base the n package a	etup wyproject either nd a tool chain.	on an existing project, o	r on a board	support	Į	
-Setup the	e project					
Based on	a board suppor	t package	•			
Project:			~	Browse		
BSP:	UEIPAC834x		~	Browse		
	Address mode:	32-bit kernel	~			
		Tool chain: gnu	~			
Target Ag	gent • WDB Target Ag	ent				
BSP valid	ation test suite upport to project	Options				
Setup inf Base direc	ormation ctory: C:/WindR	iver/vxworks-6.9/target/o	onfig/UEIPA	C834x		
?	<	<u>B</u> ack <u>N</u> ext >	<u> </u>	iish	Ca	incel



4. Select one of the profiles. **PROFILE_DEVELOPMENT** is recommended:

9	New VxWorks Image Project – 🗖	×
Configuration	on Profile	7
Profile: PRO	FILE_DEVELOPMENT	
Profile inform	nation	
Description:	VxWorks Kernel Development Configuration Profile	^
		~
synopsis:	VxWorks kernel including development and debugging components	^
		~
	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cance	el

5. Click Finish.



The High-Performance Alternative

2.2.2 Kernel Configuration

Set the kernel configuration options below to enable networking, file I/O and PowerDNA I/O layer drivers.

2.2.2.1 Symbol configuration

Enable standalone symbol table to be able to invoke C functions implemented in the PowerDNA driver without being connected to a symbol server.

```
#define INCLUDE_STANDALONE_SYM_TBL
```

2.2.2.2 Serial Console Configuration

UEIPAC uses 57600 as the default speed for its serial console.

1. Make sure the serial console is enabled:

#define INCLUDE_SIO	
2. Set the console baud rate to 57600:	
#define CONSOLE_BAUD_RATE 57600	



2.2.2.3 Network Configuration

UEIPAC is equipped with two independent Ethernet ports. You can keep them that way or enable the **bonding** driver to team them as a fault tolerant network.

To enable bonding of the two network ports:

1. Make sure bonding is enabled:

#define DRV_UEI_BONDING

2. Include ifconfig command:

#define INCLUDE_IPIFCONFIG_CMD

3. Configure independent Ethernet ports:

#define IFCONFIG_1 "ifname eth0","devname mottsec0","inet 192.168.100.2",
"gateway 192.168.100.1"
#define IFCONFIG_2 "ifname eth1","devname mottsec1","inet 192.168.101.102",
"gateway 192.168.101.1"

Or configure bonded adapters:

#define IFCONFIG_1 "ifname eth0","devname bonding0","inet 192.168.100.2", "gateway 192.168.100.1"

4. Disable remote commands (otherwise the system will try to connect to the

host each type you mistype a command):

#undef INCLUDE_NET_REM_IO

5. Enable WDB Ethernet connection to be able to attach the Workbench's debugger to the UEIPAC:

#define INCLUDE_WDB_COMM_END
#define WDB_END_DEVICE_NAME ``mottsec"
#define WDB_END_DEVICE_UNIT 0



2.2.2.4 File I/O Configuration

The UEIPAC is equipped with 32 MB of flash. Part of this flash is reserved to store the boot loader and the VxWorks kernel. 8 MB is available for general purpose file storage.

Enable **TFFS** and **DOSFS** to enable flash file storage.

```
#define INCLUDE_IO_FILE_SYSTEM
#define INCLUDE_TL_FTL
#define INCLUDE_TFFS
#define INCLUDE_TFFS_MOUNT
#define TFFS_MOUNT_POINT ``/tffs0"
#define INCLUDE_DOSFS
#define INCLUDE_DOSFS_MAIN
#define INCLUDE_DOSFS_FMT
#define INCLUDE_DOSFS_FAT
```

2.2.2.4.1 Formatting the flash drive

Format the flash drive with the C interpreter commands below:

```
-> sysTffsFormat
-> usrTffsConfig(0,0,"/tffs0")
-> dosfsDiskFormat("/tffs0")
```

INCLUDE_TFFS_MOUNT ensures that the formatted flash drive is automatically mounted at boot time.



2.2.2.5 PowerDNA I/O Configuration

Enable PowerDNA driver:

#define DRV_UEI_PDNA

2.2.2.6 USB Configuration

VxWorks comes with USB host and target support.

Detailed information regarding USB ports and USB device configuration is available in the "Wind River USB for VxWorks 6 Programmer's Guide". Refer to VxWorks' documentation in conjunction with this section. Also note that UEI only tests the host configuration.

To enable USB host support:

Enable the USB common stack (and its init function), as well as the EHCI host controller (and its init function).

You can also enable the **usbShow** command, which is very useful to list the enumeration of USB devices.

```
#define INCLUDE_USB
#define INCLUDE_USB_INIT
#define INCLUDE_EHCI
#define INCLUDE_EHCI_INIT
#define INCLUDE_USB_SHOW
```

This is sufficient to enumerate USB devices connected to the **USB A** port of UEIPAC.

To verify that the USB device is properly connected: Connect your USB device, and type **usbShow**.

The results of **usbShow** will print similar to the following:



Putty	—	×
[vxWorks *]# C usbShow bus count = 1		^
Enumerating Root Hub bus 0		
Node ID: 0x7F (Bus - 0x0 Address - 0x7F) Descriptor Type: 0x1 USB Release (BCD): 0x0200 Device Class: 0x09 Device SubClass: 0x00 Device Protocol: 0x01 Translation of above: Hub Max Packet Size 0: 0x40 (64) Vendor Code: 0x0000 Product Code: 0x0000 Device Version: 0x0000 Manufacturer Index: 0x00 ("") Product Index: 0x00 ("") Serial Number Index: 0x00 ("")		
The hub at Bus:000 has 2 available ports		
Bus:000 Port:0 A device is connected to Bus:000 Port:0. Node ID: 0x1 (Bus - 0x0 Address - 0x1) Descriptor Type: 0x1 USB Release (BCD): 0x0200 Device Class: 0x00 Device SubClass: 0x00 Device Protocol: 0x00 Translation of above: Unknown Max Packet Size 0: 0x40 (64) Vendor Code: 0x1516 Product Code: 0x1213		
Device Version: 0x0000 Manufacturer Index: 0x01 ("USB")		
Product Index: 0x02 ("DISK 2.0")		~

The screenshot above shows that a USB device with vendor ID=0x1516 and product ID=0x1213 is connected.



All USB devices require a host driver to function properly. VxWorks comes with drivers for USB mass storage devices, USB network adapters, USB serial ports and USB mice and keyboards.

Select the proper USB host device driver in FOLDER_USB_HOST_DEVICES.

For example, select the following options to use a USB mass storage device with USB2 transfer speed:

#define INCLUDE_USB_GEN2_STORAGE
#define INCLUDE_USB_GEN2_STORAGE_INIT
#define INCLUDE_EHCI
#define INCLUDE_EHCI_INIT
#define INCLUDE_USB_SHOW

USB storage devices are auto-mounted by default at **"/bd0**", **"/bd1**", etc. USB storage devices must be formatted using the FAT or FAT32 format.

You can use **dosfsDiskFormat("/bd0")** to format the drive.

To manipulate files on the USB storage, you can use the VxWorks **file** command or the standard UNIX commands (ls, cp, mv, rm, etc.).



Putty	_		×
			^
111111111111111111111111111111111111111			
111111111111111111111111111111111111111			
111111111111111111111111111111111111111			
111111111111111111111111111111111111111	1111	(R)	
1 11111111 11111 111111 11	1111		
	1111		
	1 1111 11	11111	
1111 111 11 1 111 11 11111 111111 11 11	1 1111 11	1111	
11111 1 1111 11111 11111 1111 1111	1111111	1111	
111111 11111 11111 1 1111 1111 1111	11111111	1111	
1111111 11111 1 11111 1 111 1111 1111	1111 1111	1111	
1111111 1111 111 111111 1 111111 1111	1111 111	1 11111	
111111111111111111111111111111111111111			
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]] Development System			
111111111111111111111111111111111111111			
]]]]]]]]]]]]]]]]]]]]]]]]]]]]] VxWorks 6.9			
[]]]]]]]]]]]]]]]]]]]]]]] KERNEL: WIND version 2.13			
]]]]]]]]]]]]]]]]]]]]]]] Copyright Wind River Syste	ms, Inc., 1	984-2016	5
CPU: UEIPAC 834x7 PBGA. Processor #0. Memory Size: 0x8000000 (128Mb). BSP version 6.9/1. Created: Jan 16 2018 11:01:32 ED&R Policy Mode: Deployed WDB Comm Type: WDB_COMM_END WDB: Ready.			
-> cmd			
[vxWorks *]# cd /bd0			
[vxWorks *]# ls			
UEIPAC VxWorks Manual.pdf			
Hello.txt			
[vxWorks *]# cp Hello.txt Hello2			
copying file Hello.txt -> Hello2			
[vxWorks *]# ls -1			
-rwxrwxrwx 1 0 0 563018 Jan 16 2018 U	EIPAC VxWor	ks Manua	1.
pdf			
-rwxrwxrwx 1 0 0 0 Jan 1 1980 H	ello.txt		
-rwxrwxrwx 1 0 0 0 Jan 1 1980 H	e11o2		
[vxWorks *]#			\checkmark

Note that the file creation date is not correct because the UEIPAC BSP doesn't handle the RTC timer.

UEIPAC VxWorks User Manual



The High-Performance Alternative

2.2.3 Build Kernel

Right-click on vxWorks target, and select Build Target.

The built kernel is at %WIND_HOME%\workspace\<project name>\default\vxWorks.



2.3 Booting VxWorks kernel on UEIPAC

2.3.1 Boot loader

UEIPAC comes with the u-boot boot loader pre-installed. This is the only supported boot loader.

VxWorks bootrom is not supported. U-Boot can directly load the VxWorks kernel using its built-in **bootvx** or **bootm** commands.

2.3.2 Manual boot

To manually boot:

- 1. Start a TFTP server on the host PC.
- 2. Connect serial cable between host PC and UEIPAC.
- 3. Start a serial terminal program with baud rate set to 57600.
- 4. Power-up UEIPAC and press any key to enter u-boot command prompt.
- 5. Transfer kernel image **vxWorks** to UEIPAC via TFTP:

=> setenv ipaddr <IP address of UEIPAC> => setenv serverip <IP address of TFTP server> => tftp 4000000 vxWorks 6. Configure environment variables: => setenv loadaddr 400000 => setenv bootargs mottsec(0,0)host:target/config/UEIPAC834x/vxWorks h=192.168.100.1 e=192.168.100.2 u=vxworks pw=vxworks



7. Boot VxWorks kernel:

=> bootvx	
COM6 - PuTTY	
	^
111111111111111111111111111111111111111	(R)
1 11111111 111111 11 111	(<i>/</i>
11 1111111 111111 111111 11 11 1111	
111 11111 1 111 11111111111111111111111	11111
1111 111 11 1 111 11 11111 111111 11 11	1111
11111 1 1111 11111 11111 1111 1111 11111	1111
111111 11111 111111 1111 1111 1111 11111	1111
1111111 11111 1 111111 1 111 1111 1111 1111	1 1111
11111111 1111 111 1111111 1 1111111 111 111	11 11111
111111111111111111111111111111111111111	
]]]]]]]]]]]]]]]]]]]]]]]]]]]] Development System	
111111111111111111111111111111111111111	
]]]]]]]]]]]]]]]]]]]]]]]]] VxWorks 6.9	
<pre>[]]]]]]]]]]]]]]]]]] KERNEL: WIND version 2.13</pre>	
]]]]]]]]]]]]]]]]]]]]]] Copyright Wind River Systems, Inc.,	1984-2013
CPU: UEIPAC 834X/ PBGA. Processor #U.	
Memory Size: 0x8000000 (128Mb). BSP Version 6.9/0.	
EDER Boligy Mode: Deployed	
WDB Comm Type: WDB COMM END	
WDB: Ready.	
->	*



8. Type the C interpreter command **pdnaShow** to verify that the PowerDNA driver is properly loaded.

🛃 COM6 - PuT	TY									x
1111111111 111111111	11111	1111111 1111111	11111 1111	KH Cor	CRNEL: WIND oyright Win) vers d Riv	ion 2.13 er Systems,	Inc.,	1984-201	13
CPU: UEIPA Memory Siz Created: H ED&R Polic	AC 83 ze: 0 Feb 1 cy Mo	4x7 PBG x800000 1 2014 de: Dep	A. Pro 0 (128M 16:08:5 0loyed	cessor b). BS 7	#0. SP version	6.9/0				
WDB Comm 7 WDB: Ready	Type: Y•	WDB_CO	MM_END							
-> pdnaShov pDrvCtrl @	v 0x0									
Address	Irq	Model	Option	Phy/Vi	irt S/N	Pri	LogicVer			
0xa0000000	2	207	1	phys	0036952	0	02.0e.00			
0xa0010000	2	208	1	phys	0030976	0	02.0e.00			
0xa0030000	2	225	1	phys	0046723	0	02.0e.00			
0xa0040000	2	211	1	phys	0049156	0	02.0f.41			
0xa0050000	2	503	1	phys	0036459	0	02.0f.32			
0xa0060000	2	448	1	phys	0039388	0	02.0e.00			
0xa0070000	2	225	801	phys	0043974	0	02.0e.00			
0xa0080000	2	508	1	phys	0039221	0	02.0f.3c			
0xa0090000	2	566	1	phys	0056782	0	02.0e.0c			
0xa00a0000	2	602	1	phys	0071299	0	02.10.76			
0xa00b0000	2	212	1	phys	0092424	0	02.10.d9			
0xa00c0000	2	20	1	phys	0037308	0	02.0d.09			
0xa00d0000	3	40	1	phys	0049634	0	02.0f.33			
0xa00e0000	2	5	1	cpu	0049816	0	02.0f.7c			=
value = 0 =	= 0x0									•

UEIPAC VxWorks User Manual



2.3.3 Store kernel in flash

2.3.3.1 Convert the VxWorks kernel image to a u-boot image

To convert the VxWorks kernel image:

1. Under Linux install mkimage:

sudo apt-get install uboot-mkimage

Under Windows use mkimage.exe that comes along with the UEIPAC VxWorks distribution.

2. Compress VxWorks kernel:

\$ cp vxWorks uVxWorks \$ gzip -v9 uVxWorks 3. Create the u-boot image: \$ mkimage -0 vxworks -C gzip -n 'UEIPAC VxWorks' -a 4000000 -d uVxWorks.gz uVxWorks



2.3.3.2 Transfer image to UEIPAC

The kernel will be stored at address 0xfe200000 in flash.

To transfer the image to UEIPAC:

1. Erase sectors:

=> erase fe200000 fe3fffff

2. Transfer kernel image and copy to flash:

```
=> tftp 4000000 uVxWorks
=> cp.b 4000000 fe200000 ${filesize}
```

2.3.3.3 Boot image

Boot kernel: => bootm fe200000

2.3.4 Automatic boot in VxWorks

Configure u-boot to automatically run VxWorks:

```
=> setenv bootcmd bootm fe200000
=> saveenv
```



2.4 Adding pdnaserver to VxWorks image

PowerDNA server emulates the behavior of a PowerDNA IO module running the standard DAQBIOS firmware. It emulates a subset of the DAQBIOS protocol so that the UEIPAC can be accessed from PowerDNA Explorer or remotely controlled using the PowerDNA C API.

The pdnaserver only works in immediate, RTDMAP and RTVMAP modes. ACB, Messaging and Asynchronous modes are not supported.

The PowerDNA server source is delivered as a tgz archive.

2.4.1 Build pdnaserver Downloadable Kernel Module

Refer to instructions in section 3.3.1 to create a **VxWorks Downloadable Kernel Module** project to build the pdnaServer source.



2.4.2 Embed pdnaserver in your VxWorks kernel image

To embed pdnaserver, do the following in the Windriver Workbench:

1. Right-click your pdnaserver project and select **Project References**, and then **Add as Project Reference**.



2. Select your kernel image project, click OK and re-build the kernel.



2.4.3 Running pdnaserver

Type the C interpreter command **pdnaServer** to start the PowerDNA server:



You can now use PowerDNA Explorer to monitor your UEIPAC-VX remotely.



3 Programming with PowerDNA API

3.1 PowerDNA API

The PowerDNA library implements the API used to program the PowerDNA IO layers.

The source code is installed in "<PowerDNA SDK directory>/src/DAQLib". Examples are located in "<PowerDNA SDK directory>/src/DAQLib_Samples".

The UEIPAC for VxWorks supports a subset of the PowerDNA API. It also allows you to control other IO modules that run the standard DAQBios firmware from the UEIPAC the same way you would from a host PC running Windows or Linux.

The PowerDNA API uses the IP address specified in the function DqOpenIOM() to determine whether you wish to access the layers local to the UEIPAC or "remote" layers installed in a remote PowerDNA IO module. Set the IP address to the loopback address "127.0.0.1" and the API will know that you want to access the "local" layers.

The PowerDNA API implements various modes to communicate with the I/O layers:

- Immediate: It is the easiest mode for point by point input/output on all layers. It also is the least efficient because it requires one call for each incoming and/or outgoing request. You cannot achieve maximum performances with that mode Immediate mode examples are named "SampleXXX"
- Data Mapping (DMAP): This is the most efficient mode for point by point input/output on AI, AO, DIO and CT layers. Incoming and outgoing data from/to multiple layers are all packed in a single call.
 DMAP mode examples are named "SampleDMapXXX"
- Buffered (ACB): Allows access to AI, AO, DIO and CT layers at full speed. It is designed to correct communication errors that might happen on the network link. The error correction mechanism will cause issues with real-time deadlines

ACB mode examples are named "SampleACBXXX"



 Messaging: Allows access to messaging layers (serial, CAN, ARINC-429) at full speed. It is designed to correct communication errors that might happen on the network link. The error correction mechanism will cause issues with real-time deadlines

Messaging mode examples are named "SampleMsgXXX"

- Variable Size Data Mapping (VMAP): Allows access to all layers at full speed, transferring incoming and outgoing data in buffers in one call.
 VMAP mode examples are named "SampleVMapXXX"
- Asynchronous: Allows I/O layers to asynchronously notify the user application upon hardware events.

The UEIPAC only supports the immediate (also known as "point by point") DMAP and VMAP modes to control the "local" layers.

The other modes (ACB and MSG) are designed to work over Ethernet and have built-in error correction which is not needed on the UEIPAC. You can, however, use those modes to control "remote" layers installed in I/O modules that run the DAQBios firmware over the network.

	Firmware Running on the IO Module								
I/O mode	DAQBios	UEIPAC (Local lavers)	UEIPAC (Remote lavers)						
Immediate	Yes	Yes	Yes						
ACB	Yes	No	Yes						
DMAP	Yes	Yes	Yes						
MSG	Yes	No	Yes						
VMAP	Yes	Yes	Yes						
Asynchronous	Yes	No	Yes						

3.1.1 UEIPAC Compatible APIs

The following section details the subset of PowerDNA APIs available when running your program on a UEIPAC.

Refer to the "PowerDNA API Reference Manual" document to get detailed information about each API.



3.1.1.1 Initialization, miscellaneous API

The following APIs are used to initialize the library, obtain a handle on the kernel driver and perform miscellaneous tasks, such as translating error codes to readable messages.

- DqInitDAQLib
- DqCleanUpDAQLib
- DqOpenIOM
- DqCloseIOM
- DqTranslateError
- All DqCmd*** APIs

3.1.1.2 Immediate mode API

Immediate Mode APIs are used to read/write I/O layers in a software-timed fashion. They are designed to provide an easy way to access I/O layers at a nondeterministic pace.

Each I/O layer comes with its own set of immediate mode APIs. For example, you will use the DqAdv204*** APIs to control an AI-204.

Most DqAdvXYZ*** APIs where XYZ is the model number of a supported I/O layer are supported on the UEIPAC.

3.1.1.3 DMAP API

In DMAP mode, the UEIPAC continuously refreshes a set of channels that can span multiple layers at a specified rate paced by a hardware clock. Values read from or written to each configured channel are stored in an area of memory called the DMAP. At each clock tick, the firmware synchronizes the DMAP values with their associated physical channels.

Supported APIs to use RTDMAP mode are DqRtDmap***.

The following is a quick tutorial on using the RTDMAP API (handling of error codes is omitted):

Initialize the DMAP to refresh at 1000 Hz: DqRtDmapInit(handle, &dmapid,1000.0);



Add channel 0 from the first input subsystem of device 1:

chentry = 0; DqRtDmapAddChannel(handle, dmapid, 1, DQ_SSOIN, &chentry, 1);

Add channel 1 from the first output subsystem of device 3: chentry = 1; DqRtDmapAddChannel(handle, dmapid, 3, DQ_SS00UT, &chentry, 1);

Start all devices that have channels configured in the DMAP:

DqRtDmapStart(handle, dmapid);

Update the value(s) to output to device 3:

outdata[0] = 5.0; DqRtDmapWriteScaledData(handle, dmapid, 3, outdata, 1);

Synchronize the DMAP with all devices:

DqRtDmapRefresh(handle, dmapid);

Retrieve the data acquired by device 1:

DqRtDmapReadScaledData(handle, dmapid, 1, indata, 1);

Stop the devices and free all resources:

DqRtDmapStop(handle, dmapid); DqRtDmapClose(handle, dmapid);

3.1.1.4 VMAP API

In VMAP mode, the UEIPAC continuously acquires/updates data in buffers.

Each layer is programmed to acquire/update data to/from its internal FIFO at a rate paced by its hardware clock.

The content of all the layers' FIFOs is accessed in one operation.

Supported APIs that use VMAP mode are DqRtDmap*** and DqRtVmap***.

The following is a quick tutorial on using the RTVMAP API (handling of error codes is

omitted):

Initialize the VMAP to acquire/generate data at 1kHz: DqRtVmapInit(handle, vmapid, 1000.0);



Add channels from the first input subsystem of device 0:

int channels = {0, 1, 2, 3 }; DqRtVmapAddChannel(handle, vmapid, 0, DQ_SSOIN, channels, flags, 1);

Start all devices that have channels configured in the VMAP:

DqRtVmapStart(handle, vmapid);

Specify how much input data to transfer during the next refresh.

DqRtVmapRqInputDataSz(handle, vmapid, 0, numScans*sizeof(uint16), &act_size, NULL);

Synchronize the VMAP with all devices:

DqRtVmapRefresh(handle, vmapid);

Retrieve the data acquired by device 0:

```
DqRtVmapGetInputData(handle, vmapid, 0, numScans*sizeof(uint16), &data_size,
&avl_size, (uint8*)bdata);
```

Stop the devices and free all resources:

```
DqRtVmapStop(handle, vmapid);
DqRtVmapClose(handle, vmapid);
```

3.2 Building PowerDNA library

3.2.1 Set-up environment

To set up your host environment:

1. Open a WindRiver Development Shell.

Under Linux:

/WindRiver/wrenv.linux -p vxworks-6.x

Under Windows:

```
C:\WindRiver\wrenv.exe -p vxworks-6.x
```



3.2.2 Install PowerDNA driver source and documentation

Untar PowerDNA library for VxWorks:

gunzip PowerDNA_vxWorks_4.x.y.z.tgz
tar xvf PowerDNA_vxWorks_4.x.y.z.tar

3.2.3 Build library using DIAB tools

1. Build library:

cd PowerDNA_4.x.y/src make CPU=PPC32 TOOL=diab

2. Change to the examples directory:

cd DAQLib_Samples

3. Build examples:

To build all examples:

make CPU=PPC32 TOOL=diab

To only build one example:

make CPU=PPC32 TOOL=diab Sample204/Sample204.out



3.2.4 Build library using GNU tools

WindRiver configures GCC to use the **-ansi** option by default. This disables C++ comment style in C source code and is not compatible with the PowerDNA library source code.

Edit the file **\${WIND_HOME}/vxworks-6.x/target/h/tool/gnu/defs.gnu** and remove **-ansi** from the **CC_COMPILER** option.

1. Build library:

cd PowerDNA_4.x.y/src make CPU=PPC32 TOOL=gnu	
2. Change to the examples directory:	
cd DAQLib_Samples	
3. Build examples	
To build all examples:	
make CPU=PPC32 TOOL=gnu	
To only build one example:	
make CPU=PPC32 TOOL=gnu Sample204/Sample204.out	



3.3 Building an example as a kernel module3.3.1 Creating workbench project

To create a workbench project:

1. Open WindRiver Workbench and create a new project.

2. Click Next and set Build type to Downloadable Kernel Module:

9	New Wind River Workbench Project – 🗖 🛛 🗙
Build Type Select the b	uild type for the project.
Build type:	Downloadable Kernel Module
Description:	Creates a munched, downloadable image, which can be dynamically linked to the operating system at run time and executes in kernel memory space in kernel mode.
Setup infor Resulting p Uses build t	mation roject type: Downloadable Kernel Module Project ool: Linker
?	< <u>Back Next > Einish Cancel</u>

3. Click **Next** and give the project a name.



3	New Wind River Workbench Project – 🗖 🗙
Project Create a ne	project with the specified data.
<u>P</u> roject nan	e: SampleRTDMap
Location	
○ Create	oject in <u>w</u> orkspace
○ Create	oject at external location
Create	oject in workspace with content at external location
Directory:	A_vxWorks_4.7.1\src\DAQLib_Samples\SampleRTDMAP v Browse
?	< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

4. Set **Location** to use source files at an external location and browse to the folder of the example you wish to build.

You can also import the source file(s) to your project later.

5. Click **Finish** to complete the project creation.



6. Right-click on the project and select Properties.

7. Select Build Properties:

9	Properties for SampleRTDMap	_ 🗆 🗙
type filter text	Build Properties	⇔ ▼ ⇔ ▼
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Analyzer Project Info Project References Run/Debug Settings Task Tags Validation 	Specify all build properties. Image: Build Support and Specs Build support Image: Build Support and Specs Image: Build Support and Specs Image: Build Support and Specs Specific Spe	riables Variables
	Available and enabled build specs: PPC32diab_SMP PPC32e500v2diab PPC32e500v2diab_SMP PPC32e500v2gnu PPC32e500v2gnu_SMP PPC32gnu_SMP PPC32gnu_SMP PPC32sfdiab PPC32sfdiab_SMP PPC32sfgnu	 ▲ Enable All Disable All New Import Rename Copy Delete
	Default build spec: PPC32gnu V Default debug mode: as specified in preferences V Active build spec: PPC32gnu V Image: Debug mode Restore Defaults Restore Defaults	Apply
?	OK	Cancel

- 8. Deselect all build specs and select PPC32gnu and PPC32diab.
- 9. Set default build spec to either PPC32gnu or PPC32diab.



10.Select the build **Tools** tab.

۹	Properties for SampleRTDMap – 🗖 💌
type filter text	Build Properties 🗘 🖛 🗘
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Analyzer Project Info Project References Run/Debug Settings Task Tags Validation 	Specify all build properties. Image: Specify all build Support and Specs Image: Specify all
	Build spec specific settings Active build spec: PPC32gnu Derived suffix: *.o Command: echo "building \$@";%ccompilerprefix% \$(TOOL_PATH)ccppc %DebugModeFlags% %ToolFlags% %Defines% \$(DEFINES) \$(ADDED_CFLAGS) %Includes% \$(ADDED_INCLUDES) -o %OutFile% -c %InFile% Tool Flags Image: S(CC_ARCH_SPEC) ansi -fno-zero-initialized-in-bss -Wall -MD -MP
	Debug mode flags Debug mode P -g Non Debug mode P -02 -fstrength-reduce -fno-builtin Restore Defaults Apply
?	OK Cancel

11.Remove the **-ansi** compiler flag from **Tool Flags...**



	Properties for SampleRTDMap	_ D >
type filter text	Build Properties	⇔ → ⇒ →
type filter text Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Analyzer Project Info Project References Run/Debug Settings Task Tags Validation 	Specify all build properties.	s \$ Variables Generate Edit Add Add to all Remove Remove from all Up
	Redirection directory: PPC32gnu Redirection root directory: Image: Comparison of the sources of the source	Default Default Browse nter an absolute path Defaults Apply K

- 12. Select the build Paths tab and click Add...
- 13.Browse to the <PowerDNA driver>/src/UeiPalLib directory and click OK.
- 14.Click Add... again, and browse to the <PowerDNA driver>/src/DAQLib directory, and click OK.



9	Properties for SampleRTDMap	_ 🗆 ×
type filter text	Build Properties	← ▼ ⇒ ▼ ▼
 Resource Binary Parser Build Properties Builders C/C++ General Code Coverage Analyzer Project Info Project References Run/Debug Settings Task Tags Validation 	Specify all build properties.	Variables Edit Add Add to all Remove Remove from all Up Down
	Restore <u>E</u>	efaults <u>A</u> pply
?	ОК	Cancel

15. Select the Libraries tab and click Add...

- 16. Select Add full qualified library file.
- 17.Type

\$(WIND_BASE)/target/lib/\$(VX_CPU_FAMILY)/\$(CPU)/common/libPDNA.a

- 18.Click OK.
- 19.Build the project.



3.3.2 Running the example

3.3.2.1 Connect to UEIPAC target server

To connect to UEIPAC target server:

- 1. In Remote Systems, click New Connection.
- 2. Select Target Server Connection and click Next:

3	New Connection	_ 🗆 🗙
Select Remot	te System Type	_
System type:		
type filter text		
▷ General ▲ ▷ VxWorks ♥ Wind ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥	s 6.x d River VxWorks 6.x Core Dump Connection d River VxWorks 6.x Simulator Connection d River VxWorks 6.x Target Server Connection	
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel



3. Enter the IP address of UEIPAC and click Finish:

		New Connection				
arget Ser	ver Options					
(eview and (customize the ta	arget server options.				
Backend se	ettings					
Processor:	(default from	target)				Select
Backend:	wdbrpc					~
Target nan	ne or address:	192.168.100.2	~	Check	Port:	
-						
Kernel ima File path	ge h from target (if	available)				
Kernel ima File path File:	ge h from target (if	available)			Y	Browse
Kernel ima File path File: Bypass	ge h from target (if checksum com	f available) parison			¥	Browse
Kernel ima File path File: Bypass Advanced	ge h from target (if checksum com target server op	f available) parison ptions			~	Browse
Kernel ima File path File: Bypass Advanced Verbose	ge h from target (if checksum com target server op target server o	f available) parison otions utput			¥	Browse
Kernel ima File path File: Bypass Advanced Verbose Options:	ge n from target (if checksum com target server op target server o -R C:/WindRive	f available) parison otions utput r/workspace -RW -Bt 3			¥	Browse • Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L	ge n from target (if checksum com target server op target server o -R C:/WindRive ine:	f available) parison otions utput r/workspace -RW -Bt 3			×	Browse Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L tgtsvr -V -R	ge h from target (if checksum com target server op target server o -R C:/WindRive ine:	f available) parison otions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2			¥	Browse Edit
Kernel ima File path File: Bypass of Advanced Verbose Options: Command L tgtsvr -V -R	ge h from target (if checksum com target server op target server o -R C:/WindRive ine: C:/WindRiver/1	f available) parison otions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2			V	Browse Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L tgtsvr -V -R	ge h from target (if checksum com target server op target server o -R C:/WindRive ine: : C:/WindRiver/1	f available) parison otions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2				Browse Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L tgtsvr -V -R	ge n from target (if checksum com target server op target server o -R C:/WindRive ine: C:/WindRiver/1	f available) parison otions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2				Browse Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L tgtsvr -V -R	ge n from target (if checksum com target server op target server o -R C:/WindRive ine: C:/WindRiver/1	f available) parison otions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2				Browse Edit
Kernel ima File path File: Bypass Advanced Verbose Options: Command L tgtsvr -V -R	ge n from target (if checksum com target server op target server o -R C:/WindRive ine: C:/WindRiver/1	f available) parison ptions utput r/workspace -RW -Bt 3 workspace -RW -Bt 3 192.168.100.2				Browse



3.3.2.2 Create Run configuration

To create run configuration:

- 1. Select Menu option Run/Run Configurations...
- 2. Select VxWorks Kernel Task, and click New Launch Configuration.
- 3. Check the connection associated with your UEIPAC, and enter the name of the entry point (all PowerDNA examples use **main**).
- 4. Click **Run** to download and start the program on your UEIPAC.

3.3.3 Debugging the example

To debug the example, do the following:

1. Select the Menu option: Run/Debug Configurations...



3		Debug Configurations		×
Create, manage, and run con Select the launch context for execut	figurations ing the launch operations.			Ť.
Image: System VxWorks 6.9 (System VxWorks 6.9 (System VxWorks 6.9 (System VxWorks 6.9) Image: System VxWorks 6.9 Image: System VxWorks Control Image: System VxWorks Kernel Task Image: System VxWorks Real Time Process Image: System VxWorks Real Time Process	Name: main - SampleRTI	DMap.out - VxWorks6x_192.168.100.2 Downloads Projects to Build Source Task Options alhost) (Wind River VxWorks 6.9) smp (Wind River VxWorks 6.9) s6x_192.168.100.2 (Wind River VxWorks 6.9) s6x_192.168.100.6 (Wind River VxWorks 6.9) in ntrols might become edit able for a connected laur Task 100 0_20000	Common	■ & +++++++++++++++++++++++++++++++++++
Filter matched 17 of 17 items			Apply	Revert
?			<u>D</u> ebug	Close

2. Select the Launch Configuration created the first time you ran the example, and click **Debug**.



3.4 Transferring kernel module to flash drive

The default VxWorks image includes the FTP client command to copy remote files to the local flash storage.

Make sure you have an FTP server running on your development PC and transfer the kernel module using the commands below (only type commands in bold):

[vxWorks *] cd /tffs0 [vxWorks *] ftp 192.168.100.101 (use your host PC IP address) 220 (vsFTPd 3.0.3) Connected to 192.168.100.101 User: frederic frederic Password: 230 Login successful. 200 EPSV ALL ok. ftp> bin bin 200 Switching to Binary mode. ftp> get Sample204.out get Sample204.out 229 Entering Extended Passive Mode (|||14462|) 150 Opening BINARY mode data connection for Sample204.out (660906 bytes). 226 Transfer complete. ftp> quit quit 221 Goodbye. [vxWorks *]# Is Sample204.out



3.5 Loading and running a kernel module

To load and run a kernel module:

1. Use the 1d command to load the kernel module:

[vxWorks *] Id Sample204.out

2. Run the C function implemented in the kernel module.

A typical VxWorks kernel module comes with start and stop C functions.

The PowerDNA examples are regular C programs that implement a main() function.

They are not designed specifically for VxWorks but you can still run them with the command below, which tells the C shell to run main():

[vxWorks *] C main

Putty		_ ×
[vxWorks *] # 1s		-
Sample204.out		
[vxWorks *] # 1d Sample204.out		
0xdb9030		
[vxWorks *] # C main		
ipaddr = 0.0.0.0		
model = 3012		
sernum = 0162792		
mfgdate = 11/2/2017		
caldate = 11/2/2017		
Model: 204 Option: 1		
Model: 650 Option: 1		
Model: 364 Option: 1		
Model: 207 Option: 1		
Model: 20 Option: 1		
Model: 40 Option: 1		
0: cl 0 flags 3f bdata 800e0000	fdata 0.000000 mA	
1: CI I Flags 3F bdata 800e0000	fdata 0.000000 mA	
2: CI 2 Flags SI bdata 800e0000	fdata 0.000000 mA	
4. cl 4 flags 31 bdata 800e0000	fdata 0.000000 mA	
4: CI 4 Flags SI bdata 80080000	fdata 0.000000 mA	
6. cl 6 flags 3f bdata 800e0000	fdata 0.000000 mA	
7. cl 7 flags 3f bdata 800e0001	fdata 0.000005 mA	
8: cl 8 flags 3f bdata 800e0000	fdata 0.000000 mA	
9: cl 9 flags 3f bdata 800e0000	fdata 0.000000 mA	
10: cl 10 flags 3f bdata 800e0001	fdata 0.000095 mA	
11: cl 11 flags 3f bdata 800e0001	fdata 0.000095 mA	

There is no clean way to stop the PowerDNA examples; you can press ^C to stop.