SL100/SL240
Software Installation Manual
for Tornado® 2.2 and VxWorks® 5.31 - 5.5
Using the PCI, PMC, or CPCI Cards

Document No. F-T-MI-VWXXGS21-A-0-A5

CURTISS WRIGHT Controls, Inc.
Embedded Computing

FOREWORD

The information in this document has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. Curtiss-Wright Controls, Inc. reserves the right to make changes without notice.

Curtiss-Wright Controls, Inc. makes no warranty of any kind with regard to this printed material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

FibreXtreme® is a registered trademark of Systran Corporation A Division of Curtiss-Wright Controls Embedded Computing Group.
Intel® is a registered trademark of Intel Corporation.
Motorola® is a registered trademark of Motorola, Inc.
PowerPC® is a registered trademark of IBM Corp.
Sun, Sun Microsystems, Sun WorkShop Compilers C, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc.
VxWorks® is a registered trademark of Wind River Systems.
Tornado® is a registered trademark of Wind River Systems.
Windows NT® is a registered trademark of the Microsoft Corporation.

Any reference made within this document to equipment from other vendors does not constitute an endorsement of their product(s).

Revised: August 5, 2004

Curtiss-Wright Controls Inc. Embedded Computing
Data Communications Center
4126 Linden Avenue
Dayton, OH 45432-3068
USA (800) 252-5601(U.S. only)
(937) 252-5601

TABLE OF CONTENTS

APPENDICES

FIGURES

TABLES

1. INTRODUCTION

1.1 How to Use This Manual

1.1.1 Purpose

This manual describes the FibreXtreme SL100/SL240 software installation process for Tornado 2.2 and VxWorks 5.3.1-5.5. Reference to VxWorks throughout the manual implicitly includes VxWorks 5.3.1-5.5 as appropriate. For information common to all platforms consult the *FibreXtreme SL100/SL240 API Guide.*

NOTE: Both the FibreXtreme SL100 and SL240 hardware will be referred to throughout this manual as SL240. The software that supports both the SL100 and SL240 hardware will also be referred to as SL240, including the driver and API. Anything that is exclusive to the SL100 or the SL240 will be described as such.

NOTE: Please read this entire document before attempting to install the SL240 software.

1.1.2 Scope

This manual contains the following information:

- Instructions for installing the software.
- Instructions for configuring, installing, and using the SL240 device driver.
- Location and organization of the example applications.
- Supported Targets and rehosting considerations.

You should have a basic understanding of Tornado 2.2 and VxWorks, compiling procedures, and loading and executing programs to effectively use this manual.

1.1.3 Style Conventions

- Called functions are italicized. For example, *OpenConnect()*
- Data types are italicized. For example, *int*
- Function parameters are bolded. For example, **Action**
- Path names are italicized. For example, *utility/sw/cfg*
- File names are bolded. For example, **config.c**
- Path file names are italicized and bolded. For example, ***utility/sw/cfg/config.c***
- Hexadecimal values are written with a "0x" prefix. For example, 0x7e
- For signals on hardware products, an 'Active Low' is represented by prefixing the signal name with a slash (/). For example, /SYNC
- Code and monitor screen displays of input and output are boxed and indented on a separate line. Text that represents user input is bolded. Text that the computer displays on the screen is not bolded. For example:

```
ls
file1          file2              file3
```

- Large samples of code are Courier font, at least one size less than context, and are usually on a separate page or in an appendix.

1.2 Related Information

- *FibreXtreme SL100/SL240 Hardware Reference Manual for PCI and PMC Cards*, Systran Corporation A Division of Curtiss-Wright Controls Embedded Computing Group.
- *VMetro User's Manual PowerMIDAS -S Series PMC I/O Subsystem for VMEbus and RACE++,* Revision 3.0, dated December 18, 2000.
- *FibreXtreme SL100/SL240 API Guide*, Curtiss-Wright Controls, Inc.
- Curtiss-Wright Controls, Inc. web site: www.cwcembedded.com.

1.3 Quality Assurance

Curtiss-Wright Controls' policy is to provide our customers with the highest quality products and services. In addition to the physical product, the company provides documentation, sales and marketing support, hardware and software technical support, and timely product delivery. Our quality commitment begins with product concept, and continues after receipt of the purchased product.

Curtiss-Wright Controls' Quality System conforms to the ISO 9001 international standard for quality systems. ISO 9001 is the model for quality assurance in design, development, production, installation, and servicing. The ISO 9001 standard addresses all 20 clauses of the ISO quality system, and is the most comprehensive of the conformance standards.

Our Quality System addresses the following basic objectives:

- Achieve, maintain, and continually improve the quality of our products through established design, test, and production procedures.
- Improve the quality of our operations to meet the needs of our customers, suppliers, and other stakeholders.
- Provide our employees with the tools and overall work environment to fulfill, maintain, and improve product and service quality.
- Ensure our customer and other stakeholders that only the highest quality product or service will be delivered.

The British Standards Institution (BSI), the world's largest and most respected standardization authority, assessed Curtiss-Wright Controls' Quality System. BSI's Quality Assurance division certified we meet or exceed all applicable international standards, and issued Certificate of Registration, number FM 31468, on May 16, 1995. The scope of Curtiss-Wright Controls' registration is: "Design, manufacture and service of high technology hardware and software computer communications products." The registration is maintained under BSI QA's bi-annual quality audit program.

Customer feedback is integral to our quality and reliability program. We encourage customers to contact us with questions, suggestions, or comments regarding any of our products or services. We guarantee professional and quick responses to your questions, comments, or problems.

1.4 Technical Support

Technical documentation is provided with all of our products. This documentation describes the technology, its performance characteristics, and includes some typical applications. It also includes comprehensive support information, designed to answer any technical questions that might arise concerning the use of this product. We also publish and distribute technical briefs and application notes that cover a wide assortment of topics. Although we try to tailor the applications to real scenarios, not all possible circumstances are covered.

Although we have attempted to make this document comprehensive, you may have specific problems or issues this document does not satisfactorily cover. Our goal is to offer a combination of products and services that provide complete, easy-to-use solutions for your application.

If you have any technical or non-technical questions or comments, contact us. Hours of operation are from 8:00 a.m. to 5:00 p.m. Eastern Standard/Daylight Time.

- Phone: (937) 252-5601 or (800) 252-5601
- E-mail: support@systran.com
- Fax: (937) 252-1349
- World Wide Web address: www.cwcembedded.com

1.5 Ordering Process

To learn more about Curtiss-Wright Controls Embedded Computing products or to place an order, please use the following contact information. Hours of operation are from 8:00 a.m. to 5:00 p.m. Eastern Standard/Daylight Time.

- Phone: (937) 252-5601 or (800) 252-5601
- E-mail: info@systran.com
- World Wide Web address: www.cwcembedded.com

This page intentionally left blank.

2. SOFTWARE OVERVIEW

2.1 Overview

The FibreXtreme SL240 device driver supports the SL100/SL240 PCI, PMC, and CPCI cards. The software provides an easy-to-use, very high-speed point-to-point communication link.

NOTE: The SL240 device driver can support between one and sixteen SL240 cards per host.

NOTE: The SL100 and SL240 cards do not, by design, communicate with each other. For more information on SL100/SL240 hardware issues, refer to the *FibreXtreme SL100/SL240 Hardware Reference Manual for PCI, PMC, and CPCI Cards*.

CAUTION: Do not attempt to install both the FibreXtreme Simplex Link and the SL100/SL240 on the same system.

2.2 Software Distribution

The SL240 software for Tornado 2.2 and VxWorks is distributed on one CD-ROM. The software contains an SL240 API (Application Programming Interface) library, and a SL240 device driver.

NOTE: The SL240 applications are distributed on a separate CD-ROM. Consult the SL240 API Guide for installation instructions.

2.3 System Requirements

The minimum system requirements include the following:

- Tornado 2.2 or VxWorks host configured according to the WindRiver specifications.
- 3.0 MB of available hard drive space.
- C compiler for rebuilding API library and applications.
- One or more SL100 or SL240 cards.

For a list of supported targets, examine the readme file located in the root of the CD-ROM.

This page intentionally left blank

3. INSTALLATION

3.1 Overview

3.1.1 Chapter Outline

This chapter describes the installation process for the SL240 software. Before installing the SL240 software, install the hardware as described in the appropriate Curtiss-Wright Controls FibreXtreme SL100/SL240 Hardware Reference Manual.

When you have ensured the hardware is correctly installed, install the software using the following steps:

- Install the software onto the host computer (section 3.2).
- Verify the directory structure (section 3.3).
- Install and configure the SL240 device driver (section 3.4).

3.1.2 Software Organization

There are several working "layers" within the SL240 system. These layers vary slightly between operating systems. Figure 3-1 shows a simplistic view of these layers.

SL240 API.....Library containing call-routines for executing calls to the device driver.

SL240 Device Driver .....A kernel-level VxWorks character device driver.

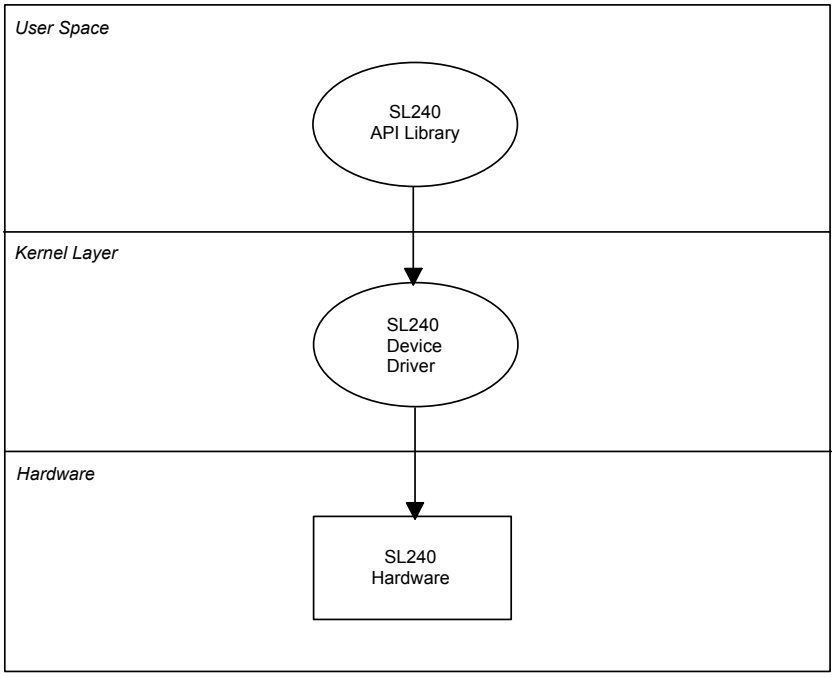

Figure 3-1 SL240 Structure

3.2 Install the Software on the Host Computer

The SL240 software installation procedures for Tornado 2.2/VxWorks depend on whether the host development system is a Solaris or a Windows workstation. Loading procedures for both types of host development systems are given below.

3.2.1 Solaris Host Loading Procedure

The instructions below assume that the login session will be using the Common Desktop Environment (CDE) and that the Volume Management daemon (**vold**) is running.

To install the SL240 software on your system:

- Log on to the system as root (superuser).
- Place CD-ROM in the drive. After a few seconds **vold** will automatically open the CD-ROM window.
- Open the File Manager by clicking on the File Drawer icon in the Front Panel.
- Double click on the *usr* folder in the File Manager window to view the *usr* directory.
- Drag the **sl240.tar** file icon from the CD-ROM window to the *usr* window and drop to copy the file from the CD-ROM to */usr*.
- Open a terminal window by right clicking on the background and selecting "Programs" then left clicking on "Terminal".
- Change directory in the terminal window to */usr*:

```
cd /usr
```

- Extract the tar file from the terminal window:

```
tar xvf sl240.tar
```

These procedures will place the contents of the CD-ROM into the */usr/sl240* directory. Once the software is fully loaded from the installation media to the host system, the software installation is complete.

3.2.2 Windows Host Loading Procedure

To install the SL240 software on your system:

1. Place the CD-ROM into the drive of the host system
2. Using Windows Explorer, view the contents of the SL240 CD-ROM and browse to the root of the CD. Next extract the contents of the file sl240.zip to the Tornado 2.2 project directory on the host (default is C:\tornado 2.2\target\proj).

Once the software is fully loaded from the installation media to the host system, the software load is complete.

3.3 Verify the Directory Structure

The SL240 software files are placed into several subdirectories. This software distribution has the directory structure shown in Figure 3-2 (assuming the default directory is */usr/sl240*):

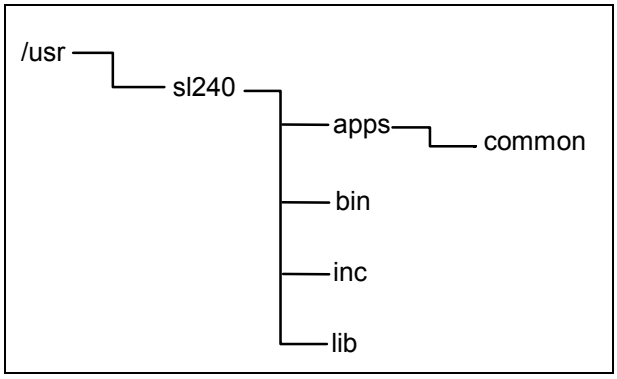

Figure 3-2 Example of Directory Structure

- The *apps* directory contains utility applications as described in Chapter 6 of the *FibreXtreme SL100/SL240 API Guide*. The *apps* directory also contains the *common* directory containing common source code used by the sample applications.
- The *bin* directory contains the device drivers. In the PowerPC distribution there are several subdirectories to hold these files for each target platform.
- The *inc* directory contains the header files for SL240 API library routines.
- The *lib* directory contains the source code to resolve calls to the SL240 API routines.

3.4 Load and Install the SL240 Device Driver

3.4.1 Loading the SL240 Driver Module on a Windows Host

To load the SL240 driver module from within Tornado 2.2 perform the following steps:

1. Login to the Windows host and load the Tornado 2.2 IDE
2. Load the SL240 driver workspace by selecting file->open workspace and then select browse. Locate the file sl240.wsp and select OK.
3. With the SL240 Workspace loaded, locate the device driver project and right click on the project and select download as shown in Figure 3-3.

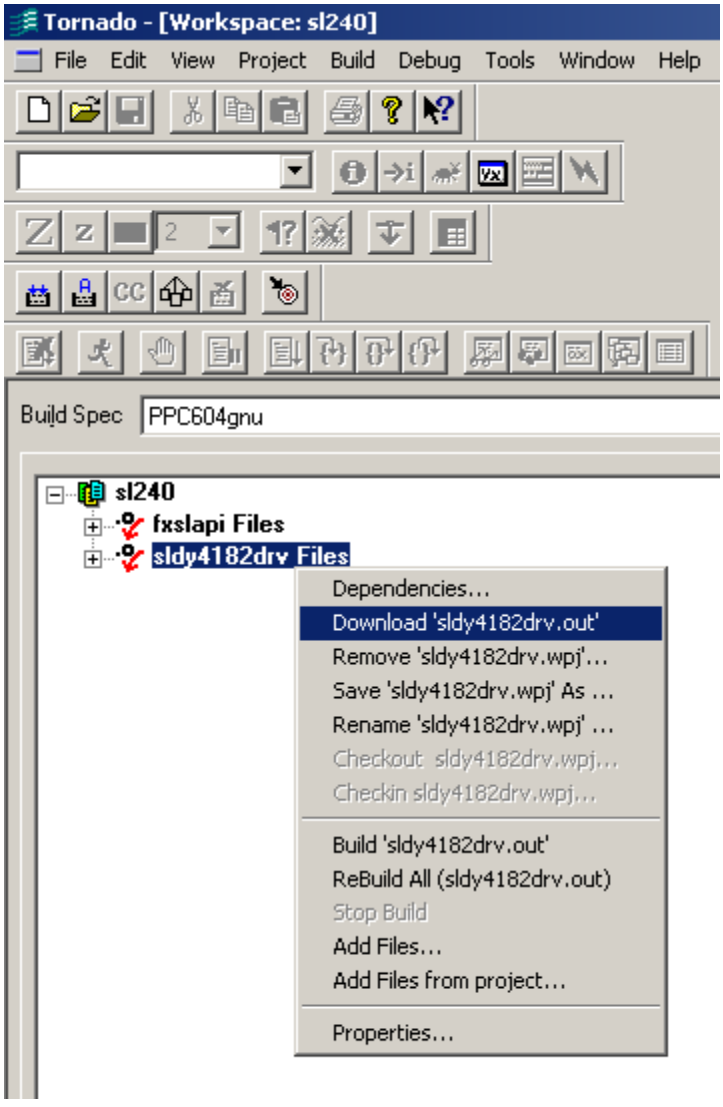

Figure 3-3 Tornado 2.2 Workspace Window

3.4.2 Loading the SL240 Driver Module On A Sun Host Running Solaris

1. Download the VxWorks image from the Sun host.
2. Using the VxWorks Shell, Change the directory containing the SL240 driver module (example:/usr/sl240/bin/dy4182)
3. Load the driver using the VxWorks loader, for example:

```
ld<sldy4182drv.o
```

3.4.3 Configure the SL240 Device Driver

To complete the SL240 device driver installation, *fxslInstall* must be called. This is the calling syntax and description of the parameters for this function.

```
int fxslInstall (fx_ulong    unit,
                 fx_ulong   *base_addr,
                 fx_ulong    irq_line,
                 fx_ulong    exchanges,
                 fx_ulong    use_flow_control,
                 fx_ulong    convert_dsync,
                 fx_ulong    halt_on_link_errors,
                 fx_ulong    allow_qing_on_lerror,
                 fx_ulong    useCRC,
                 fx_ulong    loopConfig)
```

If a parameter is not specified, the default is used. The parameters are:

unit.....Number of the unit to initialize. Valid numbers are between 0 and 15.

base_addr.....Address to place the card's memory.

irq_line.....Interrupt line for the card to use.

exchanges.....The number of exchanges to initialize with. The default is 4.

use_flow_control.....Set to '1' to use flow control.
Set to '0' to not use flow control.
The default is flow control off.

convert_dsync.....Set to '1' to convert sync with/dvalid to sync.
Set to '0' to ignore sync characters.
The default is to ignore sync characters.

halt_on_link_errors.....Set to '1' to halt receive DMA on a link error.
Set to '0' to continue normally.
The default is to continue normally.

allow_qing_on_lerror.....Set to '1' to allow queuing of new requests while the link is down.
Set to '0' to prevent queuing of new requests while the link is down.

useCRC.....Set to '1' to enable CRC generation/checking.
The default is 0.

loopConfig.....Set to any in the range 0-4, 10, and 11 to specify the loop configuration mode. Please refer to the *FibreXtreme SL100/SL240 API Guide* for a detailed description of each of these modes.

UNIT NUMBER

Unit numbers are assigned in two ways. The first is used on targets that do not support PCI plug-and-play. This includes the Motorola 2X00 and VMetro Midas 2x0 series targets. On such systems, the unit numbers start at 0 and increase to the number of available slots. The unit number is fixed according to the slot.

The second method of assigning unit numbers is used on plug-and-play systems. On such systems, the unit numbers are assigned in the order the cards are found.

NOTE: *fxslInstall* must be called for each unit in the system. Valid unit numbers are in the range from 0 to 15.

fxslInstall EXAMPLE

This is an example of using the *fxslInstall* command for the MVME2700 Target:

```
fxslInstall 0, 0xfd000000, 0x19, 8, 1, 0, 1, 1, 1, 0
```

The following shows the SL240 device driver was properly installed (since the return value is 0):

```
value = 0 = 0x0
```

The SL240 device driver has been loaded and should be operational.

NOTE: Refer to the **readme** file in the root of the CD-ROM for a platform-specific *fxslInstall* command example.

3.5 Driver Start Errors

There are several reasons the SL240 device driver may fail to start correctly. An error code will be displayed if the driver fails to start. In the example below, 0xFFFFFFFF is returned because no card is found in slot 0:

```
value = -1 = 0xffffffff = rxsem + 0xff01cfaf
```

Several error codes that may be displayed when starting the SL240 device driver and possible reasons why are as follows:

Table 3-1 Error Codes

Error Code	Explanation
0xFFFFFFFF	No SL240 card
0xFFFFFFFE	Cannot map memory
0xFFFFFFFD	Insufficient Memory
0xFFFFFFFC	IosDrvInstall Failed
0xFFFFFFFB	IosDevAdd Failed
0xFFFFFFFA	Interrupt could not be connected
0xFFFFFFF9	Kernel thread could not be created
0xFFFFFFF8	Invalid slot
0xFFFFFFF7	Cache initialization failed
0xFFFFFFF6	CacheEnable failed for instruction cache
0xFFFFFFF5	CacheEnable failed for data cache
0xFFFFFFF4	Invalid bus address

3.6 Load Application Interface

3.6.1 Loading SL240 Applications on a Windows Host

To load an SL240 application on a Windows host running Tornado 2.2, do the following:

1. Load the Tornado 2.2 IDE and then open the SL240 applications workspace by selecting file->open workspace, select browse and locate the file sl240apps.wsp then click OK.
2. Select the project called fxslapi, right click on the project, and then select download fxslapi.out.
3. With the fxslapi.out library loaded, download the required applications.

3.6.2 Loading SL240 Applications on a Sun Host

To load an SL240 application on a Sun host do the following:

1. Connect to the Sun host using VxWorks Shell.
2. Change to the host directory containing the FibreXtreme API Library (fxslapi.o).
3. Load the library and required applications using the VxWorks Loader:

```
ld<fxslapi.o
ld<sl_mon..o
```

NOTE: The FibreXtreme API Library, fxslapi, must be loaded prior to using the FibreXtreme applications. For more details refer to the SL100/SL240 API Guide.

This page intentionally left blank

APPENDIX A

SUPPORTED SBCS

TABLE OF CONTENTS

TABLES

A.1 PowerPC Targets

A.1.1 Overview

This section contains target-specific details regarding the support of the SL240 device driver on PowerPC SBCs with an SL240 PMC adapter.

The tested configuration(s) are shown in the following tables.

Table A-1 Tested Configurations for VxWorks 5.31

Target	CPUs	SL240	BSP Revision	Compiler
MVME2306	604	PMC	1.1/4	GNU
MVME2603	603	PMC	1.1/4	GNU
MVME2700	750	PMC	1.1/5	GNU
MVME2431	750	PMC	1.2/0	GNU

Table A-2 Tested Configurations for VxWorks 5.4

Target	CPUs	SL240	BSP Revision	Compiler
MVME2306	604	PMC	1.2/0	GNU
MVME2603	603	PMC	1.2/0	GNU
MVME2700	750	PMC	1.2/0	GNU
MVME2431	750	PMC	1.2/0	GNU
MIDAS3220SR	604	PMC	1.2/0.1	GNU

Table A-3 Tested Configurations for Tornado 2.2/VxWorks 5.5

Target	CPUs	SL240	BSP Revision	Compiler
DY4 182	7457	PMC	1.2/1.4	GNU
MVME 2400	750	PMC	1.2/2	GNU
MVME 5110	7410	PMC	1.2/2	GNU
Synergy VYFD	7457	PMC	2.2/2C	GNU
Synergy VYM2	7410	PMC	2.2/2C	GNU
Synergy VYMD	7410	PMC	2.2/2C	GNU

A.1.2 Map Hardware to the Kernel

The BSPs for the MVME2300, MVME2600, MVME2400, and MVME2700 SBC must be modified to work with the SL240 adapters. This section describes the changes required for the MVME2600 BSP. The MVME2300, MVME2400, and MVME2700 should be similar.

The BSPs for the MVME2600 and MIDAS3220 do not include mechanisms for automatically detecting the requirements of devices in the PMC slot and configuring them. This means that enough PCI memory space must already be set aside by the BSP

for the SL240 hardware. The **mv2600.h** and **sysLib.c** files control the configuration of the BSP for the MVME2600.

The driver for MIDAS3220 sets up the SL240 adapter explicitly. For more information see the *VMetro User's Manual PowerMIDAS -S Series PMC I/O Subsystem for VMEbus and RACE++,* Revision 3.0.

mv2600.h
Definitions for the beginning and the size of PCI memory space are included in this file.

```
#define CPU_PCI_MEM_ADRS     0xfd000000  /* base of PCI mem
space */
#define CPU_PCI_MEM_SIZE     0x01000000  /* 16 meg */
```

syslib.c
The example below is from the section of this file that shows the addresses of the PCI memory space from the CPU's perspective.

```
{
  (void *) CPU_PCI_MEM_ADRS,
  (void *) CPU_PCI_MEM_ADRS,
  CPU_PCI_MEM_SIZE,                          /* 16 meg */
  VM_STATE_MASK_VALID | VM_STATE_MASK_WRITABLE |
   VM_STATE_MASK_CACHEABLE,
  VM_STATE_VALID   | VM_STATE_WRITABLE   |
   VM_STATE_CACHEABLE_NOT
  },
```

A.1.3 Default Hardware Addresses

MOTOROLA Target

The default CPU address for the SL240 hardware is 0xFD000000. The BSP version 1.1/4 uses an extended VME addressing scheme, and it also gives the CPU and the PCI bus similar perspectives. This means that a CPU address of 0xFD000000 translates directly to a PCI physical address of 0xFD000000.

MIDAS3220

The SL240 driver for the MIDAS 3220 SBC uses a portion of the 256 MB area to map the SL240 adapter. Slot 0 starts at address 0x00400000 and slot 1 at 0xB0000000.

A.1.4 Rebuilding the VxWorks Board Support Package (BSP)

Rebuild the VxWorks kernel by running the **makefile** in the same directory as **sysLib.c** to include the changes. After successfully rebuilding the VxWorks kernel, reboot the TARGET processor board with the new kernel.

Rebuilding the MIDSA3220 BSP should not be necessary.

A.2 i960 Targets

A.2.1 Overview

This section contains target-specific details regarding the support of the SL240 device driver on an i960 SBC with an SL240 PCI/PMC adapter.

The tested configuration(s) are shown in the following tables.

Table A-4 Tested Configurations for the Cyclone SB923 for VxWorks 5.3.1

Target	CPU	SL240	BSP Revision	Compiler
Cyclone SB923	i960RN	PCI	1.2/0	GNU
MIDAS220MR	i960Rx	PMC	1.1/0	GNU

Table A-5 Tested Configurations for the Cyclone SB923 for VxWorks 5.4

Target	CPU	SL240	BSP Revision	Compiler
Cyclone SB923	i960RN	PCI	1.2/0	GNU
MIDAS220MR	i960Rx	PMC	1.2/0	GNU

A.2.2 Map Hardware to the Kernel

CYCLONE SB923

The BSP for the Cyclone SB923 includes mechanisms for automatically detecting the requirements of devices in the PMC slots and for configuring them. This means that enough PCI memory space will automatically be set aside by the BSP for the SL240 hardware. The ***config/sb923/config.h*** file controls the configuration of the BSP.

MIDAS220 MR

The BSP for the MIDAS2 does not include mechanisms for automatically detecting the requirements of devices in the PMC slots and for configuring these devices. This driver will set up the SL240 adapter explicitly.

A.2.3 Default Hardware Addresses

There are no default hardware addresses that will be used for the SL240 adapter, since the Cyclone SB923 BSP dynamically determines where to place the board in PCI memory space.

The Cyclone SB923 BSP files do provide some information about where the BSP will map the device, as shown in Table A-6.

Table A-6 Cyclone SB923 BSP Memory Map from the CPU's Perspective

Component	Memory Address
PCI Memory Space	0x20000000 - 0x2FFFFFFF
PCI I/O Space	0x30000000 - 0x3FFFFFFF

This memory map shows that the SL240 board should end up somewhere between 0x20000000 and 0x2FFFFFFF from the perspective of the CPU, which basically means that the CPU address 0x20000000 corresponds to the PCI memory space address 0x00000000.

MIDAS220

The SL240 driver for the MIDAS i960 SBC uses a portion of the 256 MB area starting at 0x70000000 to map the SL240 adapter in slot 0 of the SBC.

A.2.4 Rebuilding the VxWorks Board Support Package (BSP)

Rebuilding the BSP should not be necessary for either the SB923 or the MIDAS220.

A.3 x86 Platforms

A.3.1 Overview

This section contains target-specific details regarding the support of the SL240 device driver on x86 series SBCs.

NOTE: This software does not support PCI-to-PCI bridges. If you have a motherboard with more than one PCI bus connected by a PCI-to-PCI bridge, place the card on the first bus.

A.3.2 Map Hardware to the Kernel

The BSPs for the x86 SBC must be modified to work with the SL240 adapters. This section describes the changes required for the BSP.

The BSP for the x86 does not include mechanisms for automatically detecting the requirements of devices in the PCI/PMC slot and configuring them. This means that enough PCI memory space must already be set aside by the BSP for the SL240 hardware. The **sysLib.c** file should be modified to control the configuration of the BSP.

syslib.c

Add definitions for the size of PCI memory space and base address for the SL240 cards.

```
#define SL240_MEM_ADRS     0x40000000    /* base PCI
address for SL240*/
#define SL240_MEM_SIZE     0x00200000  /* 0x200000 for each
card */
```

The example below is one entry of the sysPhysMemDesc array in this file that shows the address of the SL240 PCI memory from the CPU's perspective. Remove one DUMMY_MMU_ENTRY for each entry used for SL240.

```
{
  (void *) SL240_MEM_ADRS,
  (void *) SL240_MEM_ADRS,
  SL240_MEM_SIZE,
  VM_STATE_MASK_VALID | VM_STATE_MASK_WRITABLE |
   VM_STATE_MASK_CACHEABLE,
  VM_STATE_VALID   | VM_STATE_WRITABLE   |
   VM_STATE_CACHEABLE_NOT
  },
```

A.3.3 Rebuilding the VxWorks Board Support Package (BSP)

Rebuild the VxWorks kernel by running **make** in the same directory as **sysLib.c** to include the changes. After successfully rebuilding the VxWorks kernel, reboot the TARGET machine using the new kernel.

This page intentionally left blank

APPENDIX B

REHOSTING CONSIDERATIONS

TABLE OF CONTENTS

B.1 Overview

NOTE: Retargeting software to a Single Board Computer can be difficult and should not be attempted without experience with VxWorks and writing device drivers. Customer Support will supply a limited amount of technical assistance to customers attempting to do their own retarget. Curtiss-Wright's Application Engineering can retarget software to a different target SBC for a fee. For information on having Curtiss-Wright software targeted to a specific SBC, please contact Curtiss-Wright Controls, Inc. Embedded Computing, Data Communications Center, Sales department.

There are enough differences in the architecture of single board computers (SBCs) and VxWorks software support of the SBCs to warrant some discussion on considerations necessary when rehosting the driver to another target SBC.

To assist in the process of porting to a new SBC, all host specific functions have been placed in the same file. To port to a new SBC, the user must:

1. Modify these SBC specific functions to work on the new system.
2. Recompile the new source code file.
3. Link the new object file with the driver's other object files.
4. Load and install the driver on the new SBC.

B.2 Modify SBC Specific Functions

All SBC specific files are included in a header and/or c source file that is named appropriately for the supported computer. For example, the **mvme2x00.c** and **mvme2x00.h** files contain the functions for the Motorola VME2X00 SBC series.

B.3 Getting Started

Included with the SL100/SL240 software is an example SBC specific source and header file. Instead of starting from scratch from the descriptions below, Curtiss-Wright's suggests you take an example set of files and modify them to fit your needs.

The SBC example files included depend on the platform ordered. For this reason, they are not listed here. To learn more about the supported SBCs, review the supported platforms appendix in this manual.

B.4 Required Global Variable Definitions

This section describes system specific global variables that are required by the driver.

DATA_SWAP

DATA_SWAP is a global variable that is set to 1 if DMA transactions should be BYTE swapped and set to 0 if DMA transactions are not to be swapped.

To not swap DMA transactions, use the following definition:

```
fx_ulong  DATA_SWAP  = 0;
```

To swap DMA transactions, use the following definition:

```
fx_ulong  DATA_SWAP  = 1;
```

B.5 Required Functions

This section describes SBC specific functions that are required by the driver.

B.5.1 SysInit

Function prototype:

fx_ulong SysInit()

Description:

The function S*ysInit* is called once before the first SL240 driver is installed. Any system setup that needs to run once and only once on a system should be placed in it.

Input:

None.

Output:

None.

Returns:

Return *FXSL_SUCCESS* (0) if function succeeds. Return any system specific SL240 error code from Table 3-1 on page 3-5 if function fails.

Pre-conditions:

None.

Post-conditions:

System specific.

B.5.2 sysPciBoardConfig

Function prototype:

*fx_ulong sysPciBoardConfig (fxsl_ctl *xc)*

Description:

The function *sysPciBoardConfig* is called once for every SL240 driver that is installed.

Input:

xc.....pointer to the board control structure.

Output:

xc.....pointer to the board control structure.

Returns:

Return *FXSL_SUCCESS* (0) if function succeeds. Return any system specific SL240 error code from Table 3-1 on page 3-5 if function fails.

Pre-conditions:

xc.....must point to a valid control structure.
sysPciBoardConfig expects the following fields to be set before the function is called:
xc->unit.....slot number to configure.
xc->basePtr.....address to map registers.
xc->irql.....interrupt level to use.

Post-conditions:

If the system has not already done so, *sysPciBoardConfig* must find the board and map the board into PCI space.

When *sysPciBoardConfig* returns, the following fields in **xc** must be set:

xc->Bus.....card's PCI bus number.
xc->Dev .....card's PCI device number.
xc->Func .....card's PCI function number.
xc->boardType .....FXSL_TX and FXSL_RX denote the card type.
xc->pRtRegs .....pointer to runtime registers.
xc->pFifo.....pointer to the FIFO.

NOTE: The function *sysPciBoardConfig* must map two regions of PCI space into memory. The run-time registers, pointed to by **xc->pRtRegs**, require at least 256 bytes. The on-board fifos, pointed to by **xc->pFifo**, require at least one MB.

B.5.3 sysSetIntr

Function prototype:

*fx_ulong sysSetIntr (fxsl_ctl *xc)*

Description:

The function *sysSetIntr* performs any system specific initialization required to receive an interrupt.

Input:

xc.....pointer to the board control structure.

Output:

None.

Returns:

Return *FXSL_SUCCESS* (0) if function succeeds. Return any system specific SL240 error code from Table 3-1 on page 3-5 if function fails.

Pre-conditions:

xc.....must point to a valid control structure.
SysSetIntr expects the following fields to be set before the function is called:

xc->unit.....board slot number.

xc->irql.....irq level to use.

Post-conditions:

The interrupt for the board **xc->unit** must be connected using interrupt line

xc->irql.....the connected interrupt handler must be a wrapper that calls *kcoreHandleInterrupt* to determine if the interrupt is for this board.

An example of an acceptable interrupt service routine wrapper is contained below:

```
int sysIsrWrapper(fxsl_ctl *xc)
{
   if (kcoreHandleInterrupt(xc))
   {
      kcoreCompleteDma(xc);
      return FXSL_SUCCESS;
   }
   else
   {
      return FXSL_DRIVER_ERROR;
   }
}
```

B.5.4 sysCacheFlush

Function prototype:

*void sysCacheFlush (fxsl_ctl *xc, void *ptr, fx_ulong size)*

Description:

The function *sysCacheFlush* must flush the cache of the memory range specified by **ptr** and **size**.

Input:

xc.....pointer to board control structure.
ptr.....pointer to the region to flush.
size.....number of bytes to flush.

Output:

None.

Returns:

Nothing.

Pre-conditions:

xc .....must point to a valid control structure.
ptr .....must point to a valid physical memory address.
size .....must be the number of bytes to flush.

Post-conditions:

The specified region has been flushed from all caches.

B.5.5 sysCacheInvalidate

Function prototype:

*void sysCacheInvalidate (fxsl_ctl *xc, void *ptr, fx_ulong size)*

Description:

The function *sysCacheInvalidate* must invalidate any cached values of the memory range specified by **ptr** and **size**.

Input:

xc.....pointer to board control structure.
ptr .....pointer to the region to invalidate.
size.....number of bytes to invalidate.

Output:

None.

Returns:

Nothing.

Pre-conditions:

xc .....must point to a valid control structure.
ptr .....must point to a valid physical memory address.
size .....must be the number of bytes to invalidate.

Post-conditions:

The specified region has been invalidated in all caches.

B.5.6 sysVirtToPhysAddr

Function prototype:

*fx_ulong sysVirtToPhysAddr (fxsl_ctl *xc, void *ptr)*

Description:

The function *sysVirtToPhysAddr* converts a virtual address to a physical address.

Input:

xc.....pointer to board control structure.
ptr.....pointer of the address to convert.

Output:

None.

Returns:

32 bit physical address of the virtual pointer.

Pre-conditions:

xc .....must point to a valid control structure.
ptr .....must point to a valid virtual memory address.

Post-conditions:

Returns the physical address of **ptr**.
Unless the system is using virtual memory, this function will return **ptr**.

B.5.7 sysPhysToBusAddr

Function prototype:

*fx_ulong sysPhysToBusAddr (fxsl_ctl *xc, void *ptr)*

Description:

The function *sysPhysToBusAddr* converts a physical address in the processor's system memory to a PCI bus address.

Input:

xc .....pointer to board control structure.
ptr .....pointer of the address to convert.

Output:

None.

Returns:

32 bit PCI bus address of the physical pointer.

Pre-conditions:

xc .....must point to a valid control structure.
ptr .....must point to a valid physical system memory address.

Post-conditions:

Returns the PCI bus address of **ptr**.

B.5.8 sysAlignAddrOn16Bytes

Function prototype:

*void *sysAlignAddrOn16Bytes (void *ptr)*

Description:

The function *sysAlignAddrOn16Bytes* is used to get a 16-byte-aligned pointer from within an allocated buffer.

Input:

ptr ..... pointer to a buffer to align.

Output:

None.

Returns:

16 byte aligned address from within the buffer pointed to by **ptr**.

Pre-conditions:

ptr ..... must point to a valid physical system memory address. The buffer pointed to by ptr must be 16 bytes longer than the required aligned buffer.

Post-conditions:

Returns a pointer to the buffer that is 16-byte aligned.

The returned pointer must reference an address that is between 0 and 15 bytes lower than the address referenced by **ptr**.

B.5.9 syskMalloc

Function prototype:

*void *syskMalloc(fx_ulong numBytes)*

Description:

The function *syskMalloc* is used to allocate memory at the kernel level.

Input:

numBytes ..... number of bytes to allocate.

Output:

None.

Returns:

A pointer to a buffer of **numBytes** bytes or NULL if the memory is not available.

Pre-conditions:

None.

Post-conditions:

Returns a pointer to a buffer in kernel space of size **numBytes** or NULL if the memory if not available.

B.5.10 syskFree

Function prototype:

*void syskFree(void *buf, fx_ulong numBytes)*

Description:

The function *syskFree* is used to free memory allocated by *syskMalloc*.

Input:

buf ..... pointer to the buffer to return.
numBytes ..... number of bytes in the buffer.

Output:

None.

Returns:

Nothing.

Pre-conditions:

buf ..... must be a valid pointer to a buffer allocated by a call to *syskMalloc*.

Post-conditions:

The buffer pointed to by **buf** has been returned to pool of available dynamic memory.

Notes:

numBytes is not used by the VxWorks implementation.

B.6 Recompile and Link the New File with the Driver's Other Object Files

To complete the integration of changes into the driver, compile the new file and link the new object file with the provided object files. This can be done using the provided makefile. Depending on your systems, the makefile may require some changes.

NOTE: The main object files, that must be linked with the modified object file, are located in subdirectories under */usr/sl240/driver*.

B.7 Load and Install the Driver on the New SBC.

To load and install the driver, following the instructions in Chapter 3.

This page intentionally left blank.