Release Notes
CCURDPRC (WC-DPRC)

Driver	ccurdprc (WC-DPRC)	
Platform	RedHawk Linux® *(CentOS/Rocky/RHEL & Ubuntu)*, Native Ubuntu® and Native Red Hat Enterprise Linux®[1]	
Vendor	Concurrent Real-Time	
Hardware	Digital Programmable Resister Card (DPRC)	
Author	Darius Dubash	
Date	November 10th, 2025	Rev 2025.2

concurrent REAL-TIME

[1] All trademarks are the property of their respective owners

This page intentionally left blank

Table of Contents

This page intentionally left blank

1. Introduction

This document assists the user in installing the CCUR-PCIe-DPRC Linux ***ccurdprc*** driver and related software on the RedHawk OS, Native Ubuntu and Native Red Hat for use with the CCUR-PCIe Digital Programmable Resister Card (***DPRC***). The directions in this document supersede all others – they are specific to installing the software on Concurrent Real-Time's RedHawk and Native Ubuntu and Native Red Hat systems. Other information provided as part of this release, when it may contradict these directions, should be ignored and these directions should prevail.

Current versions of Native Operating Systems that are supported are:

1) Ubuntu 22.04, kernel 6.5 or 6.8, gcc-11 & gcc-12
2) Ubuntu 24.04, kernel 6.14, gcc-13
3) Red Hat RHEL 9.4, kernel 5.14 (minor 427.28.1 or less)
4) Red Hat RHEL 9.6, kernel 5.14 (minor 570.39.1 or less)

For additional information on this driver and usage refer to the ***ccurdprc*** man page.

Features and Characteristics of the DPRC are:

- 16-channel Digital Programmable Resistance
- 45 to 1M Ohm Range (In 5 Ohm Steps)
- 10 Ohm Low Scale Selection
- +/-14V @ 10 Milliamp
- Open, Ground and V+ Fault Insertion
- Galvanic Isolation
- Overvoltage Protection
- Overcurrent Protection
- Analog Devices AD5293 Digital Potentiometers
- Industry Standard SCSI 68-pin Connector for I/O
- PCI Express x1 Revision 1.0a
- NIST Traceable Calibration Standard (Optional)

2. Requirements

- CCUR-DPRC PCIe board physically installed in the system.
- This driver supports various versions of RedHawk and a selected set of Native Ubuntu and Native Red Hat. Actual supported versions depend on the driver being installed.

3. Documentation

- PCIe Digital Programmable Resister Card (DPRC) Software Interface by Concurrent Real-Time.
- PCIe Digital Programmable Resister Card (DPRC) Design Specification by Concurrent Real-Time.

4. Running on Native Red Hat

Though this driver and hardware work best on Concurrent Real-Time ***RedHawk*** systems, the driver will also be able to run on some selected versions of ***Red Hat*** with some limitations. Some of these limitations are highlighted below. The rest of the document is applicable to all systems.

When compiling the driver, you may get the following message that can be ignored:

Skipping BTF generation for /usr/local/CCRT/drivers/ccurdprc/driver/ccurdprc.ko due to unavailability of vmlinux

4.1. Support to build 3rd party modules

If your system isn't setup to build 3rd party modules, you will need to install some of the following packages if they havn't already been installed before being able to compile the driver. Installation process of these modules may differ from system to system. Refer to the particular system for installation of the modules.

```
# yum install ncurses-devel        (to run curses)
# yum install gnuplot              (to run plots for various tests)
# yum install <any other package you want to install>
```

4.2. Support for MSI interrupts

- The driver can operate with either MSI or wired interrupts. This is a configuration option that can be selected by editing the *ccurdprc_nomsi* parameter located in the …/*driver/ccurdprc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccurdprc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccurdprc_nomsi=1 disable MSI support

  Red Hat systems do not have kernel level hooks like CCRT RedHawk systems to enable MSI on a per board basis for cards using a PLX chip for generating interrupts. This is specially true for the later X11SPA-TF SuperMicro Mother boards and onwards. In this case, if the user wishes to use MSI instead of wired interrupts, they can enable them in various ways as outlined below.

- If MSI interrupts are not being generated and the user wishes to continue using MSI interrupts instead of wired interrupts, they can try to resolve the problem by implementing one the following:

  - Reload the kernel with the grub option "iommu=pt"
  - Reload the kernel with the grub option "iommu=off"
  - Disable IOMMU in the BIOS
  - Reload the kernel with the grub option "intremap=nosid"
  - Reload the kernel with the grub option "intremap=off"
  - Disable VT-d in the BIOS
  - Disable VT-d MSI Interrupt Remapping in the BIOS
  - Disable 4G Decoding in the BIOS

- To add/remove/display the ***intremap*** command to grub, issue the following commands:

  - # grubby --update-kernel=ALL --args=iommu=pt *(add the parameter)*
  - # grubby --update-kernel=ALL --args=iommu=off *(add the parameter)*
  - # grubby --update-kernel=ALL --args=intremap=nosid *(add the parameter)*
  - # grubby --update-kernel=ALL --remove-args=intremap=nosid *(remove the parameter)*
  - # grubby --info=ALL *(display parameters)*
  - # reboot
  - After system reboots, issue the command "***cat /proc/cmdline***" to see if the added entry is present.

4.3. BIOS and Kernel Level Tuning

BIOS tuning for real-time is specific to the mother board where the Red Hat kernel is running. The various BIOS settings need to be studied and changed accordingly to make sure that it is running at optimal performance with minimal interference from other processes.

Some Red Hat kernel level tuning can be performed to see if they are helpful in getting a more real-time performance.

Disable features that allows SCHED_OTHER tasks to use up to 5% or RT CPUs.

```
sysctl kernel.sched_rt_runtime_us=-1
echo -1 > /proc/sys/kernel/sched_rt_runtime_us
```

Disable timer migration:

```
sysctl kernel.timer_migration=0
echo 0 > /proc/sys/kernel/timer_migration
```

Add follwing parameters to ***/etc/default/grub*** line and running ***update-grub*** and ***reboot***.

```
GRUB_CMDLINE_LINUX="skew_tick=1 rcu_nocb_poll rcu_nocbs=1-95 nohz=on nohz_full=1-95
kthread_cpus=0 irqaffinity=0 isolcpus=managed_irq,domain,1-95 intel_pstate=disable
nosoftlockup tsc=nowatchdog"
```

Isolate CPUs e.g *(this command has been officially marked deprecated)*

```
isolcpus=1-8,26-30 rcu_nocbs=1-8,26-30 nohz_full=1-8,26-30 rcu_nocb_poll=1-8,26-30
```

5. Running on Native Ubuntu

Though this driver and hardware work best on Concurrent Real-Time ***RedHawk*** systems, the driver will also be able to run on some selected versions of ***Ubuntu*** with some limitations. Some of these limitations are highlighted below. The rest of the document is applicable to all systems.

When compiling the driver, you may get the following message that can be ignored:

Skipping BTF generation for /usr/local/CCRT/drivers/ccurdprc/driver/ccurdprc.ko due to unavailability of vmlinux

5.1. Support to build 3rd party modules

If your system isn't setup to build 3rd party modules, you will need to install some of the following packages if they havn't already been installed before being able to compile the driver. Installation process of these modules may differ from system to system. Refer to the particular system for installation of the modules.

```
# apt install build-essential
# apt install libssl-dev
# apt install nfs-common          (to mount nfs file systems)
# apt install libncurses-dev      (to run curses)
# apt install gnuplot             (to run plots for various tests)
# apt install chrony              (for more accurate clock time)
# apt install                     <any other package you want to install>
```

5.2. Support for MSI interrupts

- The driver can operate with either MSI or wired interrupts. This is a configuration option that can be selected by editing the *ccurdprc_nomsi* parameter located in the …/*driver/ccurdprc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccurdprc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccurdprc_nomsi=1 disable MSI support

  Red Hat systems do not have kernel level hooks like CCRT RedHawk systems to enable MSI on a per board basis for cards using a PLX chip for generating interrupts. This is specially true for the later X11SPA-TF SuperMicro Mother boards and onwards. In this case, if the user wishes to use MSI instead of wired interrupts, they can enable them in various ways as outlined below.

- If MSI interrupts are not being generated and the user wishes to continue using MSI interrupts instead of wired interrupts, they can try to resolve the problem by implementing one the following:

- Reload the kernel with the grub option “iommu=pt”
- Reload the kernel with the grub option “iommu=off”
- Disable IOMMU in the BIOS
- Reload the kernel with the grub option “intremap=nosid”
- Reload the kernel with the grub option “intremap=off”
- Disable VT-d in the BIOS
- Disable VT-d MSI Interrupt Remapping in the BIOS
- Disable 4G Decoding in the BIOS

- To add/remove/display the ***intremap*** command to grub, issue the following commands:

  - Edit ***/etc/default/grub*** and add "iommu=pt" or “iommu=off” and/or add “intremap=nosid” to “GRUB_CMDLINE_LINUX=” entry
  - # update-grub
  - # reboot
  - After system reboots, issue the command “***cat /proc/cmdline***” to see if the added entry is present.

5.3. Compiling the driver with installed gcc

Depending on the Ubuntu kernel version supported, you will need to make sure that the driver is compiled with the same gcc as the kernel.

Currently, for Ubuntu release 22.04, the kernel 5.15 uses gcc-11 while kernel 6.4 or 6.8 uses gcc-12 and kernel 6.14 is compiled with gcc-13.

If gcc-12 is not installed, you can do the following:

```
# apt install gcc-12
```

Then create alternate entries for each available version:

```
# sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 11
# sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-12 12

# sudo update-alternatives --install /usr/bin/x86_64-linux-gnu-gcc x86_64-linux-gnu-gcc /usr/bin/x86_64-linux-gnu-gcc-11 11

# sudo update-alternatives --install /usr/bin/x86_64-linux-gnu-gcc x86_64-linux-gnu-gcc /usr/bin/x86_64-linux-gnu-gcc-12 12
```

You can select the appropriate gcc with the following commands:

```
# sudo update-alternatives --config gcc
# sudo update-alternatvies --config x86_64-linux-gnu-gcc
```

All of this will ensure you have the compiler versions that match what the kernel was compiled with.

5.4. BIOS and Kernel Level Tuning

BIOS tuning for real-time is specific to the mother board where the Red Hat kernel is running. The various BIOS settings need to be studied and changed accordingly to make sure that it is running at optimal performance with minimal interference from other processes.

Some Red Hat kernel level tuning can be performed to see if they are helpful in getting a more real-time performance.

Disable features that allows SCHED_OTHER tasks to use up to 5% or RT CPUs.

```
sysctl kernel.sched_rt_runtime_us=-1
echo -1 > /proc/sys/kernel/sched_rt_runtime_us
```

Disable timer migration:

sysctl kernel.timer_migration=0
echo 0 > /proc/sys/kernel/timer_migration

Add follwing parameters to ***/etc/default/grub*** line and running ***update-grub*** and ***reboot***.

GRUB_CMDLINE_LINUX="skew_tick=1 rcu_nocb_poll rcu_nocbs=1-95 nohz=on nohz_full=1-95 kthread_cpus=0 irqaffinity=0 isolcpus=managed_irq,domain,1-95 intel_pstate=disable nosoftlockup tsc=nowatchdog"

Isolate CPUs e.g *(this command has been officially marked deprecated)*
isolcpus=1-8,26-30 rcu_nocbs=1-8,26-30 nohz_full=1-8,26-30 rcu_nocb_poll=1-8,26-30

6. Installation and Removal

6.1. Hardware Installation

The CCUR-DPRC card is a Gen 1 PCI Express product and is compatible with any PCI Express slot. The board must be installed in the system before attempting to use the driver.

The ***ccurdprc*** driver is designed to support IRQ sharing. If this device's IRQ is being shared by another device then this driver's performance could be compromised. Hence, as far as possible, move this board into a PCI slot who's IRQ is not being shared with other devices. The default driver configuration uses MSI interrupts. If the kernel supports MSI interrupts, then sharing of interrupts will not occur, in which case the board placement will not be an issue.

Caution: when installing the card insure the computer is powered off and the machine's power cord is disconnected. Please observe electrostatic discharge precautions such as the use of a grounding strap.

An ***'lspci -v'*** or the ***'lsirq'*** command can be used to determine the IRQs of various devices in the system.

```
# lspci -vv -d1542:9310
```

08:04.0 System peripheral: Concurrent Computer Corporation Device 9310 (rev 01)
Subsystem: PLX Technology, Inc. Device 9056
Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV+ VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
Status: Cap+ 66MHz+ UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
Latency: 96, Cache Line Size: 32 bytes
Interrupt: pin A routed to IRQ 55
Region 0: Memory at c4c01000 (32-bit, non-prefetchable) [size=512]
Region 2: Memory at c4c00000 (32-bit, non-prefetchable) [size=2K]
Capabilities: <access denied>

```
# lsirq
```

55 08:04.0 Concurrent Computer Corporation Unknown device (rev 01)

After installing the card, reboot the system and verify the hardware has been recognized by the operating system by executing the following command:

```
# lspci -d 1542:9310
```

For each CCUR-DPRC PCIe board installed, a line similar to one of the following will be printed, depending on the revision of the system's ***/usr/share/hwdata/pci.ids*** file:

08:04.0 System peripheral: Concurrent Computer Corporation Device 9310 (rev 01)

If a line like the above is not displayed by the **lspci** command, the board has not been properly installed in the system. Make sure that the device has been correctly installed prior to attempting to use the software. One similar line should be found for each installed card.

6.2. Software Installation

Concurrent Real-Time™ port of the ***ccurdprc*** software is distributed in RPM format for CentOS and DEB format for Ubuntu OS on a CD-ROM. Source for the API library and kernel loadable driver are not included, however, source for example test programs as well as documentation is provided in PDF format.

The software is installed in the **/usr/local/CCRT/drivers/ccurdprc** directory. This directory will be referred to as the "top-level" directory by this document.

Warning: Before installing the software, for RedHawk kernels, the build environment **must** be set up and match the current OS kernel you are using. If you are running one of the preconfigured kernels supplied by Concurrent Real-Time and have not previously done so, run the following commands while logged in as the ***root*** user before installing the driver software:

```
# cd /lib/modules/`uname -r`/build
# ./ccur-config -c -n
```

If you have built and are running a customized kernel configuration the kernel build environment should already have been set up when that custom kernel was built.

To install the ***ccurdprc*** package, load the CD-ROM installation media and issue the following commands as the **root** user. The system should auto-mount the CD-ROM to a mount point in the **/media** or **/run/media** directory based on the CD-ROM's volume label – in this case **ccurdprc_driver**. The example's ***[user_name]*** may be ***root***, or the logged-in user. Then enter the following commands from a shell window:

```
== as root ==
       --- on RedHawk 6.5 and below ---
# cd /media/ccurdprc_driver
       --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/ccurdprc_driver
      --- or on Ubuntu RedHawk ---
# cd /media/[user_name]/ccurdprc_driver

# rpm -ivh ccurdprc_RedHawk_driver*.rpm  (on a RedHawk CentOS/Rocky based system)
                    --or--
# dpkg -i ccurdprc_RedHawk_driver*.deb   (on a RedHawk Ubuntu based system)
                    --or–
# rpm -ivh ccurdprc_RedHat_driver*.rpm   (on a Native RedHat based system)
                    --or--
# dpkg -i ccurdprc_Ubuntu_driver*.deb    (on a Native Ubuntu based system)

# cd /
# eject
```

On successful installation, the source tree for the ***ccurdprc*** package, including the loadable kernel module, API libraries, and test programs is extracted into the **/usr/local/CCRT/drivers/ccurdprc**

directory by the rpm installation process, which will then compile and install the various software components.

The loadable kernel module is installed in the **/lib/modules/`uname –r`/misc** directory.

Once the package is installed, the driver needs to be loaded with one of the following commands:

```
== as root ==
# cd /usr/local/CCRT/drivers/ccurdprc
# make load
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccurdprc start
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl start ccurdprc
      --- or on Ubuntu RedHawk ---
# /bin/systemctl start ccurdprc
```

Issue the command below to view the boards found by the driver:

```
# cat /proc/ccurdprc

Version            : 23.1.1
Built              : Tue Apr 14 12:33:40 EST 2020
Boards             : 1
  card=0: [08:04.0] bus=8, slot=4, func=0, irq=55, msi=1, ID=680593,
BoardInfo=0x93100102
```

Note: With RedHawk 7.5 you may see a cautionary message similar to the following when the **ccurdprc** driver is loaded on the system console or via *dmesg* command:

CHRDEV "ccurdprc" major number 233 goes below the dynamic allocation range

As documented in the kernel driver **Documentation/devices.txt** file a range of character device numbers from 234 to 254 are officially available for dynamic assignment. Dynamic assignments start at 254 and grow downward. This range is sometimes exceeded as additional kernel drivers are loaded. Note that this was also the case with earlier kernels – the newer 7.5 kernel has added a runtime check to produce this warning message that the lower bound has been exceeded, not reduced the range of numbers officially available for dynamic assignment. If you see this message please verify the assigned number(s) isn't being used by a device installed on your system.

6.3. Software Removal

The ***ccurdprc*** driver is a dynamically loadable driver that can be unloaded, uninstalled and removed. Once removed, the only way to recover the driver is to re-install the ***rpm*** or ***deb*** from the installation CDROM:

If any changes have been made to the driver package installed in ***/usr/local/CCRT/drivers/ccurdprc*** directory, they need to be backed up prior to invoking the removal; otherwise, all changes will be lost.

```
== as root ==
# rpm -e ccurdprc     (driver unloaded, uninstalled, and deleted – on an RPM based system)
  --or--
# dpkg -P ccurdprc    (driver unloaded, uninstalled, and deleted – on an Debian based
                       system)
```

If, for any reason, the user wishes to un-load and uninstall the driver and not remove it, they can perform the following:

```
== as root ==
# cd /usr/local/CCRT/drivers/ccurdprc
# make unload            (unload the driver from the kernel)
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccurdprc stop
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl stop ccurdprc
      --- or on Ubuntu RedHawk ---
# /bin/systemctl stop ccurdprc
```

To uninstall the ***ccurdprc*** driver, do the following after it has been unloaded:

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurdprc
# make uninstall         (uninstall the driver and library)
```

In this way, the user can simply issue the ***'make install'*** and ***'make load'*** in the ***/usr/local/CCRT/drivers/ccurdprc*** directory later to re-install and re-load the driver.

7. Auto-loading the Driver

The ***ccurdprc*** driver is a dynamically loadable driver. Once you install the package or perform the ***'make install'***, appropriate installation files are placed in the /etc/rc.d/rc*.d or /usr/lib/system/systemd directories so that the driver is automatically loaded and unloaded when Linux is booted and shutdown. If, for any reason, you do not wish to automatically load and unload the driver when Linux is booted or shutdown, you will need to manually issue the following command to enable/disable the automatic loading of the driver:

```
=== as root ===
      --- on RedHawk 6.5 and below ---
# /sbin/chkconfig --add ccurdprc        (enable auto-loading of the driver)
# /sbin/chkconfig --del ccurdprc        (disable auto-loading of the driver)
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl enable ccurdprc    (enable auto-loading of the driver)
# /usr/bin/systemctl disable ccurdprc   (disable auto-loading of the driver)
      --- or on Ubuntu RedHawk ---
# /bin/systemctl enable ccurdprc        (enable auto-loading of the driver)
# /bin/systemctl disable ccurdprc       (disable auto-loading of the driver)
```

8. Testing and Usage

Build and run the driver test programs, if you have not already done so:

```
# cd /usr/local/CCRT/drivers/ccurdprc
# make test                          (build the test programs)
```

Several tests have been provided in the ***/usr/local/CCRT/drivers/ccurdprc/test*** directory and can be run to test the driver and board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurdprc
# make test                          (build the test programs)
# ./test/ccurdprc_dump               (dump all board resisters)
# ./test/ccurdprc_rdreg              (display board resisters)
# ./test/ccurdprc_reg                (Display board resisters)
# ./test/ccurdprc_regedit            (Interactive board register editor test)
# ./test/ccurdprc_tst                (Interactive test to test driver and board)
# ./test/ccurdprc_wreg               (edit board resisters)
```

```
# ./test/Eeprom/ccurdprc_eeprom       (Eeprom: Burn Eeprom)

# ./test/Flash/ccurdprc_flash         (Flash: Flash firmware)
# ./test/Flash/ccurdprc_fwreload      (Flash: Firmware reload)

# ./test/lib/ccurdprc_adc_calibrate     (library: display or calibrate)
# ./test/lib/ccurdprc_disp              (library: display, program & test)
# ./test/lib/ccurdprc_fault_protection  (library: display fault protection information)
# ./test/lib/ccurdprc_fault_trip_test   (library: perform fault trip testing)
# ./test/lib/ccurdprc_identify          (library: identify cards in the system)
# ./test/lib/ccurdprc_info              (library: provide information of all boards)
# ./test/lib/ccurdprc_tst_lib           (library: Interactive test for driver & board)
```

9. Re-building the Driver, Library and Tests

If for any reason the user needs to manually rebuild and load an *installed* ***rpm*** or ***deb*** package, they can go to the installed directory and perform the necessary build.

Warning: Before installing the software, for RedHawk kernels, the build environment **must** be set up and match the current OS kernel you are using. If you are running one of the preconfigured kernels supplied by Concurrent Real-Time and have not previously done so, run the following commands while logged in as the root user before installing the driver software:

```
# cd /lib/modules/`uname -r`/build
# ./ccur-config -c -n
```

If you have built and are running a customized kernel configuration the kernel build environment should already have been set up when that custom kernel was built.

To build the driver and tests:

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurdprc
# make clobber     (perform cleanup)
# make               (make package and build the driver, library and tests)
```

*(Note: if you only wish to build the driver, you can enter the '**make driver**' command instead)*

After the driver is built, you will need to install the driver. This install process should only be necessary if the driver is re-built with changes.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurdprc
# make install     (install the driver software, library and man page)
```

Once the driver and the board are installed, you will need to ***load*** the driver into the running kernel prior to any access to the CCUR DPRC board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurdprc
# make load        (load the driver)
```

10. Software Support

- This driver package includes extensive software support and test programs to assist the user in communicating with the board. Refer to the PCIe Digital Programmable Resister Card (DPRC) Design Specification by Concurrent Real-Time for more information on the product.

10.1. Device Configuration

After the driver is successfully loaded, the device to card association file ***ccurdprc_devs*** will be created in the ***/usr/local/CCRT/drivers/ccurdprc/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccurdprc*** directory as well. If the user wishes to keep the default one-to-one device to card association, no further action is required. If the device to card association needs to be changed, this file can be edited by the user to associate a particular device number with a card number that was found by the driver. The commented portion on the top of the ***ccurdprc_devs*** file is automatically generated every time the user issues the ***'make load'*** or ***'/sbin/service ccurdprc start' (on RedHawk 6.5 and below)*** or ***'/usr/bin/systemctl start ccurdprc' (on RedHawk 7.0 and above)*** command with the current detected cards, information. Any device to card association edited and placed in this file by the user is retained and used during the next ***'make load'*** or ***'/sbin/service ccurdprc load'*** or ***'/usr/bin/systemctl start ccurdprc'*** process.

If the user deletes the ***ccurdprc_devs*** file and recreates it as an empty file and performs a ***'make load'*** or if the user does not associate any device number with card number, the driver will provide a one to one association of device number and card number. For more information on available commands, view the commented section of the ***ccurdprc_devs*** configuration file.

Warning: If you edit the ***ccurdprc_devs*** file to associate a device to a card, you will need to re-issue the ***'make load'*** or ***'/sbin/service ccurdprc start'*** or ***'/usr/bin/systemctl start ccurdprc'*** command to generate the necessary device to card association. This device to card association will be retained until the user changes or deletes the association. **If any invalid association is detected, the loading of the driver will fail**.

10.2. Library Interface

There is an extensive software library that is provided with this package. For more information on the library interface, please refer to the PCIe Digital Programmable Resister Card (DPRC) Software Interface by Concurrent Real-Time for more information.

10.3. Debugging

This driver has some debugging capability and should only be enabled while trying to trouble-shoot a problem. Once resolved, debugging should be disabled otherwise it could adversely affect the performance and behavior of the driver.

To enable debugging, the ***Makefile*** file in ***/usr/local/CCRT/drivers/ccurdprc/driver*** should be edited to un-comment the statement (*remove the preceding '#'*):

```
# BUILD_TYPE=debug
```

Next, use and install the debug driver

```
# cd /usr/local/CCRT/drivers/ccurdprc/driver
# make
# make install
```

Next, edit the ***ccurdprc_config*** file in ***/usr/local/CCRT/drivers/ccurdprc/driver*** to un-comment the statement (remove the preceding '#'):

```
# ccurdprc_debug_mask=0x00002040
```

Additionally, the value of the debug mask can be changed to suite the problem investigated. Once the file has been edited, the user can load the driver by issuing the following:

```
# cd /usr/local/CCRT/drivers/ccurdprc/driver
# make load
```

The user can also change the debug flags after the driver is loaded by passing the above debug statement directly to the driver as follows:

```
# echo “ccurdprc_debug_mask=0x00082047” > /proc/driver/ccurdprc
```

Following are the supported flags for the debug mask as shown in the ***ccurdprc_config*** file.

```
###############################################################################
#                                                                             #
#         D_ENTER         0x00000001  /* enter routine */                     #
#         D_EXIT          0x00000002  /* exit routine */                      #
#                                                                             #
#         D_L1            0x00000004  /* level 1 */                          #
#         D_L2            0x00000008  /* level 2 */                          #
#         D_L3            0x00000010  /* level 3 */                          #
#         D_L4            0x00000020  /* level 4 */                          #
#                                                                             #
#         D_ERR           0x00000040  /* level error */                      #
#         D_WAIT          0x00000080  /* level wait */                       #
#                                                                             #
#         D_INT0          0x00000100  /* interrupt level 0 */                #
#         D_INT1          0x00000200  /* interrupt level 1 */                #
#         D_INT2          0x00000400  /* interrupt level 2 */                #
#         D_INT3          0x00000800  /* interrupt level 3 */                #
#         D_INTW          0x00001000  /* interrupt wakeup level */           #
#         D_INTE          0x00002000  /* interrupt error */                  #
#                                                                             #
#         D_RTIME         0x00010000  /* display read times */               #
#         D_WTIME         0x00020000  /* display write times */              #
#         D_REGS          0x00040000  /* dump registers */                   #
#         D_IOCTL         0x00080000  /* ioctl call */                       #
#                                                                             #
#         D_DATA          0x00100000  /* data level */                       #
#         D_DMA           0x00200000  /* DMA level */                        #
#         D_DBUFF         0x00800000  /* DMA buffer allocation */            #
#                                                                             #
#         D_NEVER         0x00000000  /* never print this debug message */   #
#         D_ALWAYS        0xffffffff  /* always print this debug message */  #
#         D_TEMP          D_ALWAYS    /* Only use for temporary debug code */ #
###############################################################################
```

Another variable ***ccurdprc_debug_ctrl*** is also supplied in the ***ccurdprc_config*** that the driver developer can use to control the behavior of the driver. The user can also change the debug flags after the driver is loaded by passing the above debug statement directly to the driver as follows:

```
# echo “ccurdprc_debug_ctrl=0x00001234” > /proc/driver/ccurdprc
```

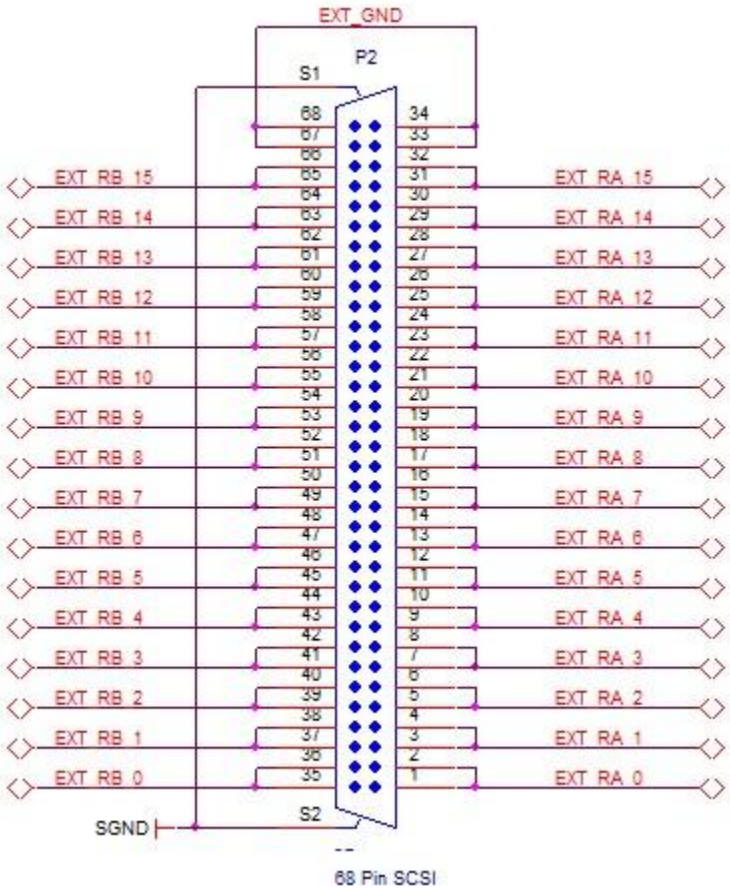
In order to make use of this variable, the driver must be coded to interrogate the bits in the ***ccurdprc_debug_ctrl*** variable and alter its behavior accordingly.

11. Notes and Errata

- In some kernel releases, when a package is installed or uninstalled, you may see a warning message on the system console similar to ***"systemd-rc-local-generator[22094]: /etc/rc.d/rc.local is not marked executable, skipping."***. This is for informational purpose only and can be ignored.
- If a kernel is configured with the CONFIG_DEBUG_LOCK_ALLOC define, the driver will fail to compile due to mutex_lock_nested() call being included with GPL requirement. If you want to successfully compile the driver, you will need to remove the CONFIG_DEBUG_LOCK_ALLOC define and rebuild the kernel.
- Ubuntu kernels RH8.0 onwards may have the default ***systemd-timesyncd*** daemon installed which does not accurately adjust the system.You may want to replace the default with the ***chrony*** package for a more accurate time asjustment.
- The board is designed to protect itself from excess voltage and current faults so as not to damage it. Programmable trip points are fine-tuned to perform this function. It is imperative that the user does not attempt to change these trip thresholds as that could damage the card.
- This card does not use interrupts or performs DMA.
- It is possible that *lspci* calls will still display the device with the old name of ***"Concurrent Computer Corporation"*** instead of ***"Concurrent Real-Time"*** if the OS has not been updated.
- The potentiometers must be enabled before the ADC's are enabled. The ADC's may generate an error if this sequence is not followed.
- The ADC's should also be disabled if the potentiometers are disabled to follow the sequence for re-enabling.
- A potentiometer is considered "activated" after the first resistance value has been written to it.
- A potentiometer is considered "powered down" if the test register has selected power down and the potentiometer is then activated.
- Calibration voltages (+2.5V, +10V & -10V) can only be selected if none of the potentiometers have been activated or if they are all powered down.
- Calibration currents (+8ma, -8ma & +16ma) can only be selected if none of the potentiometers are powered down.
- External I/O (including fault insertion) for a channel can only be selected if the potentiometer is active and if no sections are powered down or all sections are powered down.
- External fault switch testing for a channel can only be selected if the potentiometer is not active.
- The potentiometers must be disabled to de-activate them. This is the only way to restore any potentiometer from a powered down state or a forced failed condition.
- An electronic fuse trip condition will suspend all operations to the affected channel until the condition is cleared. The affected channels potentiometer will have to be re-written to restore the desired value.

Appendix A: External Connections and Pin-outs

The input/output signals from the DPRC are connected via an industry standard 68-pin SCSI type connector with the following pin-out:



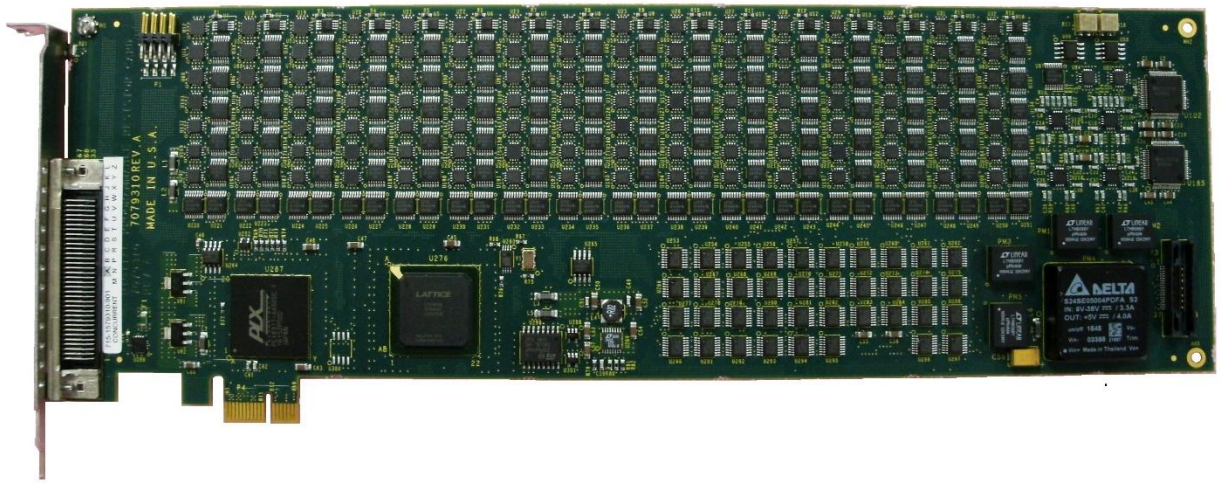
Appendix B: External Connections and Pin-outs

The DPRC has a single multicolor LED indicator located at the top front edge of the board visible via a hole in the front panel. If the board is in a reset state the indicator will be solid Red. After reset is complete, the indicator will cycle through Red, Green and Blue for approximately 1 second each as a lamp test. If the indicator remains Red after reset is complete it would indicate a board malfunction. Other states of the board during operation are indicated as follows:

Color	Description	Input/Outputs
Red	Board in Reset	Not Active
Green	Board Operational	Not Active
Blue	Board Operational	Active

1) The Green or Blue indicators will *flash* once per second if the Identify Board bit is set.
2) The Blue indicator will *blink* twice per second if any channel has been tripped offline with an electronic fuse condition.

Appendix C: The Digital Programmable Resistance Card



This page intentionally left blank