Release Notes
CCURAOCC (WC-DA3218)

Driver	ccuraocc (WC-DA3218)	
Platform	RedHawk Linux® *(CentOS/Rocky/RHEL & Ubuntu)*, Native Ubuntu® and Native Red Hat Enterprise Linux®[1]	
Vendor	Concurrent Real-Time	
Hardware	PCIe 8-Channel (CP-DA0818) or 32-Channel (CP-DA3218) DA Converter Card	
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 CCRT-PCIe-AOCC Linux ***ccuraocc*** driver and related software on the RedHawk OS, Native Ubuntu and Native Red Hat for use with the CCRT-PCIe-AOCC board. 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 ***ccuraocc*** man page.

The AOCC is an 8 or 32-channel 18-bit digital to analog converter card with a PCI express interface. It is implemented using Linear Technology LTC2758 dual channel DAC's. The PCI interface utilizes a PLX Technology PEX-8311AA PCI-express-to-local bus bridge. There is a Lattice ECP2M FPGA for control of board functions including registers and storage. An adjustable main clock source is generated by a low jitter PLL. The external synchronizing interface consists of LVDS signaling connected via RJ-12 (6-pin phone) style cabling.

Features and Characteristics of the AOCC are:

- 8 or 32-channel 18-bit D to A Conversion.
- Differential or Single-ended Output (Build Option).
- 0 to +5V, 0 to +10V, +/-2.5V, +/-5V or +/-10V Output Range Selection.
- 10 Milliamp Maximum Output Drive.
- 400K Updates Per Second.
- Industry Standard SCSI 68-pin Connector for Inputs.
- RJ-12 (6-pin phone style) Connectors for External Synchronization.
- PCI Express x1 Revision 1.0a.
- Supports MSI Interrupts.
- Low Jitter Phase Lock Loop (PLL) Clock Generator.
- Supports Multi-board Synchronization.
- Directly Addressable Conversion Data Registers.
- 128K Word Conversion Data FIFO with DMA.
- Low Noise Analog Power Generation.
- On Board Calibration ADC.
- Gain and Offset Calibration DAC's Per Channel.
- Gain and Offset Calibration Values Fully Accessible.
- Non-volatile Storage of Calibration Data & User Configuration.
- NIST Traceable Calibration Standard.

The board and driver provide support for MSI interrupts. This is the default configuration.

2. Requirements

- CCRT-AOCC 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 8-Channel or 32-Channel Digital to Analog Output Converter Card (AOCC) Software Interface 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/ccuraocc/driver/ccuraocc.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 *ccuraocc_nomsi* parameter located in the …/*driver/ccuraocc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccuraocc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccuraocc_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

It is possible that some tests may get overflow or underflow errors as the card is capable of high sample rate transfers. You may need to lower the sample rates for these tests to run successfully if BIOS and kernel level tuning does not help.

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/ccuraocc/driver/ccuraocc.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 *ccuraocc_nomsi* parameter located in the …/*driver/ccuraocc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccuraocc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccuraocc_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

It is possible that some tests may get overflow or underflow errors as the card is capable of high sample rate transfers. You may need to lower the sample rates for these tests to run successfully if BIOS and kernel level tuning does not help.

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 CCRT-AOCC card is a x1 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.

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.***

The ***ccuraocc*** 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.

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

```
# lspci -v -d1542:9287
02:04.0 System peripheral: Concurrent Computer Corporation Device 9287 (rev 01)
        Subsystem: PLX Technology, Inc. Device 9056
        Flags: bus master, 66MHz, medium devsel, latency 96, IRQ 88
```

```
        Memory at c0100800 (32-bit, non-prefetchable) [size=512]
        Memory at c0100000 (32-bit, non-prefetchable) [size=2K]
        Capabilities: <access denied>

# lsirq
88      02:04.0 Concurrent Computer Corporation Unknown device (rev 01)
```

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.

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 1542:9287
```

For each CCRT-AOCC 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:

```
02:04.0 System peripheral: Concurrent Computer Corporation Device 9287 (rev 01)
```

If a line similar to 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. Add Device to Restricted List

For kernels that have *iommu* enabled, these devices will fail DMA read and write access with a message similar to the following:

```
DMAR: [DMA Write] Request device [1d:00.0] fault addr 5eec0000
      [fault reason 01] Present bit in root entry is clear
```

You can issue the '*cat /proc/cmdline*' command to determine if *iommu* is enabled in the kernel after booting the system. If you see the *'intel_iommu=on'* entry, the kernel has *iommu* enabled for the entire operating system. In this case you will need to restrict *iommu* usage for these devices.

To enable DMA to work, you will need to add the following entries to the kernel grub line:

1. iommu=pt *(this passthrough option is needed for restricting the selected device)*
2. intel_iommu.blacklist_ids=1542:9287 *(vendor:device id if 9287 card is installed)*

Use the ***ccur-grub2*** or ***blscfg*** command depending on the loaded kernel. You can use the following argument ***'--help'*** to either command for additional information on its usage.

1. ccur-grub2 --kopt-add iommu=pt 0,1,2 *(for kernel entries 0, 1 and 2)*
2. ccur-grub2 --kopt-add intel_iommu.blacklist_ids=1542:9287 0,1,2

Reboot the kernel for the device restriction to take effect. You should get a message similar to the one below if the command took effect:

DMAR: add [1542:9287] to intel_iommu blacklist

Note!!!

If you wish to disable *iommu* for all devices under a PLX bridge you can use the following option instead:

1. intel_iommu=on,plx_off

If you want DMA to work for kernels that do *not* support *plx_off* or *intel_iommu.blacklist_ids* you will need to disable iommu in the kernel.

1. intel_iommu=off

6.3. Software Installation

Concurrent Real-Time™ port of the ***ccuraocc*** software is distributed in RPM and DEB format on a DVD. Source for the API library, example test programs, and kernel loadable driver are included, as is documentation in PDF format.

The software is installed in the **/usr/local/CCRT/drivers/ccuraocc** 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 ***ccuraocc*** package, load the DVD installation media and issue the following commands as the **root** user. The system should auto-mount the DVD to a mount point in the **/media** or **/run/media** directory based on the DVD's volume label – in this case **ccuraocc_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/ccuraocc_driver
      --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/ccuraocc_driver
      --- or on Ubuntu RedHawk ---
# cd /media/[user_name]/ccuraocc_driver

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

# cd /
# eject
```

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

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

```
# cat /proc/ccuraocc

Version           : 23.3.2
Built             : Tue Oct  1 12:58:57 EDT 2019
Boards            : 2
  card=0: [86:04.0] bus=134, slot=4, func=0, irq=77, msi=1, ID=691349, BoardInfo=0x92870203
                                                                          (32ch/Single-Ended)
  card=1: [89:04.0] bus=137, slot=4, func=0, irq=78, msi=1, ID=652005, BoardInfo=0x92870103
                                                                          (32ch/Differential)
```

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

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

6.4. Software Removal

The ***ccuraocc*** 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*** from the installation DVD:

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

```
=== as root ===
# rpm -e ccuraocc   (driver unloaded, uninstalled, and deleted – on an RPM based system)
--- or ---
# dpkg -P ccuraocc  (driver unloaded, uninstalled, and deleted – on a 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/ccuraocc
# make unload    (unload the driver from the kernel)
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccuraocc stop
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl stop ccuraocc
      --- or on Ubuntu RedHawk ---
# /bin/systemctl stop ccuraocc
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuraocc
# 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/ccuraocc*** directory later to re-install and re-load the driver.

7. Auto-loading the Driver

The ***ccuraocc*** 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 directory 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 ccuraocc        (enable auto-loading of the driver)
# /sbin/chkconfig --del ccuraocc        (disable auto-loading of the driver)
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl enable ccuraocc    (enable auto-loading of the driver)
# /usr/bin/systemctl disable ccuraocc   (disable auto-loading of the driver)
      --- or on Ubuntu RedHawk ---
# /bin/systemctl enable ccuraocc        (enable auto-loading of the driver)
# /bin/systemctl disable ccuraocc       (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/ccuraocc
# make test                          (build the test programs)
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuraocc
# make test                                    (build the test programs)
# ./test/ccuraocc_dump                         (dump all board resisters)
# ./test/ccuraocc_rdreg                        (display board resisters)
# ./test/ccuraocc_reg                          (Display board resisters)
# ./test/ccuraocc_regedit                      (Interactive board register editor test)
# ./test/ccuraocc_tst                          (Interactive test to test driver and board)
# ./test/ccuraocc_wreg                         (edit board resisters)
# ./test/lib/ccuraocc_calibrate                (library: get/set board calibration)
# ./test/lib/ccuraocc_compute_pll_clock        (library: compute pll clock)
# ./test/lib/ccuraocc_disp                     (library: display channel data)
# ./test/lib/ccuraocc_identify                 (library: identify board)
# ./test/lib/ccuraocc_setchan                  (library: generate waves in various
                                                modes)
# ./test/lib/ccuraocc_smp_affinity             (library: display/set IRQ CPU affinity)
# ./test/lib/ccuraocc_sshot                    (library: performance of channel write
                                                modes)
# ./test/lib/ccuraocc_tst_lib                  (library: Interactive test to test driver and
                                                board)
# ./test/lib/ccuraocc_volt                     (library: validate voltage conversion
                                                routines)
# ./test/lib/Sprom/ccuraocc_sprom              (library: serial prom view/update
                                                calibration utility)
```

9. Re-building the Driver, Library and Tests

If for any reason the user needs to manually rebuild and load an *installed* ***rpm*** 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/ccuraocc
# 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/ccuraocc
# 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 CCURAOCC board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuraocc
# 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 *Concurrent Real-Time PCIe 8-Channel or 32-Channel Digital to Analog Output Converter Card (AOCC) Software Interface* document for more information on the product.

10.1. Device Configuration

After the driver is successfully loaded, the device to card association file ***ccuraocc_devs*** will be created in the ***/usr/local/CCRT/drivers/ccuraocc/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccuraocc*** 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 ***ccuraocc_devs*** file is automatically generated every time the user issues the ***'make load'*** or ***'/sbin/service ccuraocc start' (on RedHawk 6.5 and below)*** or ***'systemctl start ccuraocc' (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'***, ***'/sbin/service ccuraocc start',*** or ***'systemctl start ccuraocc'*** process.

If the user deletes the ***ccuraocc_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 ***ccuraocc_devs*** configuration file.

Warning: If you edit the ***ccuraocc_devs*** file to associate a device to a card, you will need to re-issue the ***'make load', '/sbin/service ccuraocc start',*** or ***'/usr/bin/systemctl start ccuraocc'*** 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. Associate Device Names to Cards

By default, this driver creates a two device names for each board found in the system. The name of the devices are /*dev/ccuraocc<bno>* and */dev/ccuraocc_wave<bno>* where <bno> corresponds the card number found in the system. An optional ***aoccstream_wave*** package may be purchased separately that contains an API to interface to this driver and generate user defined waves. This AOCCStream API only opens the */dev/ccuraocc_wave<bno>*. If the user needs to change the association of device names to cards, they need to edit the *ccuraocc_devs* that is created by the driver and located in the */usr/local/CCRT/drivers/ccuraocc/driver* directory and provide the device to card association. e.g. if we have 6 cards in a system and we need to perform wave generation on three of the cards only, then we would do something like:

```
device=0            card=0
device=1            card=1
device=2            card=2
device_wave=0card=3
device_wave=1card=4
device_wave=2card=5
device=4          ID=12345678
```

The following devices will be created:

/dev/ccuraocc0, /dev/ccuraocc1 and /dev/ccuraocc2 for boards that are not planning to use the AOCCStream API.

/dev/ccuraocc_wave0, /dev/ccuraocc_wave1 and /dev/ccuraocc_wave2 for boards that are planning to use the AOCCStream API.

/dev/ccuraocc4 will be assigned to the board that has a board serial number of 12345678.

NOTE: The wave files and AOCCStream API is only available for CCURAOCC cards.

10.3. 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 8-Channel or 32-Channel Digital to Analog Output Converter Card (AOCC) Software Interface by Concurrent Real-Time* document.

10.4. Calibration

Warning: Whenever auto-calibration is performed, the channel outputs will be affected. It is important that prior to calibration, any sensitive equipment be disconnected; otherwise it could result in damage to the equipment.

Several library calls are provided to assist the user in calibrating the board. Additionally, the board contains factory calibration information for each of the output voltage ranges. Users can view this information using the supplied API or the serial prom test utility *ccuraocc_sprom*. Though the API and test utility provides capability to edit and change the factory calibration, users should refrain from making any changes to it, as it will no longer reflect the factory calibration shipped with the card. Users can use the factory calibration to restore the calibration information stored for each configured channel prior to commencing a test run. The restore API will update the calibration information for all the channels based on their current voltage range. Note that the factory calibration values were obtained under specific conditions, such as temperature, that may not be the same as the user application. In most cases it will always be better to perform auto-calibration after the board is stabilized in the user environment.

Additionally, the users can perform up to two independent user controlled checkpoints where the active channel configuration and calibration information is stored in the serial prom for all the channels. At any time, the user can restore either of the two checkpoints with an API call or the serial prom test utility *ccuraocc_sprom* prior to a test run. These checkpoints will allow the user to store specific values pertaining to their calibration conditions.

10.5. Firmware Updates

This board is capable of being re-programmed in the field as new firmware updates are made available by *Concurrent Real-Time™*. The procedure for re-programming the firmware will be supplied to the user at the time when a firmware update is necessary.

10.6. 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/ccuraocc/driver*** should be edited to un-comment the statement (*remove the preceding '#'*):

```
#BUILD_TYPE=debug
```

Next, compile and install the driver

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

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

```
# ccuraocc_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/ccuraocc/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 "ccuraocc_debug_mask=0x00082047" > /proc/driver/ccuraocc
```

Following are the supported flags for the debug mask as shown in the ***ccuraocc_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 ***ccuraocc_debug_ctrl*** is also supplied in the ***ccuraocc_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 “ccuraocc_debug_ctrl=0x00001234” > /proc/driver/ccuraocc
```

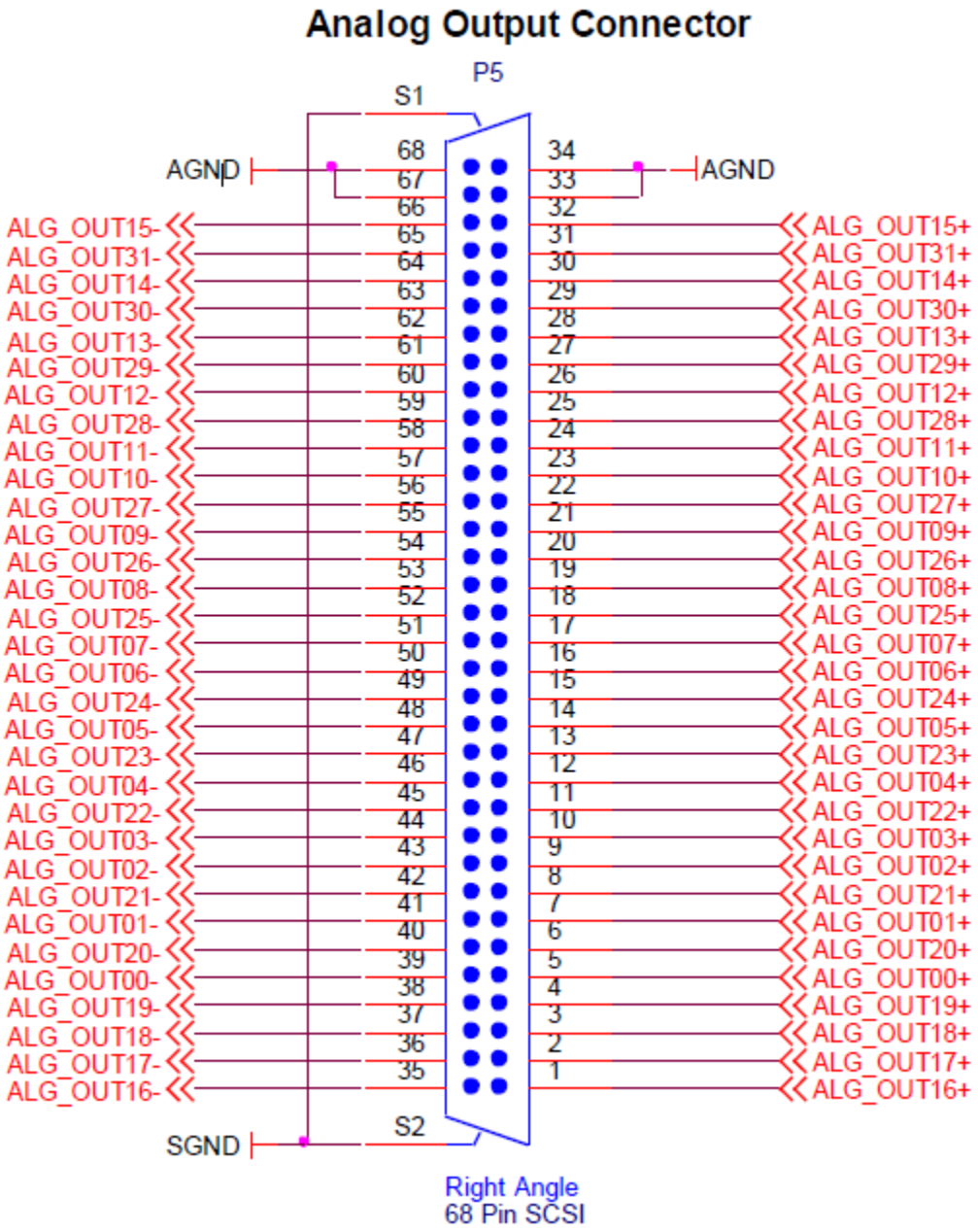
To make use of this variable, the driver must be coded to interrogate the bits in the ***ccuraocc_debug_ctrl*** variable and alter its behavior accordingly.

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 can be ordered as an 8-Channel or 32-channel single-ended or differential card.
- Driver and board supports MSI interrupts. The default configuration is to perform MSI interrupts.
- When writing to channel registers, you need to first reset the FIFO as contents from the FIFO could override the outputs.
- Some new SuperMicro Mother Boards (X11SPA-TF) have a problem with supporting MSI interrupts on these cards. The driver detects this problem and attempts to switch to alternate MSI support. If that also fails, then wired interrupts configured by the driver. If the board detects this issue, an appropriate error message is inserted in the kernel log message (which can be viewed with the command *dmesg*).
- On some kernel logs, you may see warnings about module verifications and tainted kernel. These can be ignored as they are generated due to the fact that this is a proprietary driver.
- On some SuperMicro Mother Boards, if the BIOS has enabled VT-d MSI interrupt remapping, there is a problem with some kernels where interrupts will not be generated due to source-id verification failure. Currently, the driver has implemented hooks into the RedHawk 6.5 onwards kernels to fix this problem.
- 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:

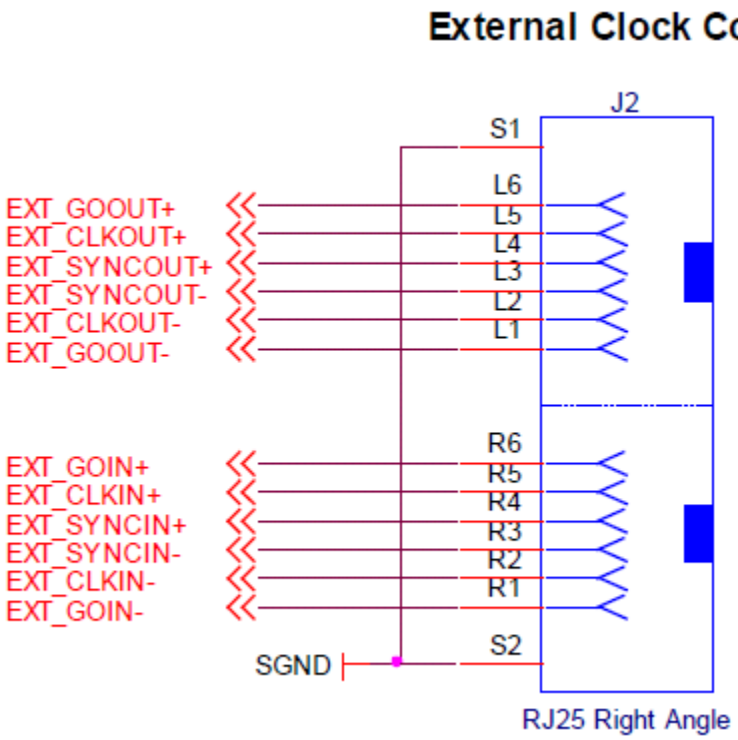
    - Reload the kernel with the grub option "intremap=nosid"
    - Reload the kernel with the grub option "intremap=off"
    - Disable VT-d in the BIOS
    - Disable VT-d Msi Interrupt Remapping in the BIOS
    - Disable 4G Decoding in the BIOS

Appendix A: External Connections and Pin-outs

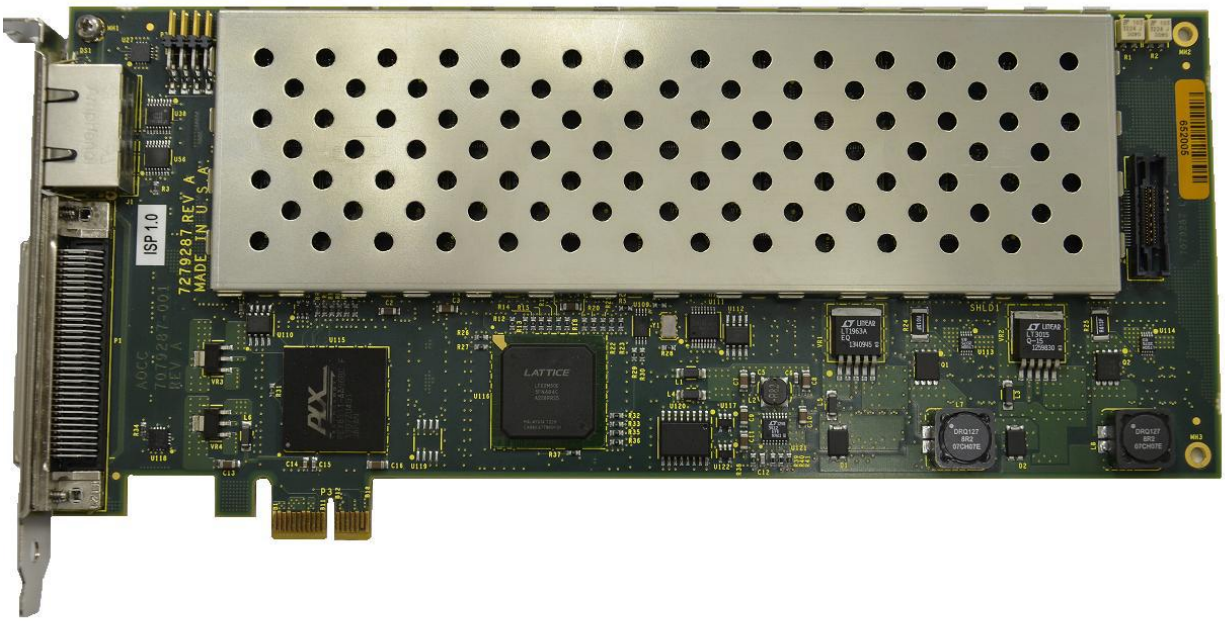


>>>An analog ground connection is always required for the ESD and over/under voltage protection circuits to function correctly.

The multi-board clock/synchronization signals connect AOCC boards together via two industry standard RJ-12 (6-pin phone style) connectors with the following pin-out:



Appendix B: The Board



CCRT-PCIe-AOCC Card

This page intentionally left blank