Release Notes CCURAOCC (WC-DA3218)



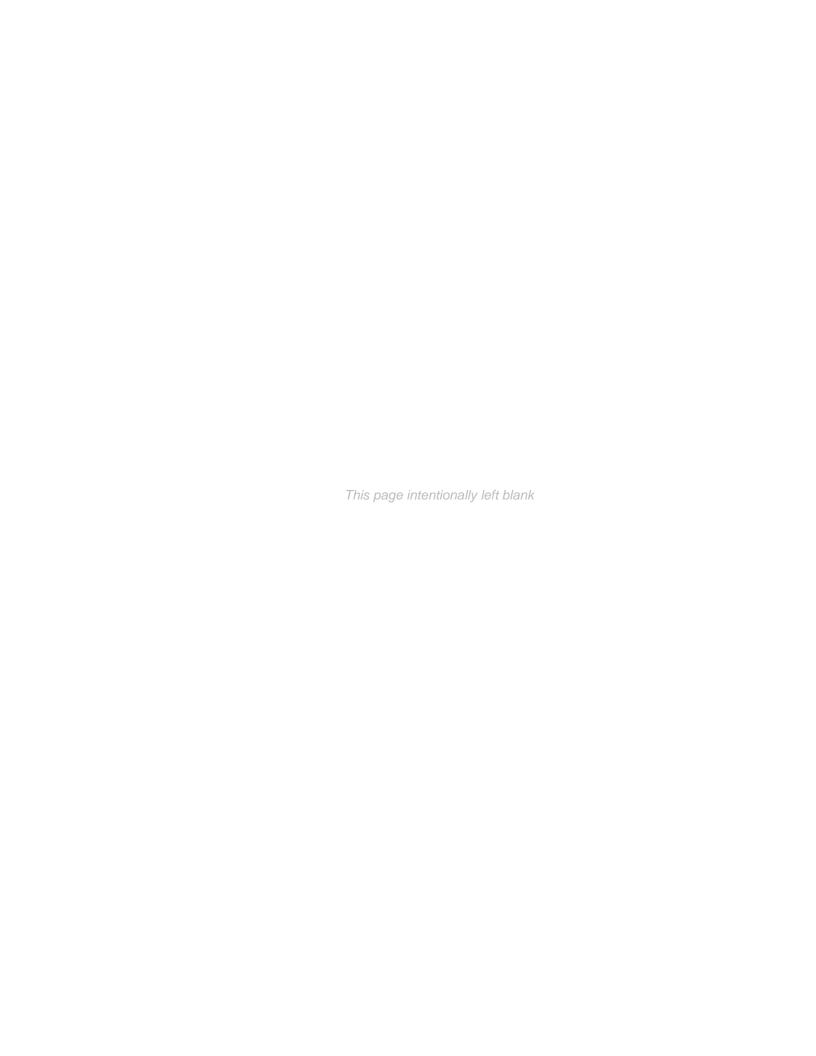
Driver	ccuraocc (WC-DA3218)	
OS	RedHawk (CentOS/Rocky or Ubuntu based)	
Vendor	Concurrent Real-Time	
Hardware	PCIe 8-Channel (CP-DA0818) or 32-Channel (CP-DA3218) DA Converter Card	
Author	Darius Dubash	
Date	October 3 rd , 2023	Rev 2023.1





Table of Contents

1.	INT	FRODUCTION	1
2.	RE	QUIREMENTS	1
3.	DO	CUMENTATION	1
4.	INS	STALLATION AND REMOVAL	2
4.	1.	Hardware Installation	2
4.	2.	Add Device to Restricted List	2
4.	3.	Software Installation	3
4.	4.	Software Removal	4
5.	AU'	TO-LOADING THE DRIVER	5
6.	TES	STING AND USAGE	5
7.	RE-	-BUILDING THE DRIVER, LIBRARY AND TESTS	6
8.	SO	FTWARE SUPPORT	7
8.	1.	Device Configuration	7
8.	2.	Associate Device Names to Cards	7
8.	3.	Library Interface	8
8.	4.	Calibration	8
8.	5.	Firmware Updates	8
8.	6.	Debugging	8
9.	NO	OTES AND ERRATA	.11
API	PEN	IDIX A: EXTERNAL CONNECTIONS AND PIN-OUTS	12
API	PEN	DIX B: THE BOARD	. 14



1. Introduction

This document assists the user in installing the CCRT-PCle-AOCC Linux *ccuraocc* driver and related software on the RedHawk OS for use with the CCRT-PCle-AOCC board. The directions in this document supersede all others – they are specific to installing the software on Concurrent Real-Time's RedHawk systems. Other information provided as part of this release, when it may contradict these directions, should be ignored and these directions should prevail.

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. 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. Installation and Removal

4.1. 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.



<u>Caution</u>: 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 'Ispci -v' or the 'Isirq' command can be used to determine the IRQs of various devices in the system.

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 **Ispci** 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.

4.2. 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:

iommu=pt (this passthrough option is needed for restricting the selected device)
 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

4.3. 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.



<u>Warning:</u> Before installing the software, the kernel 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:

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

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

4.4. 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:

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.

5. 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/system/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 ---
                                                (enable auto-loading of the driver)
# /sbin/chkconfig --add ccuraocc
# /sbin/chkconfig --del ccuraocc
                                               (disable auto-loading of the driver)
       --- or on RedHawk 7.0 and above ---
                                                (enable auto-loading of the driver)
# /usr/bin/systemctl enable ccuraocc
                                                (disable auto-loading of the driver)
# /usr/bin/systemctl disable ccuraocc
       --- or on Ubuntu RedHawk ---
                                         (enable auto-loading of the driver)
# /bin/systemctl enable ccuraocc
# /bin/systemctl disable ccuraocc
                                         (disable auto-loading of the driver)
```

6. Testing and Usage

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

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)

Concurrent Real-Time, Inc.™ ccuraocc Driver for RedHawk Linux™ – Release Notes
```

(display board resisters) # ./test/ccuraocc rdreg # ./test/ccuraocc reg (Display board resisters) (Interactive board register editor test) # ./test/ccuraocc regedit (Interactive test to test driver and board) # ./test/ccuraocc tst # ./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) (library: display channel data) # ./test/lib/ccuraocc disp (library: identify board) # ./test/lib/ccuraocc identify # ./test/lib/ccuraocc setchan (library: generate waves in various modes) (library: display/set IRQ CPU affinity) # ./test/lib/ccuraocc smp affinity (library: performance of channel write # ./test/lib/ccuraocc sshot modes) (library: Interactive test to test driver and # ./test/lib/ccuraocc tst lib board) # ./test/lib/ccuraocc volt (library: validate voltage conversion routines) # ./test/lib/Sprom/ccuraocc sprom (library: serial prom view/update calibration utility)

7. 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.



<u>Warning:</u> Before installing the software, the kernel 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)
```

8. 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 PCle 8-Channel or 32-Channel Digital to Analog Output Converter Card (AOCC) Software Interface* document for more information on the product.

8.1. Device Configuration

After the driver is successfully loaded, the device to card association file <code>ccuraocc_devs</code> will be created in the <code>/usr/local/CCRT/drivers/ccuraocc/driver</code> directory, if it did not exist. Additionally, there is a symbolic link to this file in the <code>/usr/lib/config/ccuraocc</code> 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 <code>ccuraocc_devs</code> file is automatically generated every time the user issues the 'make <code>load</code>' or '/sbin/service <code>ccuraocc start</code>' (on <code>RedHawk 6.5 and below</code>) or 'systemctl start <code>ccuraocc</code>' (on <code>RedHawk 7.0 and above</code>) 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 <code>load</code>', '/sbin/service <code>ccuraocc start</code>', or 'systemctl start <code>ccuraocc</code>' 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.



<u>Warning:</u> 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.

8.2. 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:

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.

8.3. 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.

8.4. Calibration



<u>Warning</u>: 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 <code>ccuraocc_sprom</code>. 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.

8.5. Firmware Updates

This board is capable of being re-programmed in the field as new firmware updates are made available by $Concurrent\ Real\text{-}Time^{\intercal M}$. The procedure for re-programming the firmware will be supplied to the user at the time when a firmware update is necessary.

8.6. 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.

```
0x00000004 /* level 1 */
0x00000008 /* level 2 */
        D L1
        D_L2
        D_L3
                    0x00000010 /* level 3 */
        ^{-}L4
                    0x00000020 /* level 4 */
        0x00000100 /* interrupt level 0 */
0x00000200 /* interrupt level 1 */
0x00000400 /* interrupt level 2 */
0x00000800 /* interrupt level 3 */
0x00001000 /* interrupt wakeup level */
0x00002000 /* interrupt error */
        D INTO
        D INT1
        D INT2
        D_INT3
        D_INTW
        D INTE
       D_DATA
                    0x00100000 /* data level */
        D_NEVER 0x00000000 /* never print this debug message */
D_ALWAYS 0xfffffff /* always print this debug message */
D_TEMP D_ALWAYS /* Only use for temporary debug code?
#
                     D ALWAYS /* Only use for temporary debug code */
```

Another variable <code>ccuraocc_debug_ctrl</code> is also supplied in the <code>ccuraocc_config</code> 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

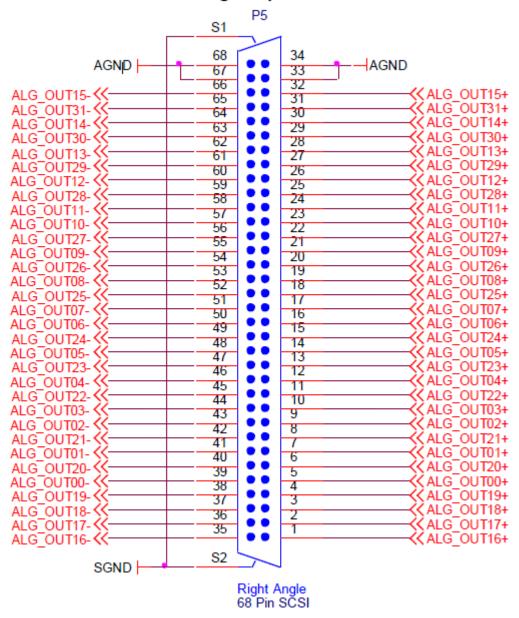
To make use of this variable, the driver must be coded to interrogate the bits in the **ccuraocc_debug_ctrl** variable and alter its behavior accordingly.

9. 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

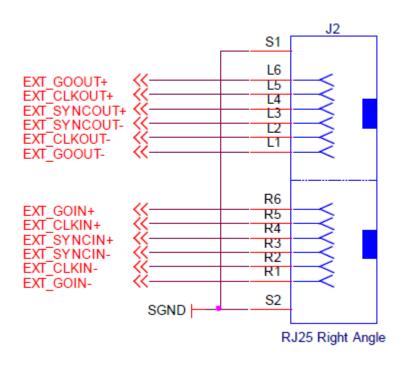
Analog Output Connector



>>>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:

External Clock Connector



Appendix B: The Board



CCRT-PCIe-AOCC Card

