Release Notes
CCURAOCC (WC-DA3218)

Driver	ccuraocc (WC-DA3218)	
OS	RedHawk (CentOS 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 2nd, 2020	rev 2020.2

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 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 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.

⚠️ *__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.

4.2.   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, 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:

```
== 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 an RPM based system)
    --- or ---
# dpkg –i ccuraocc_RedHawk_driver*.deb   (on a Debian 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
```

When the driver is loaded with the /sbin/service or /usr/bin/systemctl call, automatic calibration will commence for all the cards installed in the system and run in the background. This is also true when the system is rebooted. Issuing the 'make load' in the driver directory will *not* initiate an automatic calibration.

4.3.   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.

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/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)
```

6. 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)
```

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.

> ⚠️ ***Warning:*** 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 PCIe 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 **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**.

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:

```
device=0          card=0
device=1          card=1
device=2          card=2
device_wave=0     card=3
device_wave=1     card=4
device_wave=2     card=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.

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

> ⚠️ **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.

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-Time™*. 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.

```
##############################################################################
#                                                                            #
#         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:

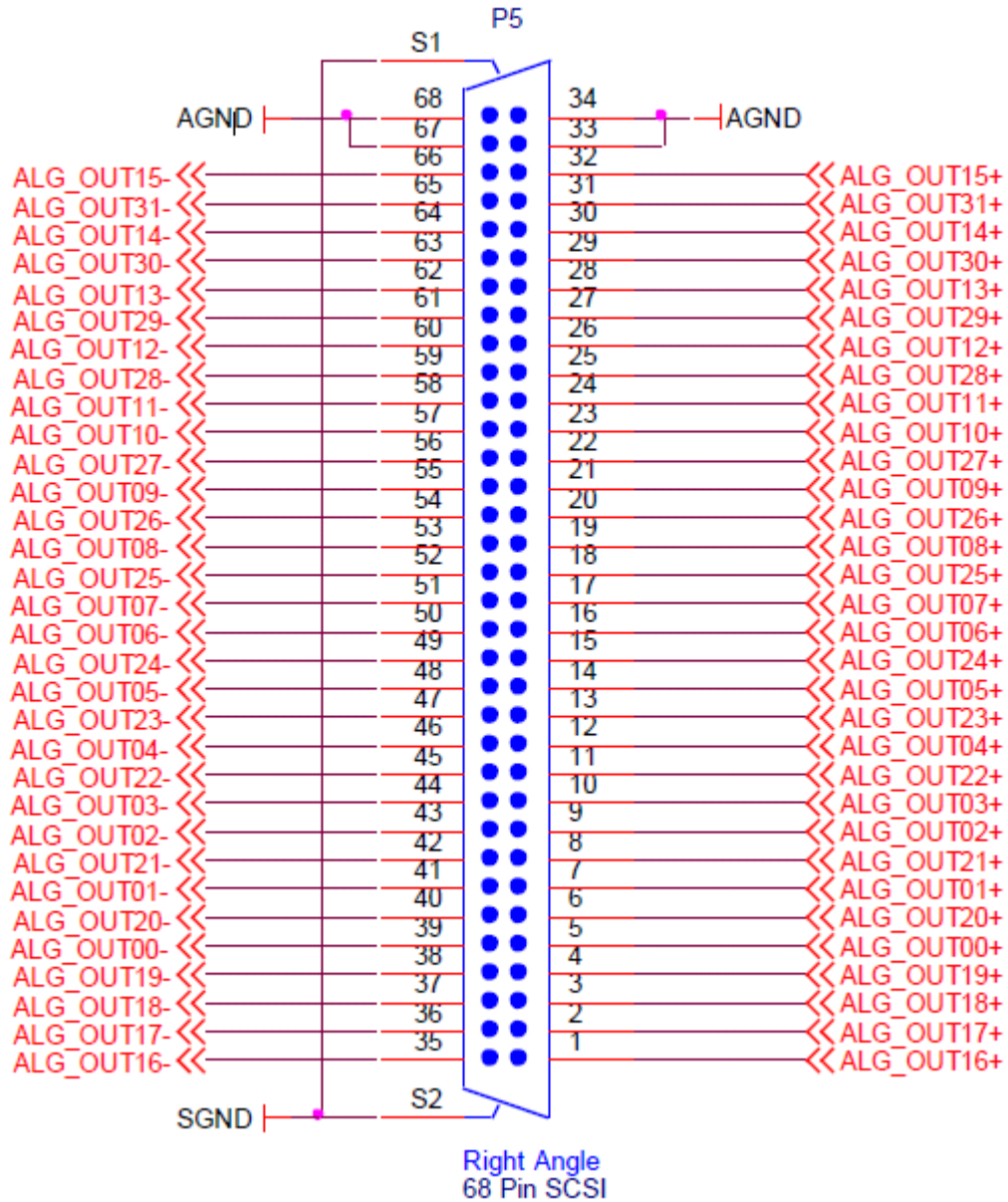   # **echo "ccuraocc_debug_ctrl=0x00001234" > /proc/driver/ccuraocc**

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.

9. Notes and Errata

- 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
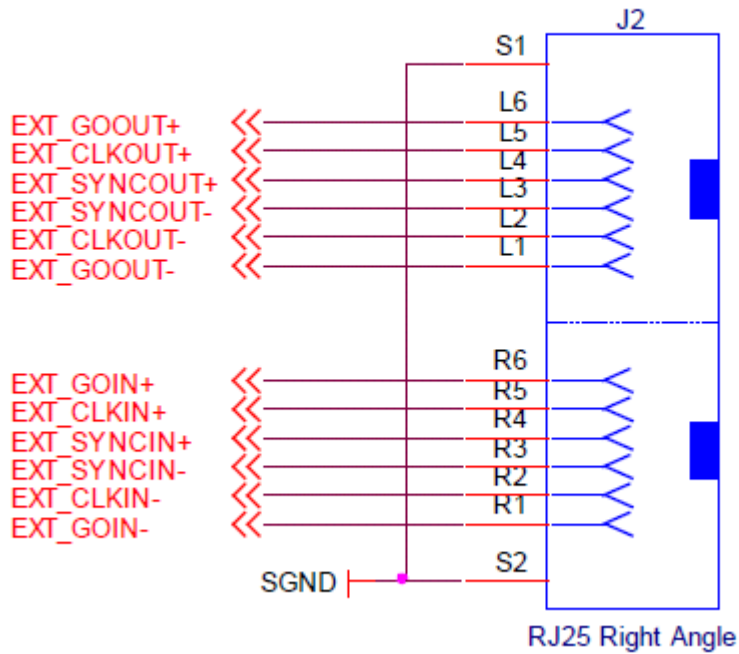
Appendix A: External Connections and Pin-outs

Analog Output Connector

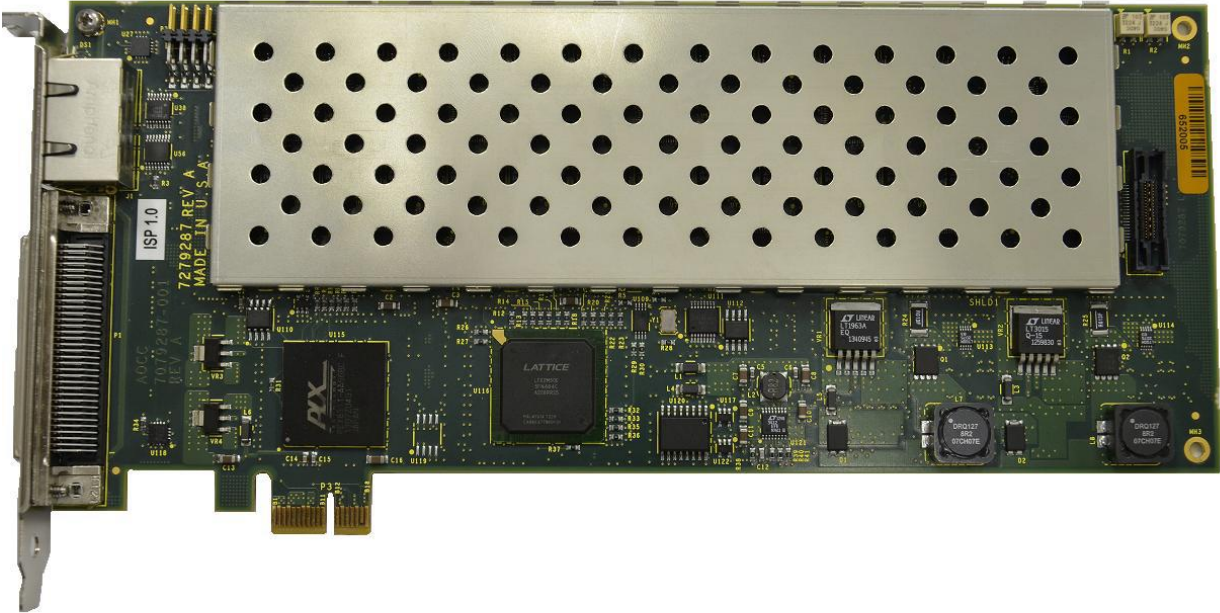|			
AGND	68	34	AGND
	67	33	
	66	32	
ALG_OUT15-	65	31	ALG_OUT15+
ALG_OUT31-	64	30	ALG_OUT31+
ALG_OUT14-	63	29	ALG_OUT14+
ALG_OUT30-	62	28	ALG_OUT30+
ALG_OUT13-	61	27	ALG_OUT13+
ALG_OUT29-	60	26	ALG_OUT29+
ALG_OUT12-	59	25	ALG_OUT12+
ALG_OUT28-	58	24	ALG_OUT28+
ALG_OUT11-	57	23	ALG_OUT11+
ALG_OUT10-	56	22	ALG_OUT10+
ALG_OUT27-	55	21	ALG_OUT27+
ALG_OUT09-	54	20	ALG_OUT09+
ALG_OUT26-	53	19	ALG_OUT26+
ALG_OUT08-	52	18	ALG_OUT08+
ALG_OUT25-	51	17	ALG_OUT25+
ALG_OUT07-	50	16	ALG_OUT07+
ALG_OUT06-	49	15	ALG_OUT06+
ALG_OUT24-	48	14	ALG_OUT24+
ALG_OUT05-	47	13	ALG_OUT05+
ALG_OUT23-	46	12	ALG_OUT23+
ALG_OUT04-	45	11	ALG_OUT04+
ALG_OUT22-	44	10	ALG_OUT22+
ALG_OUT03-	43	9	ALG_OUT03+
ALG_OUT02-	42	8	ALG_OUT02+
ALG_OUT21-	41	7	ALG_OUT21+
ALG_OUT01-	40	6	ALG_OUT01+
ALG_OUT20-	39	5	ALG_OUT20+
ALG_OUT00-	38	4	ALG_OUT00+
ALG_OUT19-	37	3	ALG_OUT19+
ALG_OUT18-	36	2	ALG_OUT18+
ALG_OUT17-	35	1	ALG_OUT17+
ALG_OUT16-			ALG_OUT16+

SGND

Right Angle
68 Pin SCSI

>>>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

	J2
S1	
EXT_GOOUT+	L6
EXT_CLKOUT+	L5
EXT_SYNCOUT+	L4
EXT_SYNCOUT-	L3
EXT_CLKOUT-	L2
EXT_GOOUT-	L1
EXT_GOIN+	R6
EXT_CLKIN+	R5
EXT_SYNCIN+	R4
EXT_SYNCIN-	R3
EXT_CLKIN-	R2
EXT_GOIN-	R1
SGND	S2

RJ25 Right Angle

Appendix B: The Board

CCRT-PCIe-AOCC Card