Release Notes
CCRTAICC (WC-ADS6418)

Driver	CCRTAICC (WC-ADS6418)	
Platform	RedHawk Linux® *(CentOS/Rocky/RHEL & Ubuntu)*, Native Ubuntu® and Native Red Hat Enterprise Linux®[1]	
Vendor	Concurrent Real-Time	
Hardware	PCIe 64-Channel Analog Input Converter Card (CP-ADS6418)	
Author	Darius Dubash	
Date	November 10th, 2025	Rev 2025.2

concurrent REAL-TIME

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

Table of Contents

1. Introduction

This document assists the user in installing the CCRT-PCIe-AICC Linux ***ccrtaicc*** driver and related software on the RedHawk OS, Native Ubuntu and Native Red Hat for use with the CCRT-PCIe Analog Input I/O Card (***AICC***). 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)

This release provides the optional support for the new Cloning of data among peripheral components and main system *(CCRT US Patent US 11.281.584 B1, Inventor Darius Dubash)*.

For additional information on the driver and its usage, refer to the ***ccrtaicc*** man page.

The AICC is a 64-Channel 18-bit Analog Input card with a PCI express interface.

Features and Characteristics of the ***AICC*** are:

- Cyclone V series FPGA control
- 64-channel 18-bit Analog-to-Digital Conversion
- Differential or Single-ended Input
- 0-5.12V, 0-10V, +/-5V ,+/-10V Input Range
- 1 Meg ohm Input Impedance
- +/-50V Input Over Voltage Protection
- 500Khz Maximum Sampling Rate (maximum 64-channels – see notes)
- 700Khz Maximum Sampling Rate (maximum 32-channels – High Speed Mode: see Notes)
- 1000Khz Maximum Sampling Rate (maximum 32-channels – Multiplexed Mode: see Notes)
- Dual DMA Engines
- Programmable Clock Generator
- Temperature Compensated Oscillator (TCXO)
- Multi-board Synchronization
- In System Firmware Update
- PCI Express Gen 1 x4 Lane
- MSI Interrupts
- Low Noise Analog Power Generation
- In System Calibration
- NIST Traceable Calibration Standard
- Directly Addressable Conversion Data Registers
- 128K Word Conversion Data FIFO's with DMA
- Industry Standard Very High Density SCSI 68-pin Connectors
- RJ-45 Synchronization Connectors

2. Requirements

- CP-ADS6418 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 64-Channel Analog Input I/O Card (AICC) 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/ccrtaicc/driver/ccrtaicc.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 *ccrtaicc_nomsi* parameter located in the *…/driver/ccrtaicc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccrtaicc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccrtaicc_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 HyperThread in BIOS.
To check for number of hyperthreads in system:

```
lscpu | grep "Thread(s)"
```

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/ccrtaicc/driver/ccrtaicc.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 *ccrtaicc_nomsi* parameter located in the *…/driver/ccrtaicc_config* file where the driver is installed. Reloading the driver will cause the MSI interrupt handling option to switch.

  - ccrtaicc_nomsi=0 enable MSI support *(default for RedHawk systems)*
  - ccrtaicc_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 and 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 HyperThread in BIOS.
To check for number of hyperthreads in system:
lscpu | grep "Thread(s)"

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 CP-ADS6418 card is a Gen 1 PCI Express product and is compatible with any PCI Express slot. The board must be installed in the system before attempting to use the driver.

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

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

```
# lspci -v -d 1542:9350

03:00.0 System peripheral: Concurrent Real-Time Device 9350 (rev 01)
        Subsystem: Concurrent Real-Time Device 0100
        Physical Slot: 5
        Flags: bus master, fast devsel, latency 0, IRQ 59
        Memory at bd520000 (32-bit, non-prefetchable) [size=32K]
        Memory at bd500000 (32-bit, non-prefetchable) [size=128K]
        Capabilities: [50] MSI: Enable+ Count=1/4 Maskable- 64bit+
        Capabilities: [78] Power Management version 3
        Capabilities: [80] Express Endpoint, MSI 00
        Capabilities: [100] Virtual Channel
        Capabilities: [200] Vendor Specific Information: ID=1172 Rev=0 Len=044 <?>
        Capabilities: [800] Advanced Error Reporting
```

```
# lsirq

        59      03:00.0 Concurrent Real-Time Unknown device (rev 01)
```

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

```
# lspci -d 1542:9350
```

For each CP-ADS6418 PCIe board installed, a line like one of the following will be printed, depending on the revision of the system's ***/usr/share/hwdata/pci.ids*** file:

```
03:00.0 System peripheral: Concurrent Real-Time Device 9350 (rev 01)
```

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

6.2. Software Installation

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

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

Warning: Before installing the software, for RedHawk kernel, 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 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.

Warning: RedHawk kernel release 8.2.1 onwards has enabled Supervisor Mode Access Protection (SMAP), which is incompatible with driver releases 24.1.0 or earlier. It is possible that even though the kernel has SMAP enabled, some platforms may not support it. If you issue the command '*lspcu* **|** *grep smap*' and it shows '*smap'* as enabled, then you will need to add the '*nosmap'* argument to the grub entry and reboot the kernel.

To install the ***ccrtaicc*** 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 **ccrtaicc_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/ccrtaicc_driver
      --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/ccrtaicc_driver

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

# cd /
# eject
```

On successful installation the source tree for the ***ccrtaicc*** package, including the loadable kernel module, API libraries, and test programs is extracted into the **/usr/local/CCRT/drivers/ccrtaicc** directory by the rpm installation process, which will then compile and install the various software components.

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

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

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

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

```
# cat /proc/ccrtaicc

Version            : 2022.1.0
```

```
Built              : Thu Jan 20 08:44:36 EST 2022
Boards             : 1
  card=0: [03:00.0] bus=3, slot=0, func=0, irq=59, msi=1, BInfo=9350.01.01
  FM=04/03/2019 (2.0) FLV=00000000 FWB=00000000 ID=687377 MC=C7 RLS=150
(AICC)
```

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

```
CHRDEV "ccrtaicc" major number 233 goes below the dynamic allocation range
```

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

In addition to the above message, on some systems you may also see messages from APEI *(ACPI Platform Error Interface)* or AER *(Advanced Error Reporting)* which have these error reporting capabilities. These messages will be of the form of unrecoverable hardware errors or some other form of hardware errors for the board when the driver/firmware is loaded and started. This is because during the driver load operation, a fresh copy of the firmware is installed and started. This process of starting is equivalent to issuing a power shutdown and restart of the card. Some operating systems see the device as being no longer present, and generate the message.

On RedHawk 8.x kernels, you may see cautionary messages on the system console or via *dmesg* command similar to the following when the **ccrtaicc** driver is loaded, as this is a proprietary driver:

ccrtaicc: module verification failed: signature and/or required key missing - tainting kernel

6.3. Software Removal

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

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

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

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

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

```
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl stop ccrtaicc
      --- or on Ubuntu RedHawk ---
# /bin/systemctl stop ccrtaicc
```

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

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

NOTE: On some Debian RedHawk systems, the following message may appear and can be ignored when the package is removed. *"dpkg: warning: while removing ccrtaicc, directory '/usr/local' not empty so not removed"*.

7. Auto-loading the Driver

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

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

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccrtaicc
# make test                          (build the test programs)
# ./test/ccrtaicc_disp               (display board registers)
# ./test/ccrtaicc_dump               (dump all board resisters)
# ./test/ccrtaicc_rdreg              (display board resisters)
# ./test/ccrtaicc_reg                (Display board resisters)
# ./test/ccrtaicc_regedit            (Interactive board register editor test)
```

```
# ./test/ccrtaicc_tst                  (Interactive test to test driver and
                                        board)
# ./test/ccrtaicc_wreg                 (edit board resisters)

# ./test/Flash/ccrtaicc_flash          (Flash: Flash FPGA)
# ./test/Flash/ccrtaicc_label          (Flash: Label FPGA)
# ./test/Flash/ccrtaicc_dump_license   (Flash: Dump License)

# ./test/lib/ccrtaicc_adc              (library: test ADC channel registers)
# ./test/lib/ccrtaicc_adc_calibrate    (library: test ADC calibrate)
# ./test/lib/ccrtaicc_adc_fifo         (library: test ADC FIFO channels)
# ./test/lib/ccrtaicc_adc_sps          (library: test ADC SPS for channels)
# ./test/lib/ccrtaicc_check_bus        (library: test system jitter)
# ./test/lib/ccrtaicc_clock            (library: test clock)
# ./test/lib/ccrtaicc_disp             (library: display board registers)
# ./test/lib/ccrtaicc_dma              (library: run dma test)
# ./test/lib/ccrtaicc_example          (library: run example test)
# ./test/lib/ccrtaicc_expires          (library: run expires information test)
# ./test/lib/ccrtaicc_identify         (library: identify cards in the system)
# ./test/lib/ccrtaicc_info             (library: provide information of all boards)
# ./test/lib/ccrtaicc_msgdma           (library: modular scatter-gather DMA test)
# ./test/lib/ccrtaicc_msgdma_clone     (library: modular scatter-gather DMA cloning
                                                 test)
# ./test/lib/ccrtaicc_msgdma_info      (library: modular scatter-gather DMA info)
# ./test/lib/ccrtaicc_msgdma_multi_clone
                                       (library: modular scatter-gather DMA
                                                 multi-cloning test)
# ./test/lib/ccrtaicc_smp_affinity     (library: display/set IRQ CPU affinity)
# ./test/lib/ccrtaicc_transfer         (library: run DMA and PIO transfer test)
# ./test/lib/ccrtaicc_tst_lib          (library: Interactive test to test driver
                                                 and board)
```

9. Re-building the Driver, Library and Tests

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

Warning: Before installing the software, for Redhawk kernels, the build environment **must** be set up and match the current OS kernel you are using. If you are running one of the preconfigured kernels supplied by Concurrent 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/ccrtaicc
# 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/ccrtaicc
# 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 CCRT AICC board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccrtaicc
# 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 PCIe 64-Channel Analog Input I/O Card (AICC) 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 ***ccrtaicc_devs*** will be created in the ***/usr/local/CCRT/drivers/ccrtaicc/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccrtaicc*** 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 ***ccrtaicc_devs*** file is automatically generated every time the user issues the ***'make load'*** or ***'/sbin/service ccrtaicc start' (on RedHawk 6.5 and below)*** or ***'/usr/bin/systemctl start ccrtaicc' (on RedHawk 7.0 and above)*** command with the current detected cards, information. Any device to card association edited and placed in this file by the user is retained and used during the next ***'make load'*** or ***'/sbin/service ccrtaicc load'*** or ***'/usr/bin/systemctl start ccrtaicc'*** process.

If the user deletes the ***ccrtaicc_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 ***ccrtaicc_devs*** configuration file.

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

10.2. Library Interface

There is an extensive software library that is provided with this package. For more information on the library interface, please refer to the *PCIe 64-Channel Analog Input I/O Card (AICC) Software Interface by Concurrent Real-Time* document.

10.3. Debugging

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

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

```
# EXTRA_CFLAGS += -DCCRTAICC_DEBUG
```

Next, compile and install the driver

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

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

```
# ccrtaicc_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/ccrtaicc/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 "ccrtaicc_debug_mask=0x00082047" > /proc/ccrtaicc
```

Following are the supported flags for the debug mask as shown in the ***ccrtaicc_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 ***ccrtaicc_debug_ctrl*** is also supplied in the ***ccrtaicc_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 "ccrtaicc_debug_ctrl=0x00001234" > /proc/ccrtaicc
```

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

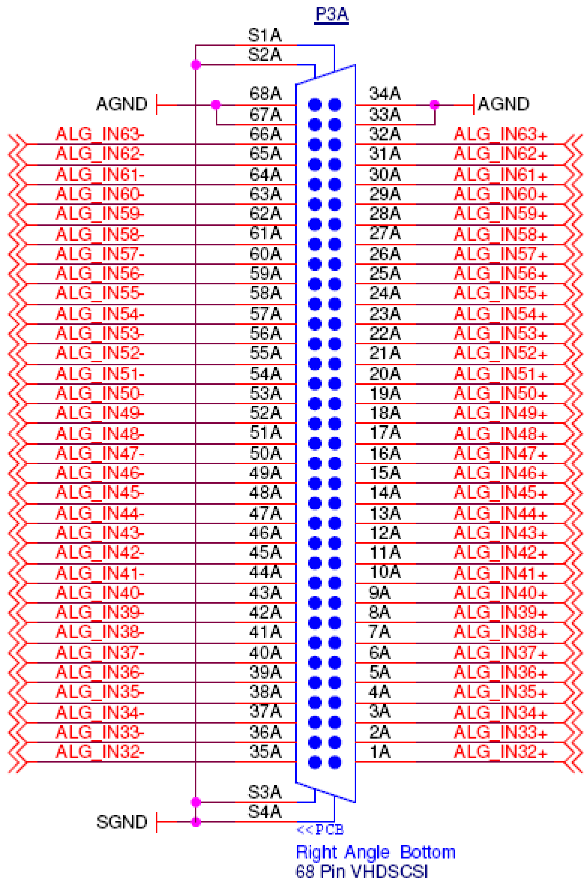
11. Notes and Errata

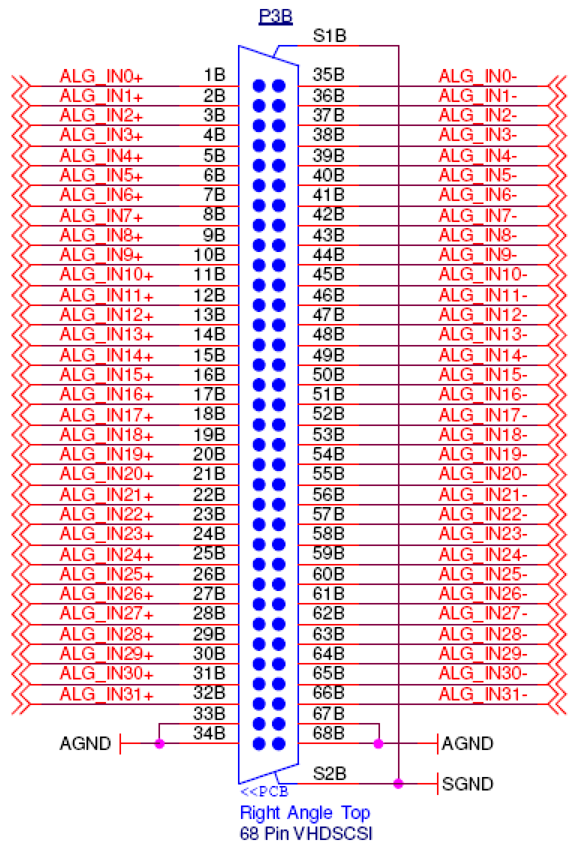
- In some kernel releases, when a package is installed or uninstalled, you may see a warning message on the system console similar to ***"systemd-rc-local-generator[22094]: /etc/rc.d/rc.local is not marked executable, skipping."***. This is for informational purpose only and can be ignored.
- This driver provides support for the new MsgDma Cloning *(Patent-Pending)* feature and enabled after purchase of the appropriate licenses.
- An additional feature of the Cloning support is Region Addressing. The user needs to clear the Physical Memory structure when calling ccrtAICC_MMap_Physical_Memory() as it is looking at the address being passed. The call will fail if the address supplied is invalid.
- Region Addressing allows a user to supply a physical address to access memory beyond its domain. Care should be taken in supplying a valid physical address, otherwise results can be unpredictable including kernel crash or system hang. ***On RH6.0 through RH7.2 kernels, a dmesg 'ioremap' kernel-warning message may be generated when kernel I/O mapping is performed on a user supplied physical memory address that is allocated and reserved by someone other than the user. If this is exactly what the user is intending to do, the message can be ignored as this warning should have no effect on the driver operation or the system. It may appear only once since a system reboot.***
- Only one Cloning or MsgDma operation can be active at a given time. Additionally, it is meaningless to perform Cloning on a FIFO region for two reasons. Firstly, each data in a FIFO is synchronous, however, the Cloned region is accessed asynchronously. Secondly, when the FIFO runs empty *(underflow)* or cannot accept more data *(overflow)* the results are unpredictable as there is no flow control in the Cloning operation.
- 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.
- RedHawk kernel release 8.2.1 onwards has enabled Supervisor Mode Access Protection (SMAP), which is incompatible with driver releases 24.1.0 or earlier. It is possible that even though the kernel has SMAP enabled, some platforms may not support it. If you issue the command '*lscpu* **|** *grep smap*' and it shows '*smap'* as enabled, then you will need to add the '*nosmap'* argument to the grub entry and reboot the kernel.
- Ubuntu kernels RH8.0 onwards may have the default ***systemd-timesyncd*** daemon installed which does not accurately adjust the system clock causing the Sample/Second test to fall out of tolerance and fail. You may want to replace the default with the ***chrony*** package for a more accurate time asjustment.
- On some Debian systems, the following message can be ignored when the package is removed. *"dpkg: warning: while removing ccrtaicc, directory '/usr/local' not empty so not removed"*
- Driver and board support MSI interrupts. It can also be configured for wired interrupts. MSI support is the default.
- On certain systems, the current DMA engine is not fast enough to sustain the maximum throughput of the card when using the internal FIFO. In that case, FIFO overflow will occur. If that happens, you will need to reduce the number of selected FIFO channels and/or reduce the clock speed of the converters until the FIFO overflow condition is resolved.
- Though there are two DMA engines, only DMA0 has full access of the entire board. DMA1 for old firmware is restricted to the section of the board that is above the Diagnostic Ram area. If you use DMA1 engine below that location, the results are unpredictable including but not limited to crashing the kernel. If the API is used, the user will get an error when accessing incorrect regions using the DMA1 engine.
- For new firmware, DMA0 and DMA1 engines are identical.
- The old firmware does not support modular scatter-gather DMA (MsgDma). The new firmware does support MsgDma.
- This card does not support Serial Prom.
- This card does not support SDRAM.
- In the old firmware, the *ADC_FifoData* is the location pointed to the FIFO Data area. However, for new firmware onwards, the *ADC_NextGen_FifoData* location holds the new FIFO location. The *ADC_FifoData* is no longer available and is unused.

- 500KSPS capture for all 64 channels, or 700KSPS for 32 channels is only possible using the modular scatter-gather DMA.
- To achieve 1000KSPS, the card will need to have one ADC set to 500KSPS and another to the inverted 500KSPS clock. In this way, when the same input is connected to two channels on the two different ADC, the sampling will occur at twice the speed for the input signal. With proper merging of the paired channels, 1000KSPS capture will be achieved. A maximum of 32 channels can therefore be setup to capture at 1000KSPS.
- If the two ADCs are configured for high-speed selection, then each ADC can sample at a rate up to 700KSPS for a total of 32 channels. 1400KSPS can be achieved by setting one ADC at the maximum 700KSPS and the other with an inverted clock of 700KSPS. Once agin, when the same input is connected to two channels on the two different ADc, the sampling will occur at twice the speed for the input signal. With proper merging of the paired channels, 1400KSPS capture will be achieved. A maximum of 16 channels can therefore be setup to capture at 1400KSPS.

Appendix A: External Connections and Pin-outs

The front panel I/O connectors are industry standard 68-pin VHD SCSI type connectors with the following pin-out when looking at the board:

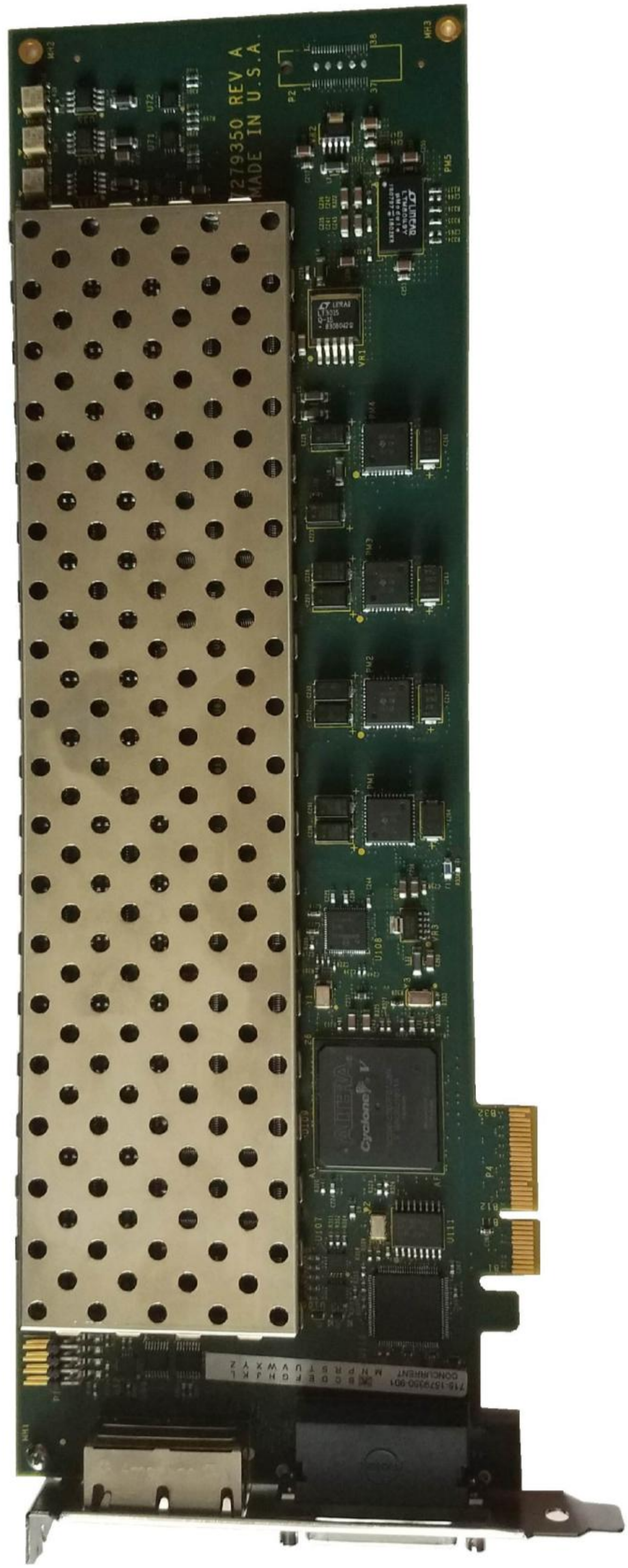




External Connector Notes

1. An analog ground connection is required for the ESD and over/under voltage protection circuits to function correctly for the analog signals.
2. Connector J1 located at the top of the front panel is used for multi-board synchronization. CAT5 capable or greater shielded cable should be used. The top position of J1 is the output from the Master board. The bottom position of J1 is the input to the Slave board.
3. All other connectors on the board are used for manufacturing test and should not have anything attached to them.

Appendix B: The 64-Channel Analog Input FPGA Board



This page intentionally left blank