

Release Notes

CCURPMFC (WC-CP-FIO)



<i>Driver</i>	ccurpmfc (WC-CP-FIO)	
<i>OS</i>	RedHawk (CentOS or Ubuntu based)	
<i>Vendor</i>	Concurrent Real-Time	
<i>Hardware</i>	PCIe Programmable Multi-Function I/O Card (CP-FPGA-Ax)	
<i>Date</i>	March 18 th , 2020	Rev 2020.1



This page intentionally left blank

Table of Contents

1. INTRODUCTION	1
2. REQUIREMENTS	2
3. DOCUMENTATION	2
4. INSTALLATION AND REMOVAL	2
4.1. Hardware Installation	2
4.2. Software Installation	3
4.3. Software Removal	5
5. AUTO-LOADING THE DRIVER	5
6. TESTING AND USAGE	5
7. RE-BUILDING THE DRIVER, LIBRARY AND TESTS.....	6
8. SOFTWARE SUPPORT.....	7
8.1. Device Configuration	7
8.2. Library Interface.....	8
8.3. Debugging	8
9. 256K BASE ADDRESS REGISTER (BAR) FIRMWARE SUPPORT.....	9
9.1. Determining BAR of current installed Firmware.....	9
9.2. Installing the new 256K BAR Base Level Firmware.....	10
10. NOTES AND ERRATA.....	11
APPENDIX A: EXTERNAL CONNECTIONS AND PIN-OUTS	12
APPENDIX B: THE MULTI-FUNCTION FPGA BOARD.....	13

This page intentionally left blank

1. Introduction

This document assists the user in installing the CCUR-PCIe-PMFC Linux **ccurpmfc** driver and related software on the RedHawk OS for use with the CCUR-PCIe-Multi-Function I/O Card (**MIOC**). 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 **ccurpmfc** man page.

The MIOC is a programmable multi-function card with a PCI express interface.

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

❖ General

- Altera Arria V FPGA Control
- 1GB SDRAM
- 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
- Isolated I/O Power
- Low Noise Analog Power Generation
- In System Calibration
- Non-volatile Storage of Calibration Data
- NIST Traceable Calibration Standard
- Directly Addressable Conversion Data Registers
- 128K Word Conversion Data FIFO's with DMA
- Industry Standard High Density SCSI 68-pin Connectors
- RJ-45 Synchronization Connectors

❖ Analog Input Section

- 16-channel 16-bit Digital-to-Analog Conversion
- Differential or Single-ended Input
- +/-5V or +/-10V Input Range
- Input Impedance >1Meg ohm
- Input Over-voltage Protection +/-30V
- 300Khz Maximum Sampling Rate

❖ Analog Output Section

- 16-channel 16-bit Digital-to-Analog Conversion
- 16-channel Single-ended Output or up to 8-channel Differential outputs.
- Differential channels are consecutive even/odd pairs
- 0 to +10V, +/-5V or +/-10V Single-ended Output Range
- +/-10V or +/-20V Differential Output Range
- 10 Milliamp Maximum Output Drive

- 100K Maximum Updates Per Second
- ❖ Digital I/O Section
 - 96-channel Input/Output
 - 5V TTL Levels (3.3V Optional)
 - 4 Milliamp Sink/Source
 - 25Mhz Maximum I/O Rate
 - Outputs Selectable per Nibble
 - Input Channel Snapshot
 - Output Channel Synchronization
 - Change-of-state detection
- ❖ Intellectual Property Core (IpCore) Section (*Special Cards Required*)
 - 64-channel Change of State IP Core Card

2. Requirements

- CCUR-MIOC 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 Programmable Multi-Function I/O Card (PMFC) Software Interface by Concurrent Real-Time.

4. Installation and Removal

4.1. Hardware Installation

The CCUR-MIOC 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.



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 ***ccurpmfc*** 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 -d1542:9290
```

05:00.0 System peripheral: Concurrent Real-Time FPGA Card (rev 01)

Subsystem: Concurrent Real-Time Device 0100

Physical Slot: 3

Flags: bus master, fast devsel, latency 0, IRQ 59

Memory at bd340000 (32-bit, non-prefetchable) [size=32K]

Memory at bd300000 (32-bit, non-prefetchable) [size=256K]
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
```

66 87:00.0 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 1542:9290
```

For each CCUR-MIOC 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:

87:00.0 Unclassified device [0008]: Concurrent Real-Time Device 9290 (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 `ccurpmfc` software is distributed in RPM format for CentOS and DEB format for Ubuntu OS on a CD-ROM. Source for the API library and kernel loadable driver are not included, however, source for example test programs as well as documentation is provided in PDF format.

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

```
# rpm -ivh ccurpmfc_RedHawk_driver*.rpm (on a CentOS based system)
--or--
# dpkg -i ccurpmfc_RedHawk_driver*.deb (on an Ubuntu based system)

# cd /
# eject
```

On successful installation the source tree for the **ccurpmfc** package, including the loadable kernel module, API libraries, and test programs is extracted into the **/usr/local/CCRT/drivers/ccurpmfc** 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/ccurpmfc
# make load
    --- or on RedHawk 6.5 and below ---
# /sbin/service ccurpmfc start
    --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl start ccurpmfc
```

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

```
# cat /proc/ccurpmfc

Version          : 25.0.1
Built            : Mon Mar 16 14:55:28 EDT 2020
Boards           : 1
  card=0: [06:00.0] bus=6, slot=0, func=0, irq=59, msi=1, BInfo=9290.02.01
                  FM=06/19/2019 (4.0) FLV=00000000 FWB=00000000 IP=0 ID=674459
                  MC=B3 RLS=100 (MultiFunc)
```

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

```
CHRDEV "ccurpmfc" 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 **ccurpmfc** driver is loaded, as this is a proprietary driver:

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


4.3. Software Removal

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



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

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

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

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

5. Auto-loading the Driver

The **ccurpmfc** 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/systemd/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 ccurpmfc (enable auto-loading of the driver)
# /sbin/chkconfig --del ccurpmfc (disable auto-loading of the driver)
--- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl enable ccurpmfc (enable auto-loading of the driver)
# /usr/bin/systemctl disable ccurpmfc (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/ccurpmfc
```

```
# make test (build the test programs)
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurpmfc
# make test (build the test programs)
# ./test/ccurpmfc_disp (display board registers)
# ./test/ccurpmfc_dump (dump all board registers)
# ./test/ccurpmfc_rdreg (display board registers)
# ./test/ccurpmfc_reg (Display board registers)
# ./test/ccurpmfc_regedit (Interactive board register editor test)
# ./test/ccurpmfc_tst (Interactive test to test driver and board)
# ./test/ccurpmfc_wreg (edit board registers)

# ./test/Flash/ccurpmfc_flash (Flash: Flash FPGA)
# ./test/Flash/ccurpmfc_label (Flash: Label FPGA)
# ./test/Flash/ccurpmfc_dump_license (Flash: Dump License)

# ./test/lib/ccurpmfc_adc (library: test ADC channel registers)
# ./test/lib/ccurpmfc_adc_calibrate (library: test ADC calibrate)
# ./test/lib/ccurpmfc_adc_fifo (library: test ADC FIFO channels)
# ./test/lib/ccurpmfc_adc_sps (library: test ADC samples/channel)
# ./test/lib/ccurpmfc_check_bus (library: test system jitter)
# ./test/lib/ccurpmfc_clock (library: test clock)
# ./test/lib/ccurpmfc_dac (library: test DAC channels)
# ./test/lib/ccurpmfc_dac_calibrate (library: test DAC calibrate)
# ./test/lib/ccurpmfc_dac_setchan (library: test DAC channels)
# ./test/lib/ccurpmfc_dio (library: test DIO channels)
# ./test/lib/ccurpmfc_dio_intr (library: test DIO change-of-state interrupt)
# ./test/lib/ccurpmfc_disp (library: display board registers)
# ./test/lib/ccurpmfc_dma (library: run dma test)
# ./test/lib/ccurpmfc_example (library: run example test)
# ./test/lib/ccurpmfc_expires (library: run expires information test)
# ./test/lib/ccurpmfc_identify (library: identify cards in the system)
# ./test/lib/ccurpmfc_info (library: provide information of all boards)
# ./test/lib/ccurpmfc_msgdma (library: modular scatter-gather DMA test)
# ./test/lib/ccurpmfc_msgdma_info (library: modular scatter-gather DMA info)
# ./test/lib/ccurpmfc_smp_affinity (library: display/set IRQ CPU affinity)
# ./test/lib/ccurpmfc_transfer (library: run DMA and PIO transfer test)
# ./test/lib/ccurpmfc_tst_lib (library: Interactive test to test driver and board)

# ./test/lib/IpCore/ccurpmfc_ipcore_cos (library:IpCore: Change-of-state test)

# ./test/lib/Sprom/ccurpmfc_sprom (library:Sprom: run SPROM test)
```

7. 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, 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 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/ccurpmfc
# 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/ccurpmfc
# make install      (install the driver software, library and man page)
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurpmfc
# 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 PCIe Programmable Multi-Function I/O Card (MIOC) 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 ***ccurpmfc_devs*** will be created in the ***/usr/local/CCRT/drivers/ccurpmfc/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccurpmfc*** 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 ***ccurpmfc_devs*** file is automatically generated every time the user issues the ***'make load'*** or ***'/sbin/service ccurpmfc start'*** (on RedHawk 6.5 and below) or ***'/usr/bin/systemctl start ccurpmfc'*** (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 ccurpmfc load'*** or ***'/usr/bin/systemctl start ccurpmfc'*** process.

If the user deletes the ***ccurpmfc_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 ***ccurpmfc_devs*** configuration file.



Warning: If you edit the ***ccurpmfc_devs*** file to associate a device to a card, you will need to re-issue the ***'make load'*** or ***'/sbin/service ccurpmfc start'*** or ***'/usr/bin/systemctl start ccurpmfc'*** 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. 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 Programmable Multi-Function I/O Card (PMFC) Software Interface by Concurrent Real-Time* document.

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

```
# EXTRA_CFLAGS += -DCCURPMFC_DEBUG
```

Next, compile and install the driver

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

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

```
# ccurpmfc_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/ccurpmfc/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 "ccurpmfc_debug_mask=0x00082047" > /proc/ccurpmfc
```

Following are the supported flags for the debug mask as shown in the **ccurpmfc_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_RUNTIME       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_DBUF          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 `ccurpmfc_debug_ctrl` is also supplied in the `ccurpmfc_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 "ccurpmfc_debug_ctrl=0x00001234" > /proc/ccurpmfc
```

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

9. 256K Base Address Register (BAR) Firmware Support

The original FPGA firmware came with 128K BAR memory support. The new FPGA firmware uses a 256K BAR memory size. For cards that have multi-level firmware (Base & Run Level Firmware), the user will be required to upgrade the Base Level Firmware to the 256K BAR Base Level Firmware if they plan to use the 256K BAR Run Level Firmware. Failure to do so will result in unpredictable behavior.

For now, there is no need to update the Multi-Function Firmware (*normally on A5 Member Code cards*) as no changes have been made to it to utilize the 256K memory size. This exercise is more for the *B3* and *B7* cards that have FPGAWB support.

The user can upgrade to the 256K BAR base level firmware at any time (even if the Run Level Firmware currently on the card is a 128K BAR) with the help of the `ccurpmfc_flash` utility.

9.1. Determining BAR of current installed Firmware

The easiest way to determine the BAR of the firmware is by using the `lspci(8)` system command. The FPGA card must be installed, however, there is no need to install the FPGA driver.

```
# lspci -v -d1542:9290
```

```
05:00.0 System peripheral: Concurrent Real-Time FPGA Card (rev 01)
```

```
Subsystem: Concurrent Real-Time Device 0100
```

```
Physical Slot: 3
```

```
Flags: bus master, fast devsel, latency 0, IRQ 59
```

```
Memory at bd340000 (32-bit, non-prefetchable) [size=32K]
```

```
Memory at bd300000 (32-bit, non-prefetchable) [size=256K]
```

```
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
```

You will notice that the second memory address has a size of **256K**. If the card shows a size of **128K** instead, then you will need to update the Base Level Firmware if you plan to use the new 256K BAR Run Level Firmware.

9.2. Installing the new 256K BAR Base Level Firmware

Before updating the Base Level Firmware, you need to make sure whether the card has a Member Code of A5, B3 or B7. In order to determine the card Member Code, the driver and card must be installed and running in the system. You can then issue the following command to get information on the card.

```
# cat /proc/ccurpmfc

Version          : 24.0.1
Built           : Tue Jun 18 11:18:12 EDT 2019
Boards          : 1
  card=0: [05:00.0] bus=5, slot=0, func=0, irq=59, msi=1, BInfo=9290.02.01
  FM=10/14/2016 (3.2) FLV=00000000 FWB=00000000 IP=0 ID=674459 MC=B3 RLS=100
  (MultiFunc)
```

In the above example, the card has a Member Code of 'B3'. We therefore will need to install a 256K BAR Base Level Firmware with a Member Code of 'B3' on this card. ***Installing the wrong Member Code will render the board useless.***

```
# cd /usr/local/CCRT/drivers/ccurpmfc/test/Flash
# ./ccurpmfc_flash -b# -w BASE/XX/*.cust (# is the board number, XX is A3, B5 or B7 depending
                                         on the member code of the card)
# <Follow the instructions until completion>
```

Once the firmware is successfully burnt, you will need to reboot the system for the Base Level Firmware to take effect.

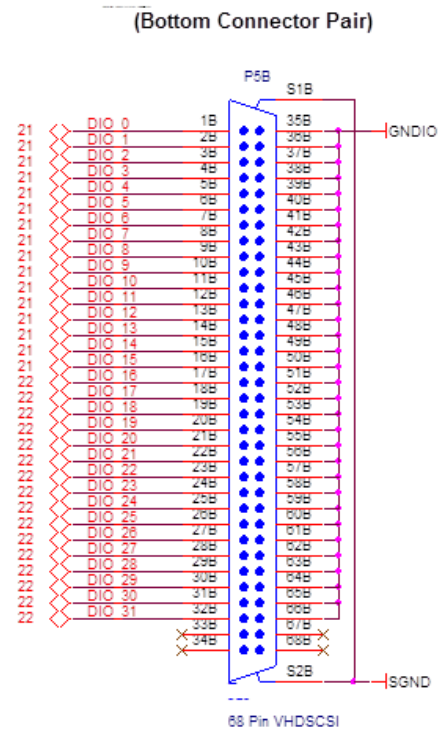
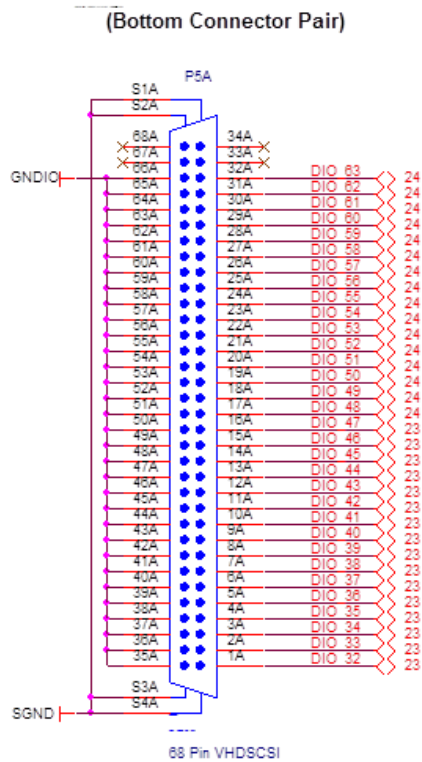
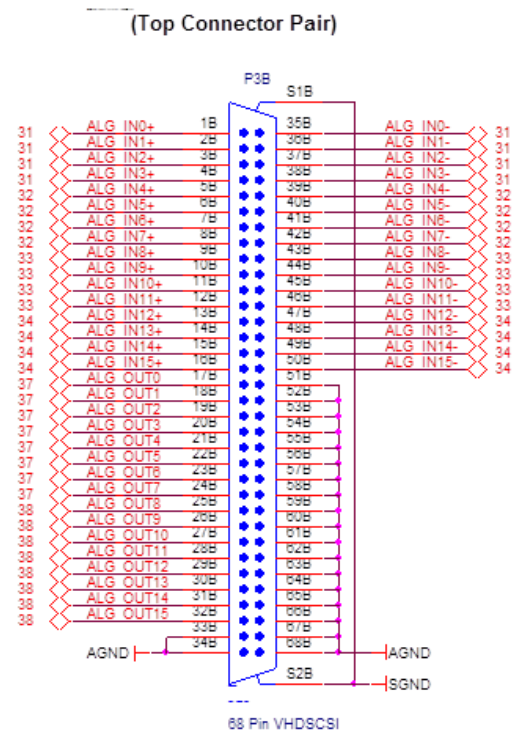
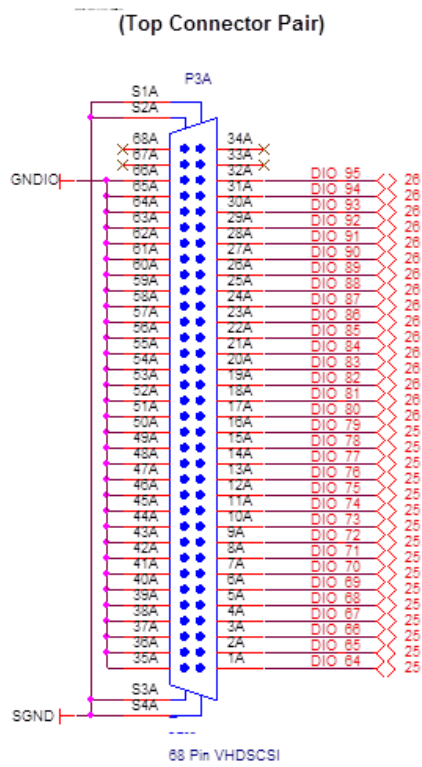
```
# reboot
```

Once the system is operational, use the ***lspci(8)*** as above to verify that the memory BAR is now showing **256K**. If you have a **256K** Run Level Firmware, you can install that at this time.

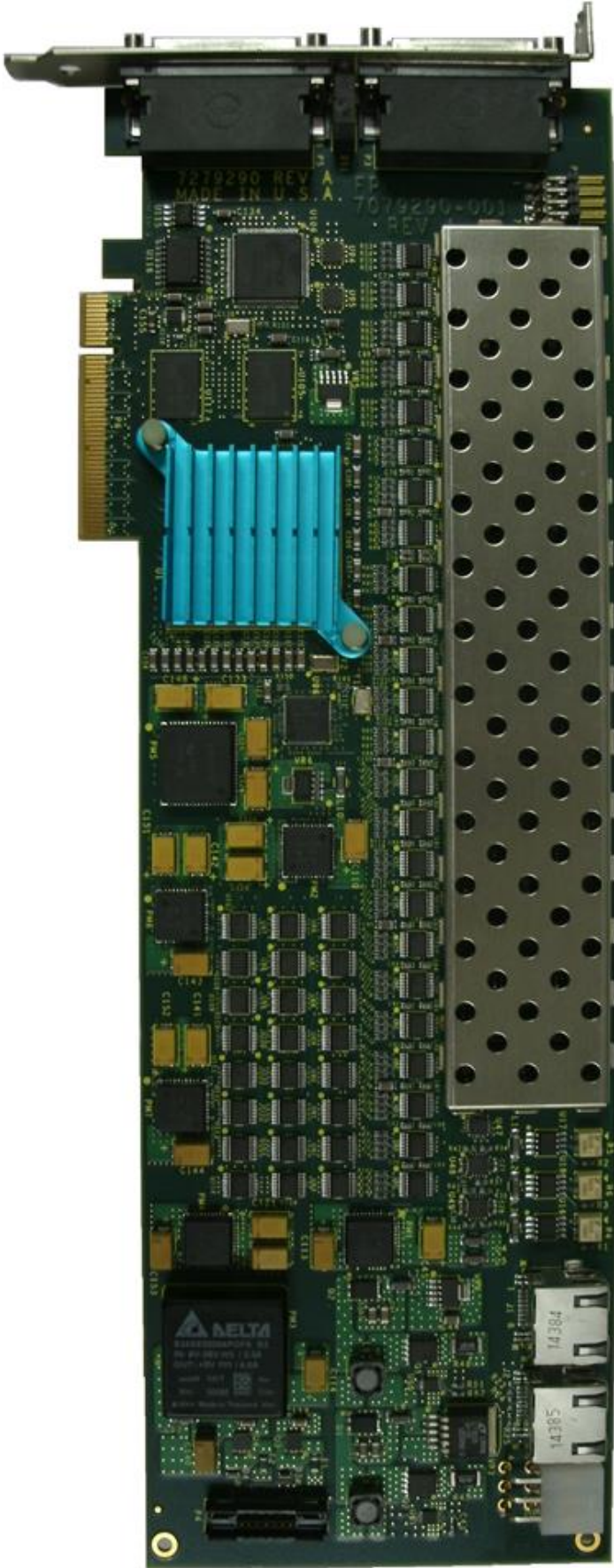
10. Notes and Errata

- This new driver provides support for the 128K and 256K board BAR sizes. If a 256K run level firmware is installed, then the 256K base level firmware must also be installed on the card otherwise, the board will not operate properly.
- Driver and board support MSI interrupts. It can be configured for wired interrupts. MSI support is the default.
- Currently, several totally different types of cards are supported by the driver. The standard Multi-Function I/O Card (*MIOC*), the specialized Engine Control, the IP Core change-of-state card and several others. Currently, all tests are tailored for the Multi-Function card so some tests may fail on the other card.
- It is possible that *lspci* calls may still display the device with the old name of “**Concurrent Computer Corporation**” instead of “**Concurrent Real-Time**” if the OS has not been updated.
- Modular scatter-gather DMA is currently not supported on the Multi-Function I/O Card. It is supported in some specific cards with limitations on specific memory regions.
- Removed Deprecated Defines
- The following defines have been renamed:
 - CCURPMFC_ADC_DATA_UPDATE_CLOCK_NONE -> CCURPMFC_ADC_UPDATE_CLOCK_NONE_MASK
 - CCURPMFC_ADC_DATA_EXTERNAL_SIGNAL -> CCURPMFC_ADC_EXTERNAL_SIGNAL_MASK
 - CCURPMFC_ADC_DATA_CALIBRATION_BUS -> CCURPMFC_ADC_CALIBRATION_BUS_MASK
 - CCURPMFC_ADC_DATA_FORMAT_OFFSET_BINARY -> CCURPMFC_ADC_OFFSET_BINARY_MASK
 - CCURPMFC_ADC_DATA_FORMAT_TWOS_COMPLEMENT -> CCURPMFC_ADC_TWOS_COMPLEMENT_MASK
 - CCURPMFC_ADC_INPUT_RANGE_BIPOLAR_10V -> CCURPMFC_ADC_BIPOLAR_10V_MASK
 - CCURPMFC_ADC_INPUT_RANGE_BIPOLAR_5V -> CCURPMFC_ADC_BIPOLAR_5V_MASK
 - CCURPMFC_DAC_UPDATE_MODE_IMMEDIATE -> CCURPMFC_DAC_MODE_IMMEDIATE_MASK
 - CCURPMFC_DAC_UPDATE_MODE_SYNCHRONIZED -> CCURPMFC_DAC_MODE_SYNCHRONIZED_MASK
 - CCURPMFC_DAC_DATA_FORMAT_OFFSET_BINARY -> CCURPMFC_DAC_OFFSET_BINARY_MASK
 - CCURPMFC_DAC_DATA_FORMAT_TWOS_COMPLEMENT -> CCURPMFC_DAC_TWOS_COMPLEMENT_MASK
 - CCURPMFC_DAC_OUTPUT_SINGLE_ENDED -> CCURPMFC_DAC_SINGLE_ENDED_MASK
 - CCURPMFC_DAC_OUTPUT_DIFFERENTIAL -> CCURPMFC_DAC_DIFFERENTIAL_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_SINGLE_ENDED_UNIPOLAR_10V -> CCURPMFC_DAC_SINGLE_ENDED_UNIPOLAR_10V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_SINGLE_ENDED_BIPOLAR_5V -> CCURPMFC_DAC_SINGLE_ENDED_BIPOLAR_5V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_SINGLE_ENDED_BIPOLAR_10V -> CCURPMFC_DAC_SINGLE_ENDED_BIPOLAR_10V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_SINGLE_ENDED_UNIPOLAR_20V -> CCURPMFC_DAC_SINGLE_ENDED_UNIPOLAR_20V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_DIFFERENTIAL_UNIPOLAR_10V -> CCURPMFC_DAC_DIFFERENTIAL_UNIPOLAR_10V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_DIFFERENTIAL_BIPOLAR_10V -> CCURPMFC_DAC_DIFFERENTIAL_BIPOLAR_10V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_DIFFERENTIAL_BIPOLAR_20V -> CCURPMFC_DAC_DIFFERENTIAL_BIPOLAR_20V_MASK
 - CCURPMFC_DAC_OUTPUT_RANGE_DIFFERENTIAL_UNIPOLAR_20V -> CCURPMFC_DAC_DIFFERENTIAL_UNIPOLAR_20V_MASK

Appendix A: External Connections and Pin-outs



Appendix B: The Multi-Function FPGA Board



This page intentionally left blank