ADC Software Interface
CCRTNGFC (WC-CP-FIO2)

PCIe Next Generation FPGA I/O Card (NGFC)

Driver	ccrtngfc (WC-CP-FIO2)	
Platform	RedHawk Linux® *(CentOS/Rocky/RHEL & Ubuntu)*, Native Ubuntu® and Native Red Hat Enterprise Linux®[1]	
Vendor	Concurrent Real-Time	
Hardware	PCIe Programmable Multi-Function Card (CP-FPGA-4 & 5)	
Author	Darius Dubash	
Date	February 10th, 2026	Rev 2026.1

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

This page intentionally left blank

This page intentionally left blank

1. Introduction

This document provides the software interface to the ***ccrtngfc*** driver ADC Daughter Card which communicates with the Concurrent Real-Time PCI Express Next Generation FPGA I/O Card (NGFC). This ADC daughter card is an optional interface that needs to be physically installed in one of two FMC slots located on the mother board. For additional information for low-level programming is contained in the *Concurrent Real-Time PCIe Next Generation FPGA I/O Cards (NGFC) Design Specification* (No. *0610111*) document.

The software package that accompanies this board provides the ability for advanced users to communicate directly with the board via the driver *ioctl(2)* and *mmap(2)* system calls. When programming in this mode, the user needs to be intimately familiar with both the hardware and the register programming interface to the board. Failure to adhere to correct programming will result in unpredictable behavior.

Additionally, the software package is accompanied with an extensive set of application programming interface (API) calls that allow the user to access all capabilities of the board. The API library also allows the user the ability to communicate directly with the board through the *ioctl(2)* and *mmap(2)* system calls. In this case, there is a risk of this direct access conflicting with API calls and therefore should only be used by advanced users who are intimately familiar with the hardware, board registers and the driver code.

Various example tests have been provided in the *test* and *test/lib* directories to assist the user in developing their applications.

1.1 Related Documents

- PCIe Next Generaton FPGA Driver Installation on RedHawk Release Notes by Concurrent Real-Time.
- PCIe Next Generation FPGA Driver Technical Guide by Concurrent Real-Time.
- PCIe Next Generation FPGA Card I/O (NGFC) Design Specification (No. *0610111*) by Concurrent Real-Time.

2. Software Support

2.1 Application Program Interface (API) Access

Before any of the ADC Daughter Card APIs can be invoked, the user first needs to issue the *ccrtNFGC_DC_ADC_Open()* call and supply the board handle that was returned with the *ccrtNGFC_Open()* call. Once the ADC daughter card is successfully opened, the call will return an ADC Handle which will be supplied as the first argument to all the remaining ADC APIs.

There are a lot of APIs that have multiple arguments to set various parameters. If the user only wishes to change certain parameters for the call, they need to get the current settings via a query API, change only those parameters that need to be modified and then invoke a setting API to update these parameters *(i.e. read/modify/write)*. This is a two API call operation.

A nice feature has been implemented in these APIs to simplify the user programming by having a common parameter CCRTNGFC_DO_NOT_CHANGE which is a #define, that can be used for a lot of these calls. Arguments with this parameter will therefore cause the API to perform the read/modify/write operation instead of the user performing the same function with two API calls. The drawback to this approach is that some compilers will complain about the use of this parameter and therefore the user will require appropriate casting to get rid of warnings/errors.

The following section describes the calls that are available to access this daughter card.

2.1.1 ccrtNGFC_DC_ADC_Activate()

This call must be the first call to activate the ADC. Without activation, all other calls to the ADC will fail. The user can also use this call to return the current state of the ADC without any change by specifying a pointer to *current_state* and setting *activate* to *CCRTNGFC_ADC_ALL_ENABLE_DO_NOT_CHANGE.* If the ADC is already active and the user issues a *CCRTNGFC_ADC_ALL_ENABLE*, no additional activation will be performed.

To cause the ADC to go through a full reset, the user needs to issue the *CCRTNGFC_ADC_ALL_RESET* which will cause the ADC to disable and then re-enable, setting all its ADC values to a default state. ADC calibration data will also be reset.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Activate (void                        *AdcHandle,
                             _ccrtngfc_adc_all_enable_t  activate,
                             _ccrtngfc_adc_all_enable_t  *current_state)

   Description: Activate/DeActivate Daughter Card ADC module

   Input:   void                                  *AdcHandle      (ADC Handle pointer)
            _ccrtngfc_adc_all_enable_t            activate        (activate/deactivate)
               # CCRTNGFC_ADC_ALL_DISABLE
               # CCRTNGFC_ADC_ALL_ENABLE
               # CCRTNGFC_ADC_ALL_RESET           (disable followed by enable)
               # CCRTNGFC_ADC_ALL_ENABLE_DO_NOT_CHANGE
   Output:  _ccrtngfc_adc_all_enable_t            *current_state  (active/deactive)
               # CCRTNGFC_ADC_ALL_DISABLE
               # CCRTNGFC_ADC_ALL_ENABLE
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR            (successful)
               # CCRTNGFC_LIB_BAD_HANDLE          (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN            (device not open)
               # CCRTNGFC_LIB_INVALID_ARG         (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION     (local region not present)
 *****************************************************************************/
```

2.1.2 ccrtNGFC_DC_ADC_Clear_Interrupt_Status()

This call clears the interrupt status for the ADC

```
/*****************************************************************************
  _ccrtngfc_lib_error_number_t
  ccrtNGFC_DC_ADC_Clear_Interrupt_Status (void                     *AdcHandle,
                                          _ccrtngfc_intsta_adc_t  adc_fifo_intr)

   Description: Clear Daughter Card ADC Interrupt Status

   Input:   void                     *AdcHandle    (ADC Handle pointer)
            _ccrtngfc_intsta_adc_t   adc_fifo_intr (pointer to ADC FIFO interrupt status)
                # CCRTNGFC_INT_ADC_FIFO_THRESHOLD_NONE
                # CCRTNGFC_INT_ADC_FIFO_THRESHOLD_RESET
                # CCRTNGFC_INT_ADC_FIFO_THRESHOLD_DO_NOT_CHANGE
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
                # CCRTNGFC_LIB_NO_ERROR            (successful)
                # CCRTNGFC_LIB_BAD_HANDLE          (no/bad handler supplied)
                # CCRTNGFC_LIB_NOT_OPEN            (device not open)
                # CCRTNGFC_LIB_NO_LOCAL_REGION     (local region error)
                # CCRTNGFC_LIB_INVALID_ARG         (invalid argument)
 *****************************************************************************/
```

2.1.3 ccrtNGFC_DC_ADC_Close()

This call is to be used when you no longer want to access the ADC daughter card that was opened with the *ccrtNGFC_DC_ADC_Open()* call.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
```

```
   ccrtNGFC_DC_ADC_Close (void *AdcHandle)

   Description: Close a Daughter Card ADC module.

   Input:   void                            *AdcHandle  (ADC Handle pointer)
   Output:  None
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR               (successful)
               # CCRTNGFC_LIB_INVALID_ARG            (invalid argument)
               # CCRTNGFC_LIB_INVALID_MODULE         (invalid module)
 ***************************************************************************/
```

2.1.4 ccrtNGFC_DC_ADC_Get_Calibration_CSR()

This call returns the current calibration control and status register.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Calibration_CSR (void                      *AdcHandle,
                                       ccrtngfc_calibration_csr_t *CalCSR)

   Description: Get Daughter Card ADC Calibration Control and Status Register

   Input:   void                                      *AdcHandle  (ADC Handle pointer)
   Output:  ccrtngfc_calibration_csr_t                *CalCSR     (pointer to calibration CSR)
              _ccrtngfc_calbus_control_t              BusControl  (bus control)
                 # CCRTNGFC_CB_GROUND
                 # CCRTNGFC_CB_POSITIVE_REFERENCE
                 # CCRTNGFC_CB_NEGATIVE_REFERENCE
                 # CCRTNGFC_CB_BUS_OPEN
                 # CCRTNGFC_CB_4V_REFERENCE
                 # CCRTNGFC_CB_PLUS_10V_REFERENCE
                 # CCRTNGFC_CB_MINUS_10V_REFERENCE
                 # CCRTNGFC_CB_DAC_CHANNEL_0
                 # CCRTNGFC_CB_DAC_CHANNEL_1
                 # CCRTNGFC_CB_DAC_CHANNEL_2
                 # CCRTNGFC_CB_DAC_CHANNEL_3
                 # CCRTNGFC_CB_DAC_CHANNEL_4
                 # CCRTNGFC_CB_DAC_CHANNEL_5
                 # CCRTNGFC_CB_DAC_CHANNEL_6
                 # CCRTNGFC_CB_DAC_CHANNEL_7
                 # CCRTNGFC_CB_DAC_CHANNEL_8
                 # CCRTNGFC_CB_DAC_CHANNEL_9
                 # CCRTNGFC_CB_DAC_CHANNEL_10
                 # CCRTNGFC_CB_DAC_CHANNEL_11
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR               (successful)
               # CCRTNGFC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN               (device not open)
               # CCRTNGFC_LIB_INVALID_ARG            (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION        (local region not present)
 ***************************************************************************/
```

2.1.5 ccrtNGFC_DC_ADC_Get_CSR()

This call returns information from the ADC registers for the selected channel group.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_CSR (void                  *AdcHandle,
                            _ccrtngfc_adc_mask_t  adc_mask,
                            _ccrtngfc_adc_csr_t   *adc_csr)
```

```
   Description: Get Daughter Card ADC Control and Status information

   Input:   void                                        *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_mask_t                        adc_mask    (selected ADC mask)
               # CCRTNGFC_ADC_MASK_0_3
               # CCRTNGFC_ADC_MASK_4_7
               # CCRTNGFC_ADC_MASK_8_11
               # CCRTNGFC_ALL_ADC_MASK
   Output:  _ccrtngfc_adc_csr_t                         *adc_csr (pointer to ADC csr)
               _ccrtngfc_adccsr_update_clock_t          adc_update_clock;
                  # CCRTNGFC_ADC_UPDATE_CLOCK_NONE
                  # CCRTNGFC_ADC_UPDATE_CLOCK_0
                  # CCRTNGFC_ADC_UPDATE_CLOCK_1
                  # CCRTNGFC_ADC_UPDATE_CLOCK_2
                  # CCRTNGFC_ADC_UPDATE_CLOCK_3
                  # CCRTNGFC_ADC_UPDATE_CLOCK_4
               _ccrtngfc_adccsr_input_signal_t          adc_input_signal;
                  # CCRTNGFC_ADC_EXTERNAL_SIGNAL
                  # CCRTNGFC_ADC_CALIBRATION_BUS
               _ccrtngfc_adccsr_dataformat_t            adc_data_format;
                  # CCRTNGFC_ADC_OFFSET_BINARY
                  # CCRTNGFC_ADC_TWOS_COMPLEMENT
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                 (successful)
               # CCRTNGFC_LIB_BAD_HANDLE               (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                 (device not open)
               # CCRTNGFC_LIB_INVALID_ARG              (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION          (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE        (ADC is not active)
 ***************************************************************************/
```

2.1.6 ccrtNGFC_DC_ADC_Get_Driver_Read_Mode()

This call returns the current driver ADC read mode. When a *read(2)* system call is issued, it is this mode that determines the type of read being performed by the driver.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Driver_Read_Mode (void                              *AdcHandle,
                                         _ccrtngfc_driver_ADC_read_mode_t  *mode)

   Description: Get current Daughter Card ADC read mode that will be selected by the
                'read()' call

   Input:   void                                *AdcHandle  (ADC Handle pointer)
   Output:  _ccrtngfc_driver_ADC_read_mode_t    *mode       (select ADC read mode)
               # CCRTNGFC_ADC_PIO_CHANNEL
               # CCRTNGFC_ADC_PIO_FIFO
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                 (successful)
               # CCRTNGFC_LIB_BAD_HANDLE               (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                 (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION          (local region not present)
               # CCRTNGFC_LIB_IOCTL_FAILED             (driver ioctl call failed)
               # CCRTNGFC_LIB_INVALID_ARG              (invalid argument)
 ***************************************************************************/
```

2.1.7 ccrtNGFC_DC_ADC_Get_Fifo_Channel_Select()

This call returns the current Fifo Channel selection mask. Only samples for these selected channels are placed in the FIFO during sample collection.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Fifo_Channel_Select (void                          *AdcHandle,
                                            _ccrtngfc_adc_channel_mask_t
                                                          *adc_fifo_channel_select_mask)

   Description: Get Daughter Card ADC Fifo Channel Selection

   Input:   void                          *AdcHandle                     (ADC Handle pointer)
   Output:  _ccrtngfc_adc_channel_mask_t *adc_fifo_channel_select_mask  (channel select mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE (ADC is not active)
 *****************************************************************************/
```

2.1.8 ccrtNGFC_DC_ADC_Get_Fifo_Info()

This call returns ADC FIFO information to the user.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Fifo_Info (void                     *AdcHandle,
                                  ccrtngfc_adc_fifo_info_t *adc_fifo)

   Description: Get Daughter Card ADC FIFO control and Status information

   Input:   void                                *AdcHandle   (AdcHandle pointer)
   Output:  ccrtngfc_adc_fifo_info_t            *adc_fifo    (pointer to ADC fifo struct)
               _ccrtngfc_adc_fifo_reset_t       reset;
                  # CCRTNGFC_ADC_FIFO_ACTIVE
                  # CCRTNGFC_ADC_FIFO_RESET
               _ccrtngfc_adc_fifo_overflow_t    overflow;
                  # CCRTNGFC_ADC_FIFO_NO_OVERFLOW
                  # CCRTNGFC_ADC_FIFO_OVERFLOW
               _ccrtngfc_adc_fifo_underflow_t   underflow;
                  # CCRTNGFC_ADC_FIFO_NO_UNDERFLOW
                  # CCRTNGFC_ADC_FIFO_UNDERFLOW
               _ccrtngfc_adc_fifo_full_t        full;
                  # CCRTNGFC_ADC_FIFO_NOT_FULL
                  # CCRTNGFC_ADC_FIFO_FULL
               _ccrtngfc_adc_fifo_threshold_t   threshold_exceeded;
                  # CCRTNGFC_ADC_FIFO_THRESHOLD_NOT_EXCEEDED
                  # CCRTNGFC_ADC_FIFO_THRESHOLD_EXCEEDED
```

```
                _ccrtngfc_adc_fifo_empty_t      empty;
                   # CCRTNGFC_ADC_FIFO_NOT_EMPTY
                   # CCRTNGFC_ADC_FIFO_EMPTY
                uint                            data_counter;
                uint                            threshold;
                uint                            max_threshold;
                uint                            driver_threshold;
                _ccrtngfc_adc_channel_mask_t    channel_select_mask;
                   # CCRTNGFC_ADC_CHANNEL_MASK_0
                   # CCRTNGFC_ADC_CHANNEL_MASK_1
                   # CCRTNGFC_ADC_CHANNEL_MASK_2
                   # CCRTNGFC_ADC_CHANNEL_MASK_3
                   # CCRTNGFC_ADC_CHANNEL_MASK_4
                   # CCRTNGFC_ADC_CHANNEL_MASK_5
                   # CCRTNGFC_ADC_CHANNEL_MASK_6
                   # CCRTNGFC_ADC_CHANNEL_MASK_7
                   # CCRTNGFC_ADC_CHANNEL_MASK_8
                   # CCRTNGFC_ADC_CHANNEL_MASK_9
                   # CCRTNGFC_ADC_CHANNEL_MASK_10
                   # CCRTNGFC_ADC_CHANNEL_MASK_11
                   # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE (ADC is not active)
 ****************************************************************************/
```

2.1.9 ccrtNGFC_DC_ADC_Get_Fifo_Threshold()

This call returns the ADC Fifo threshold information.

```
/****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Fifo_Threshold (void *AdcHandle,
                                       uint *adc_threshold)

   Description: Get Daughter Card ADC Fifo Threshold

   Input:   void                   *AdcHandle      (ADC Handle pointer)
   Output:  uint                   *adc_threshold  (ADC fifo threshold)
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (device not open)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
 ****************************************************************************/
```

2.1.10 ccrtNGFC_DC_ADC_Get_Info()

This call returns some useful ADC information.

```
/****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Info (void                *AdcHandle,
                             ccrtngfc_adc_info_t *AdcInfo)

   Description: Get Daughter Card ADC Information
```

```
   Input:    void                   *AdcHandle    (ADC Handle pointer)
   Output:   ccrtngfc_adc_info_t    *AdcInfo      (pointer to returned AdcInfo)
               # int                          Flags                // ADC flags
               # _ccrtngfc_adc_module_t       ModuleNumber         // ADC module number
               # ccrtngfc_handle_t            *Handle              // Main Device library Handle
               # ccrtngfc_local_ctrl_data_t   *local_ptr           // Main Local register pointer
               # ccrtngfc_1579321_adc_t       *local_adc_ptr       // ADC Local register pointer
               # u_int32_t                    *local_adc_fifo_ptr  // ADC Local FIFO register
                                                                      pointer
               # int                          AdcFp                // ADC file descriptor pointer
               # char                         AdcDeviceName[25]    // ADC Device Name
               # ccrtngfc_module_info_t       AdcModuleInfo        // ADC Module Information
   Return:   _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR           (successful)
               # CCRTNGFC_LIB_BAD_HANDLE         (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN           (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION    (local region not present)
               # CCRTNGFC_LIB_CANNOT_OPEN_FILE   (cannot open calib. file)
               # CCRTNGFC_LIB_INVALID_ARG        (invalid argument)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE  (ADC is not active)
 ************************************************************************/
```

2.1.11 ccrtNGFC_DC_ADC_Get_Interrupt_Status()

This call returns the ADC interrupt status

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Interrupt_Status (void                   *AdcHandle,
                                         _ccrtngfc_intsta_adc_t *adc_fifo_intr)

   Description: Get Daughter Card ADC Interrupt Status information

   Input:   void                   *AdcHandle      (ADC Handle pointer)
   Output:  _ccrtngfc_intsta_adc_t  *adc_fifo_intr  (pointer to ADC FIFO interrupt status)
               # CCRTNGFC_INT_ADC_FIFO_THRESHOLD_NONE
               # CCRTNGFC_INT_ADC_FIFO_THRESHOLD_OCCURRED
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (device not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region error)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
 ************************************************************************/
```

2.1.12 ccrtNGFC_DC_ADC_Get_Interrupt_Timeout_Seconds()

This call returns the read time out maintained by the driver. It is the time that the read call will wait before it times out. The call could time out because a DMA fails to complete. The device should have been opened in the block mode (*O_NONBLOCK* not set) for reads to wait for the operation to complete.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Interrupt_Timeout_Seconds (void    *AdcHandle,
                                                  int     *timeout_secs)

   Description: Get Daughter Card ADC Interrupt Timeout Seconds

   Input:   void            *AdcHandle           (ADC Handle pointer)
   Output:  int             *timeout_secs        (pointer to int tout secs)
   Return:  _ccrtngfc_lib_error_number_t
```

```
            # CCRTNGFC_LIB_NO_ERROR          (successful)
            # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN          (device not open)
            # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
            # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
            # CCRTNGFC_LIB_IOCTL_FAILED      (ioctl error)
 ***************************************************************************/
```

2.1.13 ccrtNGFC_DC_ADC_Get_Negative_Cal()

This call returns the ADC negative calibration information for all the channels.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Negative_Cal (void                          *AdcHandle,
                                    _ccrtngfc_adc_channel_mask_t   ChanMask,
                                    ccrtngfc_adc_cal_t             *cal)

   Description: Get Daughter Card ADC Negative Calibration Data

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  ccrtngfc_adc_cal_t         *cal         (pointer to board cal)
               uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
               double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (device not open)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
 ***************************************************************************/
```

2.1.14 ccrtNGFC_DC_ADC_Get_Offset_Cal()

This call returns the ADC offset calibration information for all the channels.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Offset_Cal (void                          *AdcHandle,
                                   _ccrtngfc_adc_channel_mask_t   ChanMask,
                                   ccrtngfc_adc_cal_t             *cal)

   Description: Get Daughter Card ADC Offset Calibration Data

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
```

```
            # CCRTNGFC_ADC_CHANNEL_MASK_0
            # CCRTNGFC_ADC_CHANNEL_MASK_1
            # CCRTNGFC_ADC_CHANNEL_MASK_2
            # CCRTNGFC_ADC_CHANNEL_MASK_3
            # CCRTNGFC_ADC_CHANNEL_MASK_4
            # CCRTNGFC_ADC_CHANNEL_MASK_5
            # CCRTNGFC_ADC_CHANNEL_MASK_6
            # CCRTNGFC_ADC_CHANNEL_MASK_7
            # CCRTNGFC_ADC_CHANNEL_MASK_8
            # CCRTNGFC_ADC_CHANNEL_MASK_9
            # CCRTNGFC_ADC_CHANNEL_MASK_10
            # CCRTNGFC_ADC_CHANNEL_MASK_11
            # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  ccrtngfc_adc_cal_t          *cal         (pointer to board cal)
            uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
            double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Return:  _ccrtngfc_lib_error_number_t
            # CCRTNGFC_LIB_NO_ERROR              (successful)
            # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN              (device not open)
            # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
            # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
            # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
 ***************************************************************************/
```

2.1.15 ccrtNGFC_DC_ADC_Get_Positive_Cal()

This call returns the ADC positive calibration information for all the channels.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Positive_Cal (void                          *AdcHandle,
                                    _ccrtngfc_adc_channel_mask_t   ChanMask,
                                    ccrtngfc_adc_cal_t             *cal)

   Description: Get Daughter Card ADC Positive Calibration Data

   Input:   void                        *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  ccrtngfc_adc_cal_t          *cal         (pointer to board cal)
               uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
               double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (device not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
```

```
              # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE      (ADC is not active)
 ***************************************************************************/
```

2.1.16 ccrtNGFC_DC_ADC_Get_Power_Up_Status()

This call returns the power up status of the ADC module. The module must be activated before the status is returned.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Power_Up_Status(void                          *AdcHandle,
                                       _ccrtngfc_adc_power_up_status_t *PowerUpStatus)

   Description: Get Power Up Status

   Input:   void                                *AdcHandle           (ADC Handle pointer)
   Output:  _ccrtngfc_adc_power_up_status_t     *PowerUpStatus       (Power Up Status)
               _ccrtngfc_adc_all_enable_t       current_state        (active/deactive)
                  # CCRTNGFC_ADC_ALL_DISABLE
                  # CCRTNGFC_ADC_ALL_ENABLE
               _ccrtngfc_presence_status_t      Plus5VoltsStatus
                  # CCRTNGFC_ADC_STATUS_NOT_PRESENT
                  # CCRTNGFC_ADC_STATUS_PRESENT
               _ccrtngfc_presence_status_t      Plus15VoltsStatus
                  # CCRTNGFC_ADC_STATUS_NOT_PRESENT
                  # CCRTNGFC_ADC_STATUS_PRESENT
               _ccrtngfc_presence_status_t      Minus15VoltsStatus
                  # CCRTNGFC_ADC_STATUS_NOT_PRESENT
                  # CCRTNGFC_ADC_STATUS_PRESENT
               _ccrtngfc_adc_power_fail_code_t  PowerFailCode
                  # CCRTNGFC_ADC_PFC_NO_ERRORS
                  # CCRTNGFC_ADC_PFC_BEFORE_PLUS_5_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_BEFORE_PLUS_15_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_BEFORE_MINUS_15_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_FIRST_PLUS_5_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_FIRST_PLUS_15_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_FIRST_MINUS_15_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_AFTER_PLUS_5_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_AFTER_PLUS_15_VOLT_OFF_ERROR
                  # CCRTNGFC_ADC_PFC_AFTER_MINUS_15_VOLT_OFF_ERROR
                char                            PowerFailCodeName[CCRTNGFC_LIB_ERROR_NAME_SIZE]
                char                            PowerFailCodeDesc[CCRTNGFC_LIB_ERROR_DESC_SIZE]
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
 ***************************************************************************/
```

2.1.17 ccrtNGFC_DC_ADC_Get_TestBus_Control()

This call is provided for internal use in testing the hardware.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_TestBus_Control (void                        *AdcHandle,
                                        _ccrtngfc_testbus_control_t *test_control)

   Description: Get Daughter Card ADC Test Bus Control

   Input:   void                    *AdcHandle    (ADC Handle pointer)
```

```
   Output:  _ccrtngfc_testbus_control_t
                                *test_control (pointer to control select)
               # CCRTNGFC_TBUS_CONTROL_OPEN
               # CCRTNGFC_TBUS_CONTROL_CAL_BUS
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR            (successful)
               # CCRTNGFC_LIB_NO_LOCAL_REGION     (local region error)
               # CCRTNGFC_LIB_BAD_HANDLE          (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN            (device not open)
 ***************************************************************************/
```

2.1.18 ccrtNGFC_DC_ADC_Get_Value()

This call allows the user to read the board registers. The actual data returned will depend on the command register information that is requested. Refer to the hardware manual for more information on what is being returned. Most commands return a pointer to an unsigned integer.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Get_Va   lue (void                    *AdcHandle,
                          CCRTNGFC_DC_ADC_CONTROL  cmd,
                          void                     *value)

   Description: Get Daughter Card ADC board register

   Input:   void                    *AdcHandle  (ADC Handle pointer)
            CCRTNGFC_DC_ADC_CONTROL cmd         (register definition)
               -- structure in ccrtngfc_adc.h
   Output:  void                *value;         (pointer to value)
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
 ***************************************************************************/
```

2.1.19 ccrtNGFC_DC_ADC_MsgDma_Configure_Channel()

This call assists the user in setting up modular scatter-gather DMA descriptors for performing channel reads. A moduler scatter-gather DMA engine needs to be first acquired via the *ccrtNGFC_MsgDma_Seize()* API.

There are currently six MsgDma engines available to the user. They can be selected via the *MsgDmaEngine* option. Since the first four MsgDma engines perform quad-word transfers, they must be aligned on a quad-word boundary and the size must be multiples of four, however they are slightly faster than the remaining two that perform single-word transfers. They can be one of the following values:

- CCRTNGFC_MSGDMA_ENGINE_0 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_1 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_2 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_3 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_4 // single-word transfers
- CCRTNGFC_MSGDMA_ENGINE_5 // single-word transfers

The normal process to setup a MsgDma operation is to first call the *ccrtNGFC_MsgDma_Seize()* API with either a specific MsgDma engine or the optional argument *CCRTNGFC_MSGDMA_ENGINE_IDLE.* In this case, the first available MsgDma engine is returned to the user. This is the MsgDma engine number that needs to be used for all the MsgDma operations until the MsgDma is freed via the *ccrtNGFC_MsgDma_Release()* API.

The user also needs to supply to this call a Virtual PCI DMA Memory address which can be obtained with the *ccrtNGFC_MMap_Physical_Memory()* API. Additionally the user will also need to supply a start and end channel number. If the start channel is not divisible by a quad-word or the number of channels selected for the transfer are not a multiple of a quad-word, then the user is restricted to using only *CCRTNGFC_MSGDMA_ENGINE_4* or *CCRTNGFC_MSGDMA_ENGINE_5* engines.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_MsgDma_Configure_Channel (void                    *AdcHandle,
                                             _ccrtngfc_msgdma_engine_t MsgDmaEngine,
                                             void                    *PciDmaMemory,
                                             uint                     StartChannelNumber,
                                             uint                     EndChannelNumber)

   Description: Configure Daughter Card ADC Modular Scatter-Gather MSG Channel Register
                descriptor

   Input:   void                        *AdcHandle          (ADC Handle pointer)
            _ccrtngfc_msgdma_engine_t   MsgDmaEngine
               # CCRTNGFC_MSGDMA_ENGINE_0
               # CCRTNGFC_MSGDMA_ENGINE_1
               # CCRTNGFC_MSGDMA_ENGINE_2
               # CCRTNGFC_MSGDMA_ENGINE_3
               # CCRTNGFC_MSGDMA_ENGINE_4
               # CCRTNGFC_MSGDMA_ENGINE_5
            void                        *PciDmaMemory       (Virtual PCI DMA Memory pointer)
            uint                        StartChannelNumber  (Start Channel Number to Write)
            uint                        EndChannelNumber    (End Channel Number to Write)
   Output:  None
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                     (successful)
               # CCRTNGFC_LIB_BAD_HANDLE                   (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                     (device not open)
               # CCRTNGFC_LIB_INVALID_ARG                  (invalid argument)
               # CCRTNGFC_LIB_MSGDMA_NOT_SUPPORTED         (modular scatter-gather DMA not
                                                            supported)
               # CCRTNGFC_LIB_MSGDMA_BUSY                  (MsgDma Busy, cannot be reset)
               # CCRTNGFC_LIB_NOT_OWNER_OF_MSGDMA          (not owner of modular scatter-gather)
               # CCRTNGFC_LIB_DATA_WIDTH_NOT_MULTIPLE_OR_ALIGNED
                                                           (Data Width Not Multiple or Aligned)
 *************************************************************************/
```

2.1.20 ccrtNGFC_DC_ADC_MsgDma_Configure_Fifo()

This call assists the user in setting up modular scatter-gather DMA descriptors for performing FIFO reads. An moduler scatter-gather DMA engine needs to be first acquired via the *ccrtNGFC_MsgDma_Seize()* API.

There are currently six MsgDma engines available to the user. They can be selected via the *MsgDmaEngine* option. Since the first four MsgDma engines perform quad-word transfers, they must be aligned on a quad-word boundary and the size must be multiples of four, however they are slightly faster than the remaining two that perform single-word transfers. They can be one of the following values:

- CCRTNGFC_MSGDMA_ENGINE_0 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_1 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_2 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_3 // quad-word transfers
- CCRTNGFC_MSGDMA_ENGINE_4 // single-word transfers
- CCRTNGFC_MSGDMA_ENGINE_5 // single-word transfers

The normal process to setup a MsgDma operation is to first call the *ccrtNGFC_MsgDma_Seize()* API with either a specific MsgDma engine or the optional argument *CCRTNGFC_MSGDMA_ENGINE_IDLE.* In this case, the first available MsgDma engine is returned to the user. This is the MsgDma engine number that needs to be used for all the MsgDma operations until the MsgDma is freed via the *ccrtNGFC_MsgDma_Release()* API.

The user also needs to supply to this call a Virtual PCI DMA Memory address which can be obtained with the *ccrtNGFC_MMap_Physical_Memory()* API. Additionally the user will also need to supply number of samples to be read from the FIFO. If the number of samples to be read from the FIFO is not a multiple of a quad-word, then the user is restricted to using only *CCRTNGFC_MSGDMA_ENGINE_4* or *CCRTNGFC_MSGDMA_ENGINE_5* engines.

This call, on successful completion will return a pointer to the Last Descriptor ID *LastDescriptorId* which will be used by subsequent *ccrtNGFC_DC_ADC_MsgDma_Fire_Fifo()* calls to acquire the samples.

```
/*************************************************************************
   ccrtNGFC_DC_ADC_MsgDma_Configure_Fifo()
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_MsgDma_Configure_Fifo (void                                  *AdcHandle,
                                          _ccrtngfc_msgdma_engine_t             MsgDmaEngine,
                                          void                                  *PciDmaMemory,
                                          uint                                  NumberOfSamples,
                                          _ccrtngfc_msgdma_descriptors_id_t     *LastDescriptorId)

   Description: Configure Daughter Card ADC Modular Scatter-Gather MSG Fifo descriptor

   Input:   void                          *AdcHandle         (ADC Handle pointer)
            _ccrtngfc_msgdma_engine_t     MsgDmaEngine
               # CCRTNGFC_MSGDMA_ENGINE_0
               # CCRTNGFC_MSGDMA_ENGINE_1
               # CCRTNGFC_MSGDMA_ENGINE_2
               # CCRTNGFC_MSGDMA_ENGINE_3
               # CCRTNGFC_MSGDMA_ENGINE_4
               # CCRTNGFC_MSGDMA_ENGINE_5
            void                          *PciDmaMemory      (Virtual PCI DMA Memory pointer)
            uint                          NumberOfSamples    (number of FIFO samples to read)
   Output:  _ccrtngfc_msgdma_descriptors_id_t *LastDescriptorId (pointer to last descriptor id)
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                       (successful)
               # CCRTNGFC_LIB_BAD_HANDLE                     (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                       (device not open)
               # CCRTNGFC_LIB_INVALID_ARG                    (invalid argument)
               # CCRTNGFC_LIB_NO_FREE_DESCRIPTORS_AVAILABLE (no free descriptors available)
               # CCRTNGFC_LIB_MSGDMA_NOT_SUPPORTED          (modular scatter-gather DMA not
                                                             supported)
               # CCRTNGFC_LIB_MSGDMA_BUSY                   (MsgDma Busy, cannot be reset)
               # CCRTNGFC_LIB_NOT_OWNER_OF_MSGDMA           (not owner of modular scatter-gather)
               # CCRTNGFC_LIB_DATA_WIDTH_NOT_MULTIPLE_OR_ALIGNED
                                                            (Data Width Not Multiple or Aligned)
 *************************************************************************/
```

2.1.21 ccrtNGFC_DC_ADC_MsgDma_Fire_Channel()

This call initiates a scatter-gather DMA operation that has been previously configured and setup by the *ccrtNGFC_DC_ADC_MsgDma_Configure_Channel()* API for a given MsgDma Engine.

Once the scatter-gather DMA operation commences, it performs a DMA operation on the selected channels in the *ccrtNGFC_DC_ADC_MsgDma_Configure_Channel()* API until the DMA operation completes.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
```

```
   ccrtNGFC_DC_ADC_MsgDma_Fire_Channel (void                              *AdcHandle,
                                       _ccrtngfc_msgdma_engine_t         MsgDmaEngine)

   Description: Fire ADC Daughter Card Channel Modular Scatter-Gather DMA descriptor

   Input:   void                                 *AdcHandle          (ADC Handle pointer)
            _ccrtngfc_msgdma_engine_t            MsgDmaEngine
               # CCRTNGFC_MSGDMA_ENGINE_0
               # CCRTNGFC_MSGDMA_ENGINE_1
               # CCRTNGFC_MSGDMA_ENGINE_2
               # CCRTNGFC_MSGDMA_ENGINE_3
               # CCRTNGFC_MSGDMA_ENGINE_4
               # CCRTNGFC_MSGDMA_ENGINE_5
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_MSGDMA_FAILED        (MsgDma failed)
               # CCRTNGFC_LIB_MSGDMA_BUSY          (MsgDma Busy, cannot be reset)
               # CCRTNGFC_LIB_MSGDMA_NOT_SUPPORTED (modular scatter-gather DMA not supported)
               # CCRTNGFC_LIB_NOT_OWNER_OF_MSGDMA  (not owner of modular scatter-gather)
 ***************************************************************************/
```

2.1.22 ccrtNGFC_DC_ADC_MsgDma_Fire_Fifo()

This call initiates a scatter-gather DMA operation that has been previously configured and setup by the *ccrtNGFC_DC_ADC_MsgDma_Configure_Fifo()* API for a given MsgDma Engine.

Once the scatter-gather DMA operation commences, it performs a DMA operation on the number of samples specified in the *ccrtNGFC_DC_ADC_MsgDma_Configure_Fifo()* API until the DMA operation completes.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_MsgDma_Fire_Fifo (void                                 *AdcHandle,
                                    _ccrtngfc_msgdma_engine_t         MsgDmaEngine,
                                    _ccrtngfc_msgdma_descriptors_id_t LastDescriptorId,
                                    int                               UseInterrupts)

   Description: Fire ADC Daughter Card Fifo Modular Scatter-Gather DMA descriptor

   Input:   void                                 *AdcHandle          (ADC Handle pointer)
            _ccrtngfc_msgdma_engine_t            MsgDmaEngine
               # CCRTNGFC_MSGDMA_ENGINE_0
               # CCRTNGFC_MSGDMA_ENGINE_1
               # CCRTNGFC_MSGDMA_ENGINE_2
               # CCRTNGFC_MSGDMA_ENGINE_3
               # CCRTNGFC_MSGDMA_ENGINE_4
               # CCRTNGFC_MSGDMA_ENGINE_5
            _ccrtngfc_msgdma_descriptors_id_t LastDescriptorId    (Last Descriptor ID)
            int                               UseInterrupts       (Use interrupts flag)
               # CCRTNGFC_TRUE
               # CCRTNGFC_FALSE
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_MSGDMA_FAILED        (MsgDma failed)
               # CCRTNGFC_LIB_MSGDMA_BUSY          (MsgDma Busy, cannot be reset)
               # CCRTNGFC_LIB_MSGDMA_NOT_SUPPORTED (modular scatter-gather DMA not supported)
               # CCRTNGFC_LIB_NOT_OWNER_OF_MSGDMA  (not owner of modular scatter-gather)
 ***************************************************************************/
```

2.1.23 ccrtNGFC_DC_ADC_Open()

This open allows the user to access the various functions on the ADC daughter card which is physically located in one of the two FMC slots on the mother board. Since there can be up to two ADC daughter cards located on a mother board, this call requires the user to identify which ADC module is to be accessed via the *ModuleNumber CCRTNGFC_ADC_MODULE_0* or *CCRTNGFC_ADC_MODULE_1* argument.

This is the first call that needs to be issued by a user to open the ADC daughter card and access the various functions on the card, through the rest of the API calls. If the ADC daughter card is not installed on the mother board, the call will fail with an *CCRTNGFC_LIB_MODULE_NOT_PRESENT* error. Prior to issuing this call, the user can always query the mother board to see if the card is installed by issuing the *ccrtNGFC_DC_ADC_Presence* call.

What is returned on successful completion of the call is a handle to an ADC pointer *AdcHandle* that is supplied as an argument to all the other ADC specific API calls that start with *ccrtNGFC_DC_ADC_*().*

The *oflag* is the flag supplied to the *open(2)* system call by this API. It is normally '0' *(zero)*, however the user may use the *O_NONBLOCK* option for *read(2)* calls which will change the default reading in block mode.

This driver allows multiple applications to open the same board by specifying an additional *oflag O_APPEND*. It is then the responsibility of the user to ensure that the various applications communicating with the same cards are properly synchronized. Various tests supplied in this package has the *O_APPEND* flags enabled, however, it is strongly recommended that only one application be run with a single card at a time, unless the user is well aware of how the applications are going to interact with each other and accept any unpredictable results.

In case of error, *errno* is also set for some non-system related errors encountered.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Open (void                    **AdcHandle,
                         void                    *Handle,
                         _ccrtngfc_adc_module_t  ModuleNumber,
                         int                     oflag)

   Description: Open a Daughter Card ADC module.

   Input:   void                   **AdcHandle      (ADC Handle pointer to pointer)
            void                   *Handle          (Device Handle pointer)
            _ccrtngfc_adc_module_t ModuleNumber     (ADC module number)
               # CCRTNGFC_ADC_MODULE_0              (ADC module on daughter card 0)
               # CCRTNGFC_ADC_MODULE_1              (ADC module on daughter card 1)
            int                    oflag            (open flags)
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_ALREADY_OPEN         (device already opened)
               # CCRTNGFC_LIB_OPEN_FAILED          (device open failed)
               # CCRTNGFC_LIB_ALREADY_MAPPED       (memory already mmapped)
               # CCRTNGFC_LIB_MMAP_SELECT_FAILED   (mmap selection failed)
               # CCRTNGFC_LIB_MMAP_FAILED          (mmap failed)
               # CCRTNGFC_LIB_MODULE_ALREADY_OPEN  (module already opened)
               # CCRTNGFC_LIB_MODULE_NOT_PRESENT   (module not present)
               # CCRTNGFC_LIB_MODULE_OPEN_FAILED   (module open failed)
 *****************************************************************************/
```

2.1.24 ccrtNGFC_DC_ADC_Perform_Auto_Calibration()

This single call performs a full ADC calibration of the selected channels using the internal reference voltages.

```
/*****************************************************************************
```

```
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_Auto_Calibration (void                          *AdcHandle,
                                            _ccrtngfc_adc_channel_mask_t   ChanMask)

   Description: Perform Daughter Card ADC Auto Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE          (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR             (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
               # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE  (Clock is not active)
 ************************************************************************/
```

2.1.25 ccrtNGFC_DC_ADC_Perform_External_Negative_Calibration()

Use this call to perform an external negative calibration. Prior to calling this function, the ADC inputs must be provided with a negative signal close to -10 Volts, otherwise this call will fail. Additionally, the user can specify a range of channels.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_External_Negative_Calibration (void            *AdcHandle,
                                                          _ccrtngfc_adc_channel_mask_t
                                                                          ChanMask,
                                                          Double          ReferenceVoltage)

   Description: Perform Daughter Card ADC External Negative Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
```

```
            # CCRTNGFC_ADC_CHANNEL_MASK_11
            # CCRTNGFC_ALL_ADC_CHANNELS_MASK
          double                          ReferenceVoltage (Reference Voltage)
  Output:  none
  Return:  _ccrtngfc_lib_error_number_t
            # CCRTNGFC_LIB_NO_ERROR               (successful)
            # CCRTNGFC_LIB_BAD_HANDLE             (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN               (library not open)
            # CCRTNGFC_LIB_INVALID_ARG            (invalid argument)
            # CCRTNGFC_LIB_NO_LOCAL_REGION        (local region not present)
            # CCRTNGFC_LIB_NO_RESOURCE            (no free PLL available)
            # CCRTNGFC_LIB_IO_ERROR               (read error)
            # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE      (ADC is not active)
            # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE    (Clock is not active)
*************************************************************************/
```

2.1.26 ccrtNGFC_DC_ADC_Perform_External_Offset_Calibration()

Use this call to perform an external offset calibration. Prior to calling this function, the ADC inputs must be provided with a offset signal close to 0 Volts, otherwise this call will fail. Additionally, the user can specify a range of channels. Once this call is executed, the user will need to perform external negative and external positive calibrations as this call resets these gains to 1.0 prior to calibration.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_External_Offset_Calibration (void                        *AdcHandle,
                                                        _ccrtngfc_adc_channel_mask_t ChanMask)

   Description: Perform Daughter Card ADC External Offset Calibration

   Input:   void                          *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (library not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE           (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR              (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
               # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
*************************************************************************/
```

2.1.27 ccrtNGFC_DC_ADC_Perform_External_Positive_Calibration()

Use this call to perform an external positive calibration. Prior to calling this function, the ADC inputs must be provided with a positive signal close to +10 Volts, otherwise this call will fail. Additionally, the user can specify a range of channels.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_External_Positive_Calibration (void              *AdcHandle,
                                                          _ccrtngfc_adc_channel_mask_t
                                                                            ChanMask,
                                                          double            ReferenceVoltage)

   Description: Perform Daughter Card ADC External Positive Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
            double                          ReferenceVoltage (Reference Voltage)
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (library not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE           (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR              (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
               # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
 *************************************************************************/
```

2.1.28 ccrtNGFC_DC_ADC_Perform_Negative_Calibration()

This call performs a negative calibration using the internal reference voltage.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_Negative_Calibration (void                          *AdcHandle,
                                                 _ccrtngfc_adc_channel_mask_t  ChanMask)

   Description: Perform Daughter Card ADC Negative Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
```

```
            # CCRTNGFC_ADC_CHANNEL_MASK_4
            # CCRTNGFC_ADC_CHANNEL_MASK_5
            # CCRTNGFC_ADC_CHANNEL_MASK_6
            # CCRTNGFC_ADC_CHANNEL_MASK_7
            # CCRTNGFC_ADC_CHANNEL_MASK_8
            # CCRTNGFC_ADC_CHANNEL_MASK_9
            # CCRTNGFC_ADC_CHANNEL_MASK_10
            # CCRTNGFC_ADC_CHANNEL_MASK_11
            # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
            # CCRTNGFC_LIB_NO_ERROR             (successful)
            # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN             (library not open)
            # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
            # CCRTNGFC_LIB_NO_RESOURCE          (no free PLL available)
            # CCRTNGFC_LIB_IO_ERROR             (read error)
            # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
            # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE  (Clock is not active)
 ***************************************************************************/
```

2.1.29 ccrtNGFC_DC_ADC_Perform_Offset_Calibration()

This call performs an offset calibration using the internal reference voltage. Once this call is executed, the user will need to perform negative and positive calibrations as this call resets these gains to 1.0 prior to calibration.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_Offset_Calibration (void                        *AdcHandle,
                                               _ccrtngfc_adc_channel_mask_t ChanMask)

   Description: Perform Daughter Card ADC Offset Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE          (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR             (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
               # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE  (Clock is not active)
 ***************************************************************************/
```

2.1.30 ccrtNGFC_DC_ADC_Perform_Positive_Calibration()

This call performs a positive calibration using the internal reference voltage.

```
/*****************************************************************************
   ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Perform_Positive_Calibration (void                          *AdcHandle,
                                                 _ccrtngfc_adc_channel_mask_t  ChanMask)

   Description: Perform Daughter Card ADC Positive Calibration

   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE          (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR             (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
               # CCRTNGFC_LIB_CLOCK_IS_NOT_ACTIVE  (Clock is not active)
 *****************************************************************************/
```

2.1.31 ccrtNGFC_DC_ADC_Presence()

This call is used to see if an ADC daughter card is installed in one of the two FMC slots on the mother board.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Presence (void                     *Handle,
                             _ccrtngfc_adc_module_t   ModuleNumber,
                             int                      *AdcPresent)

   Description: Check for ADC Daughter Card Presence

   Input:   void                        *Handle         (Board Handle pointer)
            _ccrtngfc_adc_module_t      ModuleNumber    (ADC module number to check)
               # CCRTNGFC_ADC_MODULE_0                  (ADC module on daughter card 0)
               # CCRTNGFC_ADC_MODULE_1                  (ADC module on daughter card 1)
   Output:  int                         *AdcPresent     (ADC Present)
               # CCRTNGFC_CARD_NOT_PRESENT
               # CCRTNGFC_CARD_PRESENT
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                  (successful)
               # CCRTNGFC_LIB_BAD_HANDLE                (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                  (device not open)
               # CCRTNGFC_LIB_INVALID_ARG               (invalid argument)
```

```
            # CCRTNGFC_LIB_NO_LOCAL_REGION          (local region not present)
 ***************************************************************************/
```

2.1.32 ccrtNGFC_DC_ADC_Read()

This call provides the user an easy method of reading the ADC channels. User can supply a channel mask. If pointer to *adc_csr* is NULL, then the routine itself computes the current ADC configuration. For performance, the user should get the current ADC configuration using the *ccrtNGFC_ADC_Get_CSR()* call to get the current settings and pass it to this routine. Hence, if the configuration is not changed, the user can continuously invoke *ccrtNGFC_ADC_Read_Channels()* routine without incurring the additional overhead of routine calling the *ccrtNGFC_ADC_Get_CSR()* call.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Read (void    *AdcHandle,
                         void    *buf,
                         int     size,
                         int     *bytes_read,
                         int     *error)

   Description: Perform Daughter Card ADC driver read operation.

   Input:   void     *AdcHandle                (ADC Handle pointer)
            int      size                      (number of bytes to read)
   Output:  void     *buf                      (pointer to buffer)
            int      *bytes_read               (bytes read)
            int      *error                    (returned errno)
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_IO_ERROR          (read failed)
               # CCRTNGFC_LIB_NOT_IMPLEMENTED   (call not implemented)
 ***************************************************************************/
```

2.1.33 ccrtNGFC_DC_ADC_Read_Channels()

This routine reads the ADC channel data and returns the raw and floating point voltages in the *adc_volts* argument.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Read_Channels (void                        *AdcHandle,
                                  _ccrtngfc_adc_channel_mask_t ChanMask,
                                  _ccrtngfc_adc_csr_t          *adc_csr,
                                  ccrtngfc_adc_volts_t         *adc_volts)

   Description: Read Daughter Card ADC Channels

   Input:   void                            *AdcHandle    (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t    ChanMask      (specify channel mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
```

```
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
            _ccrtngfc_adc_csr_t            *adc_csr (pointer to ADC csr)
               _ccrtngfc_adccsr_update_clock_t      adc_update_clock
                  # CCRTNGFC_ADC_UPDATE_CLOCK_NONE
                  # CCRTNGFC_ADC_UPDATE_CLOCK_0
                  # CCRTNGFC_ADC_UPDATE_CLOCK_1
                  # CCRTNGFC_ADC_UPDATE_CLOCK_2
                  # CCRTNGFC_ADC_UPDATE_CLOCK_3
                  # CCRTNGFC_ADC_UPDATE_CLOCK_4
               _ccrtngfc_adccsr_input_signal_t      adc_input_signal
                  # CCRTNGFC_ADC_EXTERNAL_SIGNAL
                  # CCRTNGFC_ADC_CALIBRATION_BUS
               _ccrtngfc_adccsr_data_format_t       adc_data_format
                  # CCRTNGFC_ADC_OFFSET_BINARY
                  # CCRTNGFC_ADC_TWOS_COMPLEMENT
   Output:  ccrtngfc_adc_volts_t            *adc_volts (pointer to ADC volts)
               uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
               double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (library not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
 ************************************************************************/
```

2.1.34 ccrtNGFC_DC_ADC_Read_Channels_Calibration()

This routine reads the ADC channel calibration registers and dumps all of them to the user specified file. If the file name specified is NULL, then information is written to *stdout.*

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Read_Channels_Calibration (void *AdcHandle,
                                              char *filename)

   Description: Read Daughter Card ADC Channels Calibration

   Input:    void   *AdcHandle                   (ADC Handle pointer)
   Output:   char   *filename                    (pointer to filename)
   Return:   _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR           (successful)
               # CCRTNGFC_LIB_BAD_HANDLE         (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN           (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION    (local region not present)
               # CCRTNGFC_LIB_CANNOT_OPEN_FILE   (cannot open calib. file)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE  (ADC is not active)
 ************************************************************************/
```

2.1.35 ccrtNGFC_DC_ADC_Reset_Calibration()

This call resets the ADC calibration values on the card.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Reset_Calibration (void                          *AdcHandle,
                                      _ccrtngfc_adc_channel_mask_t  ChanMask)

   Description: Reset Daughter Card ADC Calibration
```

```
   Input:   void                         *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE (ADC is not active)
 ***************************************************************************/
```

2.1.36 ccrtNGFC_DC_ADC_Reset_Fifo()

This call provides the ability to reset the ADC FIFO to the power-on state. User can elect to activate the FIFO after a reset in order for data collection to resume immediately.

Note: The first four set of samples that are processed after a FIFO reset are discarded as the high speed ADC requires time to settle before valid data is presented to the user.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Reset_Fifo (void                        *AdcHandle,
                               _ccrtngfc_adc_fifo_reset_t  activate)

   Description: Reset Daughter Card ADC Fifo

   Input:   void                        *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_fifo_reset_t  activate    (activate converter)
               # CCRTNGFC_ADC_FIFO_ACTIVATE
               # CCRTNGFC_ADC_FIFO_RESET
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR          (successful)
               # CCRTNGFC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN          (device not open)
               # CCRTNGFC_LIB_INVALID_ARG       (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION   (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE (ADC is not active)
 ***************************************************************************/
```

2.1.37 ccrtNGFC_DC_ADC_Set_Calibration_CSR()

This call sets the current calibration control and status register.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Calibration_CSR (void                       *AdcHandle,
                                       ccrtngfc_calibration_csr_t *CalCSR)

   Description: Set Daughter Card Calibration Control and Status Register

   Input:   void                                        *AdcHandle  (ADC Handle pointer)
            ccrtngfc_calibration_csr_t                  *CalCSR     (pointer to calibration CSR)
              _ccrtngfc_calbus_control_t                BusControl  (bus control)
                 # CCRTNGFC_CB_GROUND
                 # CCRTNGFC_CB_POSITIVE_REFERENCE
                 # CCRTNGFC_CB_NEGATIVE_REFERENCE
                 # CCRTNGFC_CB_BUS_OPEN
                 # CCRTNGFC_CB_4V_REFERENCE
                 # CCRTNGFC_CB_PLUS_10V_REFERENCE
                 # CCRTNGFC_CB_MINUS_10V_REFERENCE
                 # CCRTNGFC_CB_DAC_CHANNEL_0
                 # CCRTNGFC_CB_DAC_CHANNEL_1
                 # CCRTNGFC_CB_DAC_CHANNEL_2
                 # CCRTNGFC_CB_DAC_CHANNEL_3
                 # CCRTNGFC_CB_DAC_CHANNEL_4
                 # CCRTNGFC_CB_DAC_CHANNEL_5
                 # CCRTNGFC_CB_DAC_CHANNEL_6
                 # CCRTNGFC_CB_DAC_CHANNEL_7
                 # CCRTNGFC_CB_DAC_CHANNEL_8
                 # CCRTNGFC_CB_DAC_CHANNEL_9
                 # CCRTNGFC_CB_DAC_CHANNEL_10
                 # CCRTNGFC_CB_DAC_CHANNEL_11
   Output:     none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                 (successful)
               # CCRTNGFC_LIB_BAD_HANDLE               (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                 (device not open)
               # CCRTNGFC_LIB_INVALID_ARG              (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION          (local region not present)
 *************************************************************************/
```

2.1.38 ccrtNGFC_DC_ADC_Set_CSR()

This call sets the ADC control registers for the selected channel group.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_CSR (void                  *AdcHandle,
                            _ccrtngfc_adc_mask_t  adc_mask,
                            _ccrtngfc_adc_csr_t   *adc_csr)

   Description: Set Daughter Card ADC Control and Status

   Input:   void                                        *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_mask_t                        adc_mask    (selected ADC mask)
               # CCRTNGFC_ADC_MASK_0_3
               # CCRTNGFC_ADC_MASK_4_7
               # CCRTNGFC_ADC_MASK_8_11
               # CCRTNGFC_ALL_ADC_MASK
            _ccrtngfc_adc_csr_t                         *adc_csr (pointer to ADC csr)
               _ccrtngfc_adccsr_update_clock_t          adc_update_clock;
                  # CCRTNGFC_ADC_UPDATE_CLOCK_NONE
                  # CCRTNGFC_ADC_UPDATE_CLOCK_0
                  # CCRTNGFC_ADC_UPDATE_CLOCK_1
```

```
                  # CCRTNGFC_ADC_UPDATE_CLOCK_2
                  # CCRTNGFC_ADC_UPDATE_CLOCK_3
                  # CCRTNGFC_ADC_UPDATE_CLOCK_4
                  # CCRTNGFC_ADC_UPDATE_CLOCK_DO_NOT_CHANGE
               _ccrtngfc_adccsr_input_signal_t          adc_input_signal;
                  # CCRTNGFC_ADC_EXTERNAL_SIGNAL
                  # CCRTNGFC_ADC_CALIBRATION_BUS
                  # CCRTNGFC_ADC_INPUT_SIGNAL_DO_NOT_CHANGE
               _ccrtngfc_adccsr_data_format_t           adc_data_format;
                  # CCRTNGFC_ADC_OFFSET_BINARY
                  # CCRTNGFC_ADC_TWOS_COMPLEMENT
                  # CCRTNGFC_ADC_DATA_FORMAT_DO_NOT_CHANGE
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR                 (successful)
               # CCRTNGFC_LIB_BAD_HANDLE               (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                 (library not open)
               # CCRTNGFC_LIB_BAD_HANDLE               (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN                 (device not open)
               # CCRTNGFC_LIB_INVALID_ARG              (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION          (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE        (ADC is not active)
 ***************************************************************************/
```

2.1.39 ccrtNGFC_DC_ADC_Set_Driver_Read_Mode()

This call sets the current driver ADC read mode. When a *read(2)* system call is issued, it is this mode that determines the type of read being performed by the driver. Refer to the *read(2)* system call under *Direct Driver Access* section for more information on the various modes.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Driver_Read_Mode (void                                *AdcHandle,
                                         _ccrtngfc_driver_ADC_read_mode_t    mode)

   Description: Set Driver Daughter Card ADC Read Mode

   Input:   void                              *AdcHandle (ADC Handle pointer)
            _ccrtngfc_driver_ADC_read_mode_t mode       (select ADC read mode)
               # CCRTNGFC_ADC_PIO_CHANNEL
               # CCRTNGFC_ADC_PIO_FIFO
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (device not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
 ***************************************************************************/
```

2.1.40 ccrtNGFC_DC_ADC_Set_Fifo_Channel_Select()

This call allows the user to select a set of channels that need to be captured in the ADC Fifo.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Fifo_Channel_Select (void                *AdcHandle,
                                            _ccrtngfc_adc_channel_mask_t
                                                                adc_fifo_channel_select_mask)

   Description: Set Daughter Card ADC Fifo Channel Selection
```

```
   Input:   void                          *AdcHandle                      (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t adc_fifo_channel_select_mask   (channel select mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (device not open)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
 ***************************************************************************/
```

2.1.41 ccrtNGFC_DC_ADC_Set_Fifo_Threshold()

This call allows the user to set the ADC Fifo Threshold.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Fifo_Threshold (void *AdcHandle,
                                       uint adc_threshold)

   Description: Set Daughter Card ADC Fifo Threshold

   Input:   void                    *AdcHandle     (ADC Handle pointer)
            uint                    adc_threshold  (ADC fifo threshold)
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (device not open)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION      (local region not present)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE    (ADC is not active)
 ***************************************************************************/
```

2.1.42 ccrtNGFC_DC_ADC_Set_Interrupt_Timeout_Seconds()

This call sets the read ADC *timeout* maintained by the driver. It allows the user to change the default time out from 30 seconds to a user specified value. It is the time that the read call will wait before it times out. The call could time out if the DMA fails to complete. The device should have been opened in the blocking mode *(O_NONBLOCK not set)* for reads to wait for the operation to complete.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Interrupt_Timeout_Seconds (void    *AdcHandle,
                                                  int     timeout_secs)
```

```
   Description: Set Daughter Card ADC Interrupt Timeout Seconds

   Input:   void         *AdcHandle         (ADC Handle pointer)
            int          timeout_secs       (interrupt tout secs)
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR      (successful)
               # CCRTNGFC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN      (device not open)
               # CCRTNGFC_LIB_INVALID_ARG   (invalid argument)
 ************************************************************************/
```

2.1.43 ccrtNGFC_DC_ADC_Set_Negative_Cal()

This call allows the user to set the negative calibration data for all the channels by supplying floating point *Float* gains to the call. Additionally, this call will return the *RAW* value of the gain supplied that is written to the board.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Negative_Cal (void                          *AdcHandle,
                                    _ccrtngfc_adc_channel_mask_t   ChanMask,
                                    ccrtngfc_adc_cal_t             *cal)

   Description: Set Daughter Card ADC Negative Calibration Data

   Input:   void                        *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
            ccrtngfc_adc_cal_t          *cal        (pointer to board cal)
               uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
               double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Output:   none
   Return:   _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (library not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE           (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR              (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
 ************************************************************************/
```

2.1.44 ccrtNGFC_DC_ADC_Set_Offset_Cal()

This call allows the user to set the offset calibration data for all the channels by supplying floating point *Float* offset to the call. Additionally, this call will return the *Raw* value of the offset supplied that is written to the board.

```
/************************************************************************
```

```
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Offset_Cal (void                            *AdcHandle,
                                   _ccrtngfc_adc_channel_mask_t    ChanMask,
                                   ccrtngfc_adc_cal_t              *cal)

   Description: Set Daughter Card ADC Offset Calibration Data

   Input:   void                          *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
               # CCRTNGFC_ADC_CHANNEL_MASK_9
               # CCRTNGFC_ADC_CHANNEL_MASK_10
               # CCRTNGFC_ADC_CHANNEL_MASK_11
               # CCRTNGFC_ALL_ADC_CHANNELS_MASK
            ccrtngfc_adc_cal_t           *cal        (pointer to board cal)
               uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
               double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR              (successful)
               # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN              (library not open)
               # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
               # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCRTNGFC_LIB_NO_RESOURCE           (no free PLL available)
               # CCRTNGFC_LIB_IO_ERROR              (read error)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
 ***************************************************************************/
```

2.1.45 ccrtNGFC_DC_ADC_Set_Positive_Cal()

This call allows the user to set the positive calibration data for all the channels by supplying floating point *Float* gains to the call. Additionally, this call will return the *Raw* value of the gain supplied that is written to the board.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Positive_Cal (void                            *AdcHandle,
                                     _ccrtngfc_adc_channel_mask_t    ChanMask,
                                     ccrtngfc_adc_cal_t              *cal)

   Description: Set Daugther Card ADC Positive Calibration Data

   Input:   void                          *AdcHandle  (ADC Handle pointer)
            _ccrtngfc_adc_channel_mask_t ChanMask    (channel selection mask)
               # CCRTNGFC_ADC_CHANNEL_MASK_0
               # CCRTNGFC_ADC_CHANNEL_MASK_1
               # CCRTNGFC_ADC_CHANNEL_MASK_2
               # CCRTNGFC_ADC_CHANNEL_MASK_3
               # CCRTNGFC_ADC_CHANNEL_MASK_4
               # CCRTNGFC_ADC_CHANNEL_MASK_5
               # CCRTNGFC_ADC_CHANNEL_MASK_6
               # CCRTNGFC_ADC_CHANNEL_MASK_7
               # CCRTNGFC_ADC_CHANNEL_MASK_8
```

```
            # CCRTNGFC_ADC_CHANNEL_MASK_9
            # CCRTNGFC_ADC_CHANNEL_MASK_10
            # CCRTNGFC_ADC_CHANNEL_MASK_11
            # CCRTNGFC_ALL_ADC_CHANNELS_MASK
         ccrtngfc_adc_cal_t          *cal       (pointer to board cal)
            uint    Raw[CCRTNGFC_MAX_ADC_CHANNELS];
            double  Float[CCRTNGFC_MAX_ADC_CHANNELS];
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
            # CCRTNGFC_LIB_NO_ERROR              (successful)
            # CCRTNGFC_LIB_BAD_HANDLE            (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN              (library not open)
            # CCRTNGFC_LIB_INVALID_ARG           (invalid argument)
            # CCRTNGFC_LIB_NO_LOCAL_REGION       (local region not present)
            # CCRTNGFC_LIB_NO_RESOURCE           (no free PLL available)
            # CCRTNGFC_LIB_IO_ERROR              (read error)
            # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE     (ADC is not active)
 ************************************************************************/
```

2.1.46 ccrtNGFC_DC_ADC_Set_TestBus_Control()

This call allows the user to write the calibration registers from a user supplied calibration file. The format of the file is similar to that generated by the *ccrtNGFC_ADC_Read_Channels_Calibration()* call. File can contain comments if they start with '#', '*', or empty lines. Additionally, users need only specify those channels that they wish to calibrate and the order of specifying channels in not important, however, the format of each channel entry needs to be adhered to. E.g. <chxx:> <negative> <offset> <positive>

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_TestBus_Control (void                        *AdcHandle,
                                        _ccrtngfc_testbus_control_t test_control)

   Description: Set Daughter Card ADC Test Bus Control

   Input:   void                    *AdcHandle    (ADC Handle pointer)
   Output:  _ccrtngfc_testbus_control_t
                                    test_control  (control select)
            # CCRTNGFC_TBUS_CONTROL_OPEN
            # CCRTNGFC_TBUS_CONTROL_CAL_BUS
   Return:  _ccrtngfc_lib_error_number_t
            # CCRTNGFC_LIB_NO_ERROR            (successful)
            # CCRTNGFC_LIB_NO_LOCAL_REGION     (local region error)
            # CCRTNGFC_LIB_BAD_HANDLE          (no/bad handler supplied)
            # CCRTNGFC_LIB_NOT_OPEN            (device not open)
 ************************************************************************/
```

2.1.47 ccrtNGFC_DC_ADC_Set_Value()

This call allows the advanced user to set the writable board registers. The actual data written will depend on the command register information that is requested. Refer to the hardware manual for more information on what can be written to.

Normally, users should not be changing these registers as it will bypass the API integrity and could result in an unpredictable outcome.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Set_Value (void                     *AdcHandle,
                              CCRTNGFC_DC_ADC_CONTROL  cmd,
                              void                     *value)
```

```
   Description: Set Daughter Card ADC value of the specified board register.

   Input:   void                    *AdcHandle     (ADC Handle pointer)
            CCRTNGFC_DC_ADC_CONTROL cmd            (register definition)
               -- structure in ccrtngfc_adc.h
            void              *value               (pointer to value to be set)
   Output:  none
   Return:  _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR             (successful)
               # CCRTNGFC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN             (device not open)
               # CCRTNGFC_LIB_INVALID_ARG          (invalid argument)
 ************************************************************************/
```

2.1.48 ccrtNGFC_DC_ADC_Write_Channels_Calibration()

This call allows the user to write the calibration registers from a user supplied calibration file. The format of the file is similar to that generated by the *ccrtNGFC_ADC_Read_Channels_Calibration()* call. File can contain comments if they start with '#', '*', or empty lines. Additionally, users need only specify those channels that they wish to calibrate and the order of specifying channels in not important, however, the format of each channel entry needs to be adhered to. E.g. <chxx:> <negative> <offset> <positive>

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_ADC_Write_Channels_Calibration (void *AdcHandle,
                                               char *filename)

   Description: Write Daughter Card ADC Channels Calibration

   Input:    void   *AdcHandle                   (ADC Handle pointer)
   Output:   char   *filename                    (pointer to filename)
   Return:   _ccrtngfc_lib_error_number_t
               # CCRTNGFC_LIB_NO_ERROR           (successful)
               # CCRTNGFC_LIB_BAD_HANDLE         (no/bad handler supplied)
               # CCRTNGFC_LIB_NOT_OPEN           (library not open)
               # CCRTNGFC_LIB_NO_LOCAL_REGION    (local region not present)
               # CCRTNGFC_LIB_CANNOT_OPEN_FILE   (cannot open calib. file)
               # CCRTNGFC_LIB_INVALID_ARG        (invalid argument)
               # CCRTNGFC_LIB_ADC_IS_NOT_ACTIVE  (ADC is not active)
 ************************************************************************/
```

This page intentionally left blank