SDC 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 SDC Daughter Card which communicates with the Concurrent Real-Time PCI Express Single Daughter Card FPGA I/O Card (SDFC). This SDC daughter card is an optional interface that needs to be physically installed in the FMC slot 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.

Though the driver supports this SDC card being placed in either the Next Generation FPGA I/O Card (NGFC) or the Single Daughter Card FPGA I/O Card(SDFC), the current hardware is only supported for the SDC being placed in the single slot SDFC card, so for now, only CCRTNGFC_SDC_MODULE_0 is applicable.

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

This call must be the first call to activate the SDC. Without activation, all other calls to the SDC will fail. The user can also use this call to return the current state of the SDC without any change by specifying a pointer to *current_state* and setting *activate* to *CCRTNGFC_SDC_ALL_ENABLE_DO_NOT_CHANGE.* If the SDC is already active and the user issues a *CCRTNGFC_SDC_ALL_ENABLE*, no additional activation will be performed. To cause the SDC to go through a full reset, the user needs to issue the *CCRTNGFC_SDC_ALL_RESET* which will cause the SDC to disable and then re-enable, setting all its SDC values to a default state. SDC calibration data will also be reset.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Activate (void                          *SdcHandle,
                             _ccrtngfc_sdc_all_enable_t  activate,
                             _ccrtngfc_sdc_all_enable_t  *current_state)

   Description: Activate/DeActivate Daughter Card SDC module

   Input:   void                                  *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_all_enable_t            activate        (activate/deactivate)
               # CCRTNGFC_SDC_ALL_DISABLE
               # CCRTNGFC_SDC_ALL_ENABLE
               # CCRTNGFC_SDC_ALL_RESET           (disable followed by enable)
               # CCRTNGFC_SDC_ALL_ENABLE_DO_NOT_CHANGE
   Output:  _ccrtngfc_sdc_all_enable_t            *current_state  (active/deactive)
               # CCRTNGFC_SDC_ALL_DISABLE
               # CCRTNGFC_SDC_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_SDC_Close()

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

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Close (void *SdcHandle)

   Description: Close a Daughter Card SDC module.

   Input:   void                            *SdcHandle  (SDC 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)
 *************************************************************************/
```

2.1.3 ccrtNGFC_DC_SDC_Get_Info()

This call returns some useful SDC information.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Get_Info (void               *SdcHandle,
```

```
                                ccrtngfc_sdc_info_t *SdcInfo)

   Description: Get Daughter Card SDC Information

   Input:    void                   *SdcHandle    (SDC Handle pointer)
   Output:   ccrtngfc_sdc_info_t    *SdcInfo      (pointer to returned SdcInfo)
               # int                          Flags               // SDC flags
               # _ccrtngfc_sdc_module_t       ModuleNumber        // SDC module number
               # ccrtngfc_handle_t            *Handle             // Main Device library Handle
               # ccrtngfc_local_ctrl_data_t   *local_ptr          // Main Local register pointer
               # ccrtngfc_1579324_sdc_t       *local_sdc_ptr      // SDC Local register pointer
               # u_int32_t                    *local_sdc_fifo_ptr // SDC Local FIFO register
                                                                     pointer
               # int                          SdcFp               // SDC file descriptor pointer
               # char                         SdcDeviceName[25]   // SDC Device Name
               # ccrtngfc_module_info_t       SdcModuleInfo       // SDC 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)
 ************************************************************************/
```

2.1.4 ccrtNGFC_DC_SDC_Get_Power_Control()

This call returns the power control settings of the SDC module for the transmit channels. The module must be activated before calling this API. User can supply a channel mask.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Get_Power_Control (void                                *SdcHandle,
                                     _ccrtngfc_sdc_channel_mask_t        ChanMask,
                                     ccrtngfc_sdc_data_t                 *Data,
                                     ccrtngfc_sdc_power_control_status_t  *Status)

   Description: Get Daughter Card SDC value of the specified power control

   Input:   void                            *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t    ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_POWERCONTROL_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_POWERCONTROL_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_POWERCONTROL_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_POWERCONTROL_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_POWERCONTROL_MASK
   Output:  ccrtngfc_sdc_data_t             *Data          (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_PowerControl
                   # CCRTNGFC_SDC_PCR_VDD_STATUS_MASK
                   # CCRTNGFC_SDC_PCR_VSS_STATUS_MASK
                   # CCRTNGFC_SDC_PCR_TRANSMIT_H_OVER_TEMP_MASK
                   # CCRTNGFC_SDC_PCR_TRANSMIT_L_OVER_TEMP_MASK
                   # CCRTNGFC_SDC_PCR_POWER_FAIL_CODE_MASK
               - u_int32_t   Channel_1 | RDT2ST_PowerControl
                   # CCRTNGFC_SDC_PCR_VDD_STATUS_MASK
                   # CCRTNGFC_SDC_PCR_VSS_STATUS_MASK
                   # CCRTNGFC_SDC_PCR_TRANSMIT_H_OVER_TEMP_MASK
                   # CCRTNGFC_SDC_PCR_TRANSMIT_L_OVER_TEMP_MASK
                   # CCRTNGFC_SDC_PCR_POWER_FAIL_CODE_MASK
               - u_int32_t   Channel_2 | RDT1AD_PowerControl
                   # CCRTNGFC_SDC_PCR_VDD_STATUS_MASK
```

```
                    # CCRTNGFC_SDC_PCR_VSS_STATUS_MASK
                    # CCRTNGFC_SDC_PCR_TRANSMIT_H_OVER_TEMP_MASK
                    # CCRTNGFC_SDC_PCR_TRANSMIT_L_OVER_TEMP_MASK
                    # CCRTNGFC_SDC_PCR_POWER_FAIL_CODE_MASK
                - u_int32_t   Channel_3 | RDT2AD_PowerControl
                    # CCRTNGFC_SDC_PCR_VDD_STATUS_MASK
                    # CCRTNGFC_SDC_PCR_VSS_STATUS_MASK
                    # CCRTNGFC_SDC_PCR_TRANSMIT_H_OVER_TEMP_MASK
                    # CCRTNGFC_SDC_PCR_TRANSMIT_L_OVER_TEMP_MASK
                    # CCRTNGFC_SDC_PCR_POWER_FAIL_CODE_MASK
            ccrtngfc_sdc_power_control_status_t    *Status     (pointer to Status)
                - ccrtngfc_sdc_power_control_t      pcs   // Channel_0 | RDT1ST_PowerControl
                - ccrtngfc_sdc_power_control_t      pcs   // Channel_1 | RDT2ST_PowerControl
                - ccrtngfc_sdc_power_control_t      pcs   // Channel_2 | RDT1AD_PowerControl
                - ccrtngfc_sdc_power_control_t      pcs   // Channel_3 | RDT2AD_PowerControl
                    Each of the above ccrtngfc_sdc_power_control_t structs contain:
                    - _ccrtngfc_sdc_power_rail_status_t         Plus_VDD_Status;
                         # CCRTNGFC_SDC_POWER_RAIL_STATUS_NOT_PRESENT
                         # CCRTNGFC_SDC_POWER_RAIL_STATUS_PRESENT
                    - _ccrtngfc_sdc_power_rail_status_t         Minus_VSS_Status;
                         # CCRTNGFC_SDC_POWER_RAIL_STATUS_NOT_PRESENT
                         # CCRTNGFC_SDC_POWER_RAIL_STATUS_PRESENT
                    - _ccrtngfc_sdc_over_temperature_status_t   Transmit_High_Over_Temperature;
                         # CCRTNGFC_SDC_OVER_TEMPERATURE_NORMAL
                         # CCRTNGFC_SDC_OVER_TEMPERATURE_ABOVE_NORMAL
                    - _ccrtngfc_sdc_over_temperature_status_t   Transmit_Low_Over_Temperature;
                         # CCRTNGFC_SDC_OVER_TEMPERATURE_NORMAL
                         # CCRTNGFC_SDC_OVER_TEMPERATURE_ABOVE_NORMAL
                    - _ccrtngfc_sdc_transmit_power_fail_code_t  Power_Fail_Code;
                         # CCRTNGFC_SDC_TRANSMIT_PFC_NO_ERRORS
                         # CCRTNGFC_SDC_TRANSMIT_PFC_BEFORE_PLUS_VDD_OFF_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_BEFORE_MINUS_VSS_OFF_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_FIRST_PLUS_VDD_ON_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_FIRST_MINUS_VSS_ON_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_FIRST_HIGH_OVER_TEMP_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_FIRST_LOW_OVER_TEMP_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_AFTER_PLUS_VDD_ON_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_AFTER_MINUS_VSS_ON_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_AFTER_HIGH_OVER_TEMP_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_AFTER_LOW_OVER_TEMP_ERROR
                         # CCRTNGFC_SDC_TRANSMIT_PFC_INVALID_STATUS_DETECTED
                    - _ccrtngfc_sdc_power_control_t             Power_On;
                         # CCRTNGFC_SDC_POWER_OFF
                         # CCRTNGFC_SDC_POWER_ON
                    - 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_SDC_IS_NOT_ACTIVE    (SDC is not active)
     ************************************************************************/
```

2.1.5 ccrtNGFC_DC_SDC_Get_Power_Up_Status()

This call returns the power up status of the SDC module. The module must be activated before calling this API. User can supply a channel mask.

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

   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Get_Power_Up_Status(void                          *SdcHandle,
                                       _ccrtngfc_sdc_power_up_status_t *PowerUpStatus)

   Description: Get Power Up Status

   Input:   void                                *SdcHandle          (SDC Handle pointer)
   Output:  _ccrtngfc_sdc_power_up_status_t     *PowerUpStatus      (Power Up Status)
               _ccrtngfc_sdc_all_enable_t       current_state       (active/deactive)
                  # CCRTNGFC_SDC_ALL_DISABLE
                  # CCRTNGFC_SDC_ALL_ENABLE
               _ccrtngfc_presence_status_t      Plus5VoltsStatus
                  # CCRTNGFC_SDC_STATUS_NOT_PRESENT
                  # CCRTNGFC_SDC_STATUS_PRESENT
               _ccrtngfc_sdc_power_fail_code_t  PowerFailCode
                  # CCRTNGFC_SDC_PFC_NO_ERRORS
                  # CCRTNGFC_SDC_PFC_BEFORE_PLUS_5_VOLT_OFF_ERROR
                  # CCRTNGFC_SDC_PFC_FIRST_PLUS_5_VOLT_OFF_ERROR
                  # CCRTNGFC_SDC_PFC_AFTER_PLUS_5_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)

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

2.1.6 ccrtNGFC_DC_SDC_Get_Terminator()

This call returns the terminator state of the SDC module for the receive channels. The module must be activated before calling this API. User can supply a channel mask.

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Get_Terminator (void                          *SdcHandle,
                                   _ccrtngfc_sdc_channel_mask_t ChanMask,
                                   ccrtngfc_sdc_data_t          *Data)

   Description: Get Daughter Card SDC value of the specified terminator

   Input:   void                              *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t      ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_CMD1_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_CMD2_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_TST1_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_TST2_TERMINATOR_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TERMINATOR_MASK
   Output:  ccrtngfc_sdc_data_t               *Data           (pointer to Data)
               - u_int32_t   Channel_0 | CMD1_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
                   # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_1 | CMD2_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
```

```
                  # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_2 | TST1_Terminator
                  # CCRTNGFC_SDC_TERMINATOR_OFF
                  # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_3 | TST2_Terminator
                  # CCRTNGFC_SDC_TERMINATOR_OFF
                  # CCRTNGFC_SDC_TERMINATOR_ON
   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_SDC_IS_NOT_ACTIVE    (SDC is not active)
 ************************************************************************/
```

2.1.7 ccrtNGFC_DC_SDC_Get_Turn_Around()

This call returns the turn around state of the SDC module for the transmit channels. The module must be activated before calling this API. User can supply a channel mask.

```
/************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Get_Turn_Around (void                          *SdcHandle,
                                    _ccrtngfc_sdc_channel_mask_t ChanMask,
                                    ccrtngfc_sdc_data_t          *Data)

   Description: Get Daughter Card SDC value of the specified turn around

   Input:   void                              *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t      ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_TURNAROUND_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TURNAROUND_MASK
   Output:  ccrtngfc_sdc_data_t               *Data          (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_Turnaround
                  # CCRTNGFC_SDC_TURNAROUND_NORMAL
                  # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_1 | RDT2ST_Turnaround
                  # CCRTNGFC_SDC_TURNAROUND_NORMAL
                  # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_2 | RDT1AD_Turnaround
                  # CCRTNGFC_SDC_TURNAROUND_NORMAL
                  # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_3 | RDT2AD_Turnaround
                  # CCRTNGFC_SDC_TURNAROUND_NORMAL
                  # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
   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_SDC_IS_NOT_ACTIVE    (SDC is not active)
 ************************************************************************/
```

2.1.8 ccrtNGFC_DC_SDC_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_SDC_Get_Value (void                      *SdcHandle,
                              CCRTNGFC_DC_SDC_CONTROL  cmd,
                              void                     *value)

   Description: Get Daughter Card SDC board register

   Input:   void                    *SdcHandle  (SDC Handle pointer)
            CCRTNGFC_DC_SDC_CONTROL cmd         (register definition)
               # CCRTNGFC_DC_SDC_ENABLE

               # CCRTNGFC_DC_SDC_RECEIVER_CH0     | CCRTNGFC_DC_SDC_CMD1_RECEIVER
               # CCRTNGFC_DC_SDC_RECEIVER_CH1     | CCRTNGFC_DC_SDC_CMD2_RECEIVER
               # CCRTNGFC_DC_SDC_RECEIVER_CH2     | CCRTNGFC_DC_SDC_TST1_RECEIVER
               # CCRTNGFC_DC_SDC_RECEIVER_CH3     | CCRTNGFC_DC_SDC_TST2_RECEIVER

               # CCRTNGFC_DC_SDC_TERMINATOR_CH0   | CCRTNGFC_DC_SDC_CMD1_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH1   | CCRTNGFC_DC_SDC_CMD2_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH2   | CCRTNGFC_DC_SDC_TST1_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH3   | CCRTNGFC_DC_SDC_TST2_TERMINATOR

               # CCRTNGFC_DC_SDC_TRANSMITTER_CH0  | CCRTNGFC_DC_SDC_RDT1ST_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH1  | CCRTNGFC_DC_SDC_RDT2ST_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH2  | CCRTNGFC_DC_SDC_RDT1AD_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH3  | CCRTNGFC_DC_SDC_RDT2AD_TRANSMITTER

               # CCRTNGFC_DC_SDC_TURNAROUND_CH0   | CCRTNGFC_DC_SDC_RDT1ST_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH1   | CCRTNGFC_DC_SDC_RDT2ST_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH2   | CCRTNGFC_DC_SDC_RDT1AD_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH3   | CCRTNGFC_DC_SDC_RDT2AD_TURNAROUND

               # CCRTNGFC_DC_SDC_POWERCONTROL_CH0 | CCRTNGFC_DC_SDC_RDT1ST_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH1 | CCRTNGFC_DC_SDC_RDT2ST_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH2 | CCRTNGFC_DC_SDC_RDT1AD_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH3 | CCRTNGFC_DC_SDC_RDT2AD_POWERCONTROL
   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)
 *****************************************************************************/
```

2.1.9 ccrtNGFC_DC_SDC_Open()

This open allows the user to access the various functions on the SDC daughter card which is physically located in one of the two FMC slots on the mother board. Since there can be up to two SDC daughter cards located on a mother board, this call requires the user to identify which SDC module is to be accessed via the *ModuleNumber CCRTNGFC_SDC_MODULE_0* or *CCRTNGFC_SDC_MODULE_1* argument.

This is the first call that needs to be issued by a user to open the SDC daughter card and access the various functions on the card, through the rest of the API calls. If the SDC 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_SDC_Presence* call.

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

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_SDC_Open (void                    **SdcHandle,
                         void                    *Handle,
                         _ccrtngfc_sdc_module_t  ModuleNumber,
                         int                     oflag)
   Description: Open a Daughter Card SDC module.

   Input:   void                    **SdcHandle      (SDC Handle pointer to pointer)
            void                    *Handle          (Device Handle pointer)
            _ccrtngfc_sdc_module_t ModuleNumber     (SDC module number)
               # CCRTNGFC_SDC_MODULE_0              (SDC module on daughter card 0)
               # CCRTNGFC_SDC_MODULE_1              (SDC 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)
 *****************************************************************************/
```

2.1.10 ccrtNGFC_DC_SDC_Presence()

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

```
/*****************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Presence (void                     *Handle,
                             _ccrtngfc_sdc_module_t   ModuleNumber,
                             int                      *SdcPresent)

   Description: Check for SDC Daughter Card Presence

   Input:   void                        *Handle         (Board Handle pointer)
            _ccrtngfc_sdc_module_t      ModuleNumber    (SDC module number to check)
               # CCRTNGFC_SDC_MODULE_0                  (SDC module on daughter card 0)
               # CCRTNGFC_SDC_MODULE_1                  (SDC module on daughter card 1)
   Output:  int                         *SdcPresent     (SDC 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)
 ***************************************************************************/
```

2.1.11 ccrtNGFC_DC_SDC_Read_Receiver()

This call provides the user an easy method of reading the SDC receiver channels. The module must be activated before calling this API. User can supply a channel mask.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Read_Receiver (void                        *SdcHandle,
                                  _ccrtngfc_sdc_channel_mask_t ChanMask,
                                  ccrtngfc_sdc_data_t          *Data)

   Description: Read Daughter Card SDC value of the specified receiver

   Input:   void                             *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t     ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_CMD1_RECEIVER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_CMD2_RECEIVER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_TST1_RECEIVER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_TST2_RECEIVER_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_RECEIVER_MASK
   Output:  ccrtngfc_sdc_data_t              *Data          (pointer to Data)
               - u_int32_t   Channel_0 | CMD1_Receiver
                   # CCRTNGFC_SDC_RECEIVER_STATE_NULL
                   # CCRTNGFC_SDC_RECEIVER_STATE_HIGH
                   # CCRTNGFC_SDC_RECEIVER_STATE_LOW
                   # CCRTNGFC_SDC_RECEIVER_STATE_INVALID
               - u_int32_t   Channel_1 | CMD2_Receiver
                   # CCRTNGFC_SDC_RECEIVER_STATE_NULL
                   # CCRTNGFC_SDC_RECEIVER_STATE_HIGH
                   # CCRTNGFC_SDC_RECEIVER_STATE_LOW
                   # CCRTNGFC_SDC_RECEIVER_STATE_INVALID
               - u_int32_t   Channel_2 | TST1_Receiver
                   # CCRTNGFC_SDC_RECEIVER_STATE_NULL
                   # CCRTNGFC_SDC_RECEIVER_STATE_HIGH
                   # CCRTNGFC_SDC_RECEIVER_STATE_LOW
                   # CCRTNGFC_SDC_RECEIVER_STATE_INVALID
               - u_int32_t   Channel_3 | TST2_Receiver
                   # CCRTNGFC_SDC_RECEIVER_STATE_NULL
                   # CCRTNGFC_SDC_RECEIVER_STATE_HIGH
                   # CCRTNGFC_SDC_RECEIVER_STATE_LOW
                   # CCRTNGFC_SDC_RECEIVER_STATE_INVALID
   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_SDC_IS_NOT_ACTIVE    (SDC is not active)
 ***************************************************************************/
```

2.1.12 ccrtNGFC_DC_SDC_Read_Transmitter()

This call provides the user an easy method of reading the SDC transmitter channels. The module must be activated before calling this API. User can supply a channel mask.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Read_Transmitter (void                          *SdcHandle,
                                     _ccrtngfc_sdc_channel_mask_t ChanMask,
                                     ccrtngfc_sdc_data_t          *Data)

   Description: Read Daughter Card SDC value of the specified transmitter

   Input:   void                             *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t     ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_TRANSMITTER_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TRANSMITTER_MASK
   Output:  ccrtngfc_sdc_data_t              *Data          (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_Transmitter
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_1 | RDT2ST_Transmitter
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_2 | RDT1AD_Transmitter
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_3 | RDT2AD_Transmitter
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                   # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
   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_SDC_IS_NOT_ACTIVE   (SDC is not active)
 *************************************************************************/
```

2.1.13 ccrtNGFC_DC_SDC_Set_Power_Control()

This call provides the user an easy method of setting the SDC power control for the transmit channels. The module must be activated before calling this API. User can supply a channel mask.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Set_Power_Control (void                              *SdcHandle,
                                      _ccrtngfc_sdc_channel_mask_t     ChanMask,
                                      ccrtngfc_sdc_data_t              *Data)

   Description: Set Power Control

   Input:   void                             *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t     ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_POWERCONTROL_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_POWERCONTROL_MASK
```

```
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_POWERCONTROL_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_POWERCONTROL_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_POWERCONTROL_MASK
            ccrtngfc_sdc_data_t             *Data           (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_PowerControl
                   # CCRTNGFC_SDC_POWER_OFF
                   # CCRTNGFC_SDC_POWER_ON
               - u_int32_t   Channel_1 | RDT2ST_PowerControl
                   # CCRTNGFC_SDC_POWER_OFF
                   # CCRTNGFC_SDC_POWER_ON
               - u_int32_t   Channel_2 | RDT1AD_PowerControl
                   # CCRTNGFC_SDC_POWER_OFF
                   # CCRTNGFC_SDC_POWER_ON
               - u_int32_t   Channel_3 | RDT2AD_PowerControl
                   # CCRTNGFC_SDC_POWER_OFF
                   # CCRTNGFC_SDC_POWER_ON
   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_FPGA_PM_FAILURE      (SDC transmit power module failure)
               # CCRTNGFC_LIB_SDC_IS_NOT_ACTIVE    (SDC is not active)
 ***************************************************************************/
```

2.1.14 ccrtNGFC_DC_SDC_Set_Terminator()

This call provides the user an easy method of setting the SDC terminators for the receive channels. The module must be activated before calling this API. User can supply a channel mask.

```
/***************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Set_Terminator (void                            *SdcHandle,
                                   _ccrtngfc_sdc_channel_mask_t    ChanMask,
                                   ccrtngfc_sdc_data_t             *Data)

   Description: Set Terminator

   Input:   void                           *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t   ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_CMD1_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_CMD2_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_TST1_TERMINATOR_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_TST2_TERMINATOR_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TERMINATOR_MASK
            ccrtngfc_sdc_data_t             *Data          (pointer to Data)
               - u_int32_t   Channel_0 | CMD1_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
                   # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_1 | CMD2_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
                   # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_2 | TST1_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
                   # CCRTNGFC_SDC_TERMINATOR_ON
               - u_int32_t   Channel_3 | TST2_Terminator
                   # CCRTNGFC_SDC_TERMINATOR_OFF
                   # CCRTNGFC_SDC_TERMINATOR_ON
   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_SDC_IS_NOT_ACTIVE    (SDC is not active)
*************************************************************************/
```

2.1.15 ccrtNGFC_DC_SDC_Set_Turn_Around()

This call provides the user an easy method of setting the SDC turn around for the transmit channels. The module must be activated before calling this API. User can supply a channel mask.

```
/*************************************************************************
   _ccrtngfc_lib_error_number_t
   ccrtNGFC_DC_SDC_Set_Turn_Around (void                              *SdcHandle,
                                    _ccrtngfc_sdc_channel_mask_t      ChanMask,
                                    ccrtngfc_sdc_data_t               *Data)

   Description: Set Turn Around

   Input:   void                             *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t     ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_TURNAROUND_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_TURNAROUND_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TURNAROUND_MASK
   Output:  ccrtngfc_sdc_data_t              *Data           (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_Turnaround
                   # CCRTNGFC_SDC_TURNAROUND_NORMAL
                   # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_1 | RDT2ST_Turnaround
                   # CCRTNGFC_SDC_TURNAROUND_NORMAL
                   # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_2 | RDT1AD_Turnaround
                   # CCRTNGFC_SDC_TURNAROUND_NORMAL
                   # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
               - u_int32_t   Channel_3 | RDT2AD_Turnaround
                   # CCRTNGFC_SDC_TURNAROUND_NORMAL
                   # CCRTNGFC_SDC_TURNAROUND_OUTPUT_TO_INPUT
   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_SDC_IS_NOT_ACTIVE    (SDC is not active)
 *************************************************************************/
```

2.1.16 ccrtNGFC_DC_SDC_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_SDC_Set_Value (void                     *SdcHandle,
                              CCRTNGFC_DC_SDC_CONTROL  cmd,
```

```
                                void                    *value)

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

   Input:   void                    *SdcHandle     (SDC Handle pointer)
            CCRTNGFC_DC_SDC_CONTROL cmd            (register definition)
               # CCRTNGFC_DC_SDC_ENABLE

               # CCRTNGFC_DC_SDC_TERMINATOR_CH0   | CCRTNGFC_DC_SDC_CMD1_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH1   | CCRTNGFC_DC_SDC_CMD2_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH2   | CCRTNGFC_DC_SDC_TST1_TERMINATOR
               # CCRTNGFC_DC_SDC_TERMINATOR_CH3   | CCRTNGFC_DC_SDC_TST2_TERMINATOR

               # CCRTNGFC_DC_SDC_TRANSMITTER_CH0  | CCRTNGFC_DC_SDC_RDT1ST_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH1  | CCRTNGFC_DC_SDC_RDT2ST_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH2  | CCRTNGFC_DC_SDC_RDT1AD_TRANSMITTER
               # CCRTNGFC_DC_SDC_TRANSMITTER_CH3  | CCRTNGFC_DC_SDC_RDT2AD_TRANSMITTER

               # CCRTNGFC_DC_SDC_TURNAROUND_CH0   | CCRTNGFC_DC_SDC_RDT1ST_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH1   | CCRTNGFC_DC_SDC_RDT2ST_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH2   | CCRTNGFC_DC_SDC_RDT1AD_TURNAROUND
               # CCRTNGFC_DC_SDC_TURNAROUND_CH3   | CCRTNGFC_DC_SDC_RDT2AD_TURNAROUND

               # CCRTNGFC_DC_SDC_POWERCONTROL_CH0 | CCRTNGFC_DC_SDC_RDT1ST_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH1 | CCRTNGFC_DC_SDC_RDT2ST_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH2 | CCRTNGFC_DC_SDC_RDT1AD_POWERCONTROL
               # CCRTNGFC_DC_SDC_POWERCONTROL_CH3 | CCRTNGFC_DC_SDC_RDT2AD_POWERCONTROL
            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)
 ****************************************************************************/
```

2.1.17 ccrtNGFC_DC_SDC_Write_Transmitter()

This call provides the user an easy method of writing to the SDC transmit channels. The module must be activated before calling this API. User can supply a channel mask. Normally, the writes are posted and return immediately before the actual write is completed. The ForceReadback option can be set to CCRTNGFC_TRUE to enable Force Readback.

```
/****************************************************************************
  _ccrtngfc_lib_error_number_t
  ccrtNGFC_DC_SDC_Write_Transmitter (void                              *SdcHandle,
                                    _ccrtngfc_sdc_channel_mask_t      ChanMask,
                                    ccrtngfc_sdc_data_t               *Data,
                                    int                               ForceReadback)

   Description: Write to Transmitter

   Input:   void                             *SdcHandle      (SDC Handle pointer)
            _ccrtngfc_sdc_channel_mask_t     ChanMask        (specify channel mask)
               # CCRTNGFC_SDC_CHANNEL_MASK_0    | CCRTNGFC_SDC_RDT1ST_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_1    | CCRTNGFC_SDC_RDT2ST_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_2    | CCRTNGFC_SDC_RDT1AD_TRANSMITTER_MASK
               # CCRTNGFC_SDC_CHANNEL_MASK_3    | CCRTNGFC_SDC_RDT2AD_TRANSMITTER_MASK
               # CCRTNGFC_SDC_ALL_CHANNELS_MASK | CCRTNGFC_SDC_ALL_TRANSMITTER_MASK
            ccrtngfc_sdc_data_t              *Data          (pointer to Data)
               - u_int32_t   Channel_0 | RDT1ST_Transmitter
```

```
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_1 | RDT2ST_Transmitter
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_2 | RDT1AD_Transmitter
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
               - u_int32_t   Channel_3 | RDT2AD_Transmitter
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_NULL
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_HIGH
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_LOW
                  # CCRTNGFC_SDC_TRANSMITTER_STATE_INVALID
            int                              ForceReadback;
               # CCRTNGFC_TRUE
               # CCRTNGFC_FALSE:ww
  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_SDC_IS_NOT_ACTIVE    (SDC is not active)
*****************************************************************************/
```

This page intentionally left blank