Software Interface
CCURDPRC (WC-DPRC)

PCIe Digital Programmable Resistance Card (DPRC)

Driver	ccurdprc (WC-DPRC)	
OS	RedHawk (CentOS or Ubuntu based)	
Vendor	Concurrent Real-Time	
Hardware	PCIe Digital Programable Resister Card (DPRC)	
Author	Darius Dubash	
Date	October 3rd, 2023	Rev 2023.1

This page intentionally left blank

Table of Contents

This page intentionally left blank

1. Introduction

This document provides the software interface to the ***ccurdprc*** driver which communicates with the Concurrent Real-Time PCI Express Digital Programmable Resistance Card (DPRC).

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 Digital Programmable Resistance Card Driver Installation on RedHawk Release Notes by Concurrent Real-Time.

2. Software Support

Software support is provided for users to communicate directly with the board using the kernel system calls *(Direct Driver Access)* or the supplied *API.* Both approaches are identified below to assist the user in software development.

2.1 Direct Driver Access

2.1.1 open(2) system call

In order to access the board, the user first needs to open the device using the standard system call *open(2)*.

```
int   fp;
fp =  open("/dev/ccurdprc0", O_RDWR);
```

The file pointer '*fp'* is then used as an argument to other system calls. The user can also supply the O_NONBLOCK flag if the user does not wish to block waiting for reads to complete. In that case, if the read is not satisfied, the call will fail. The device name specified is of the format "/dev/ccurdprc<num>" where *num* is a digit 0..9 which represents the board number that is to be accessed. Basically, the driver only allows one application to open a board at a time. The reason for this is that the application can have full access to the card, even at the board and API level. If another application were to communicate with the same card concurrently, the results would be unpredictable unless proper synchronization between applications is performed external to the driver.

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.

2.1.2 ioctl(2) system call

This system call provides the ability to control and get responses from the board. The nature of the control/response will depend on the specific *ioctl* command.

```
int   status;
int   arg;
status = ioctl(fp, <IOCTL_COMMAND>, &arg);
```

where, '*fp'* is the file pointer that is returned from the *open(2)* system call. <*IOCTL_COMMAND*> is one of the *ioctl* commands below and *arg* is a pointer to an argument that could be anything and is dependent on the command being invoked. If no argument is required for a specific command, then set to *NULL*.

Driver IOCTL command:

```
IOCTL_CCURDPRC_ABORT_DMA
IOCTL_CCURDPRC_ADD_IRQ
IOCTL_CCURDPRC_DISABLE_PCI_INTERRUPTS
IOCTL_CCURDPRC_ENABLE_PCI_INTERRUPTS
IOCTL_CCURDPRC_GET_DRIVER_ERROR
IOCTL_CCURDPRC_GET_DRIVER_INFO
IOCTL_CCURDPRC_GET_PHYSICAL_MEMORY
IOCTL_CCURDPRC_GET_READ_MODE
IOCTL_CCURDPRC_GET_WRITE_MODE
IOCTL_CCURDPRC_INIT_BOARD
IOCTL_CCURDPRC_INTERRUPT_TIMEOUT_SECONDS
IOCTL_CCURDPRC_MAIN_CONTROL_REGISTERS
IOCTL_CCURDPRC_MMAP_SELECT
IOCTL_CCURDPRC_NO_COMMAND
IOCTL_CCURDPRC_PCI_BRIDGE_REGISTERS
IOCTL_CCURDPRC_PCI_CONFIG_REGISTERS
IOCTL_CCURDPRC_READ_EEPROM
IOCTL_CCURDPRC_REMOVE_IRQ
IOCTL_CCURDPRC_RESET_BOARD
IOCTL_CCURDPRC_SELECT_READ_MODE
IOCTL_CCURDPRC_SELECT_WRITE_MODE
IOCTL_CCURDPRC_WAIT_FOR_INTERRUPT
IOCTL_CCURDPRC_WRITE_EEPROM
```

IOCTL_CCURDPRC_ABORT_DMA: This *ioctl* does not have any arguments. Its purpose is to abort any DMA already in progress.

IOCTL_CCURDPRC_ADD_IRQ: This *ioctl* does not have any arguments. Its purpose is to setup the driver *interrupt handler* to handle interrupts. If support for MSI interrupts are configured, they will be enabled. Normally, there is no need to call this *ioctl* as the interrupt handler is already added when the driver is loaded. This *ioctl* should only be invoked if the user has issued the *IOCTL_CCURDPRC_REMOVE_IRQ* call earlier to remove the interrupt handler.

IOCTL_CCURDPRC_DISABLE_PCI_INTERRUPTS: This *ioctl* does not have any arguments. Its purpose is to disable PCI interrupts. This call shouldn't be used during normal reads or writes, as calls could time out. The driver handles enabling and disabling interrupts during its normal course of operation.

IOCTL_CCURDPRC_ENABLE_PCI_INTERRUPTS: This *ioctl* does not have any arguments. Its purpose is to enable PCI interrupts. This call shouldn't be used during normal reads or writes as calls could time out. The driver handles enabling and disabling interrupts during its normal course of operation.

IOCTL_CCURDPRC_GET_DRIVER_ERROR: The argument supplied to this *ioctl* is a pointer to the *ccurdprc_user_error_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. The error returned is the last reported error by the driver. If the argument pointer is *NULL*, the current error is reset to *CCURDPRC_SUCCESS.*

IOCTL_CCURDPRC_GET_DRIVER_INFO: The argument supplied to this *ioctl* is a pointer to the *ccurdprc_driver_info_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. This *ioctl* provides useful driver information.

IOCTL_CCURDPRC_GET_PHYSICAL_MEMORY: The argument supplied to this *ioctl* is a pointer to the *ccurdprc_user_phys_mem_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. If physical memory is not allocated, the call will fail; otherwise the call will return the physical memory address and size in bytes. The only reason to request and get physical memory from the driver is to allow the user to perform DMA operations and bypass the driver and library. Care must be taken when performing user level DMA, as incorrect programming could lead to unpredictable results, including but not limited to corrupting the kernel and any device connected to the system.

IOCTL_CCURDPRC_GET_READ_MODE: The argument supplied to this *ioctl* is a pointer an *unsigned long int*. The value returned will be one of the read modes as defined by the *enum _ccurdprc_driver_rw_mode_t* located in the *ccurdprc_user.h* include file. Currently, only the *CCURDPRC_PIO_CHANNEL* mode is supported for driver reads.

IOCTL_CCURDPRC_GET_WRITE_MODE: (CURRENTLY NOT IMPLEMENTED) The argument supplied to this *ioctl* is a pointer an *unsigned long int*. The value returned will be one of the write modes as defined by the *enum _ccurdprc_driver_rw_mode_t* located in the *ccurdprc_user.h* include file. This call is not supported for driver writes.

IOCTL_CCURDPRC_INIT_BOARD: This *ioctl* does not have any arguments. This call resets the board to a known initial default state. This call is currently identical to the *IOCTL_CCURDPRC_RESET_BOARD* call.

IOCTL_CCURDPRC_INTERRUPT_TIMEOUT_SECONDS: The argument supplied to this *ioctl* is a pointer to an *int*. It allows the user to change the default time out from 30 seconds to user supplied time out. This is the time that the read call will wait before it times out. The call could time out if a DMA fails to complete. The device should have been opened in the block mode (*O_NONBLOCK* not set) for reads to wait for an operation to complete.

IOCTL_CCURDPRC_MAIN_CONTROL_REGISTERS: This *ioctl* dumps all the PCI Main Control registers and is mainly used for debug purpose. The argument to this *ioctl* is a pointer to the *ccurdprc_main_control_register_t* structure. Raw 32-bit data values are read from the board and loaded into this structure.

IOCTL_CCURDPRC_MMAP_SELECT: The argument to this *ioctl* is a pointer to the *ccurdprc_mmap_select_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. This call needs to be made prior to the *mmap(2)* system call so as to direct the *mmap(2)* call to perform the requested mapping specified by this *ioctl*. The four possible mappings that are performed by the driver are to *mmap* the local register space *(CCURDPRC_SELECT_LOCAL_MMAP)*, the configuration register space *(CCURDPRC_SELECT_CONFIG_MMAP)* the physical memory *(CCURDPRC_SELECT_PHYS_MEM_MMAP)* that is created by the *mmap(2)* system call and the driver/library mapping *(CCURDPRC_SELECT_DRIVER_LIBRARY_MMAP)*.

IOCTL_CCURDPRC_NO_COMMAND: This *ioctl* does not have any arguments. It is only provided for debugging purpose and should not be used as it serves no purpose for the application.

IOCTL_CCURDPRC_PCI_BRIDGE_REGISTERS: This *ioctl* dumps all the PCI bridge registers and is mainly used for debug purpose. The argument to this *ioctl* is a pointer to the *ccurdprc_pci_bridge_register_t* structure. Raw 32-bit data values are read from the board and loaded into this structure.

IOCTL_CCURDPRC_PCI_CONFIG_REGISTERS: The argument supplied to this *ioctl* is a pointer to the *ccurdprc_pci_config_reg_addr_mapping_*t structure whose definition is located in the *ccurdprc_user.h* include file.

IOCTL_CCURDPRC_READ_EEPROM: The argument to this *ioctl* is a pointer to the *ccurdprc_eeprom_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. This call is specifically used by the supplied *eeprom* application and should not be used by the user.

IOCTL_CCURDPRC_REMOVE_IRQ: This *ioctl* does not have any arguments. Its purpose is to remove the interrupt handler that was previously setup. The interrupt handler is managed internally by the driver and the library. The user should not issue this call, otherwise reads will time out.

IOCTL_CCURDPRC_RESET_BOARD: This *ioctl* does not have any arguments. This call resets the board to a known initial default state. This call is currently identical to the *IOCTL_CCURDPRC_INIT_BOARD* call.

IOCTL_CCURDPRC_SELECT_READ_MODE: The argument supplied to this *ioctl* is a pointer an *unsigned long int*. The value set will be one of the read modes as defined by the *enum _ccurdprc_driver_rw_mode_t* located in the *ccurdprc_user.h* include file. Currently, only the *CCURDPRC_PIO_CHANNEL* mode is supported for driver reads.

IOCTL_CCURDPRC_SELECT_WRITE_MODE: (CURRENTLY NOT IMPLEMENTED) The argument supplied to this *ioctl* is a pointer an *unsigned long int*. The value set will be one of the write modes as defined by the *enum _ccurdprc_driver_rw_mode_t* located in the *ccurdprc_user.h* include file. This call is not supported for driver writes.

IOCTL_CCURDPRC_WAIT_FOR_INTERRUPT: The argument to this *ioctl* is a pointer to the *ccurdprc_driver_int_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. The user can wait for a DMA or Analog signal complete interrupt. If a time out value greater than zero is specified, the call will time out after the specified seconds, otherwise it will not time out.

IOCTL_CCURDPRC_WRITE_EEPROM: The argument to this *ioctl* is a pointer to the *ccurdprc_eeprom_t* structure. Information on the structure is located in the *ccurdprc_user.h* include file. This call is specifically used by the supplied *eeprom* application and should not be used by the user.

2.1.3 mmap(2) system call

This system call provides the ability to map either the local board registers, the configuration board registers, create and map a physical memory that can be used for user DMA or driver/library structure mapping. Prior to making this system call, the user needs to issue the *ioctl(2)* system call with the *IOCTL_CCURDPRC_MMAP_SELECT* command. When mapping either the local board registers or the configuration board registers, the *ioctl* call returns the size of the register mapping which needs to be specified in the *mmap(2)* call. In the case of mapping a physical memory, the size of physical memory to be created is supplied to the *mmap(2)* call.

```
int *munmap_local_ptr;
ccurdprc_local_ctrl_data_t  *local_ptr;
ccurdprc_mmap_select_t  mmap_select;
unsigned long mmap_local_size;

mmap_select.select = CCURDPRC_SELECT_LOCAL_MMAP;
mmap_select.offset=0;
mmap_select.size=0;
ioctl(fp, IOCTL_CCURDPRC_MMAP_SELECT,(void *)&mmap_select);
mmap_local_size = mmap_select.size;
munmap_local_ptr = (int *) mmap((caddr_t)0, map_local_size,
                  (PROT_READ|PROT_WRITE), MAP_SHARED, fp, 0);
```

```
local_ptr = (ccurdprc_local_ctrl_data_t *)munmap_local_ptr;
local_ptr = (ccurdprc_local_ctrl_data_t *)((char *)local_ptr +
                                            mmap_select.offset);
.
.
.

if(munmap_local_ptr != NULL)
    munmap((void *)munmap_local_ptr, mmap_local_size);
```

2.1.4 read(2) system call

This system call currently supports ADC programmed I/O reads of channel registers. The option selected is determined by the *ccurDPRC_Select_Driver_Read_Mode()* call.

2.1.5 write(2) system call

Currently this option is not implemented. The option selected is determined by the *ccurDPRC_Select_Driver_Write_Mode()* call.

.

2.2 Application Program Interface (API) Access

The API is the recommended method of communicating with the board for most users.

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 CCURDPRC_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 are a list of calls that are available.

```
ccurDPRC_Abort_DMA()
ccurDPRC_Activate_Board()
ccurDPRC_ADC_Activate()
ccurDPRC_ADC_Get_Negative_Cal()
ccurDPRC_ADC_Get_Offset_Cal()
ccurDPRC_ADC_Get_Positive_Cal()
ccurDPRC_ADC_Perform_Auto_Calibration()
ccurDPRC_ADC_Perform_External_Negative_Calibration()
ccurDPRC_ADC_Perform_External_Offset_Calibration()
ccurDPRC_ADC_Perform_External_Positive_Calibration()
ccurDPRC_ADC_Perform_Negative_Calibration()
ccurDPRC_ADC_Perform_Offset_Calibration()
ccurDPRC_ADC_Perform_Positive_Calibration()
ccurDPRC_ADC_Read_Channels()
ccurDPRC_ADC_Read_Channels_Calibration()
ccurDPRC_ADC_Set_Negative_Cal()
ccurDPRC_ADC_Set_Offset_Cal()
ccurDPRC_ADC_Set_Positive_Cal()
ccurDPRC_ADC_Write_Channels_Calibration()
ccurDPRC_Add_Irq()
ccurDPRC_Clear_Driver_Error()
ccurDPRC_Clear_Electronic_Fuse_Trip_Fault()
ccurDPRC_Clear_Lib_Error()
ccurDPRC_Close()
ccurDPRC_DataToVolts()
ccurDPRC_DigitalPotentiometerAndIo_Activate()
ccurDPRC_Disable_Pci_Interrupts()
ccurDPRC_Enable_Pci_Interrupts()
ccurDPRC_Fast_Memcpy()
ccurDPRC_Fast_Memcpy_Unlocked()
ccurDPRC_Fraction_To_Hex()
ccurDPRC_Get_Board_CSR()
ccurDPRC_Get_Board_Info()
ccurDPRC_Get_CalibrationBus_Control()
ccurDPRC_Get_Digital_Potentiometer()
ccurDPRC_Get_Digital_Potentiometer_Test()
ccurDPRC_Get_Driver_Error()
ccurDPRC_Get_Driver_Info()
ccurDPRC_Get_Driver_Read_Mode()
```

```
ccurDPRC_Get_Driver_Write_Mode()
ccurDPRC_Get_Electronic_Fuse_Base()
ccurDPRC_Get_Electronic_Fuse_Internals()
ccurDPRC_Get_Electronic_Fuse_Multiplier()
ccurDPRC_Get_Electronic_Fuse_Status()
ccurDPRC_Get_Electronic_Fuse_Threshold()
ccurDPRC_Get_Electronic_Fuse_Trip()
ccurDPRC_Get_Interrupt_Control()
ccurDPRC_Get_Interrupt_Status()
ccurDPRC_Get_Interrupt_Timeout_Seconds()
ccurDPRC_Get_IO_Control()
ccurDPRC_Get_Lib_Error_Description()
ccurDPRC_Get_Lib_Error()
ccurDPRC_Get_Mapped_Config_Ptr()
ccurDPRC_Get_Mapped_Driver_Library_Ptr()
ccurDPRC_Get_Mapped_Local_Ptr()
ccurDPRC_Get_Open_File_Descriptor()
ccurDPRC_Get_Physical_Memory()
ccurDPRC_Get_Value()
ccurDPRC_Hex_To_Fraction()
ccurDPRC_Identify_Board()
ccurDPRC_Initialize_Board()
ccurDPRC_MMap_Physical_Memory()
ccurDPRC_Munmap_Physical_Memory()
ccurDPRC_NanoDelay()
ccurDPRC_Open()
ccurDPRC_Read()
ccurDPRC_Read_Serial_Prom()
ccurDPRC_Read_Serial_Prom_Item()
ccurDPRC_Remove_Irq()
ccurDPRC_Reset_Board()
ccurDPRC_Select_Driver_Read_Mode()
ccurDPRC_Select_Driver_Write_Mode()
ccurDPRC_Serial_Prom_Write_Override()
ccurDPRC_Set_Board_CSR()
ccurDPRC_Set_CalibrationBus_Control()
ccurDPRC_Set_Digital_Potentiometer()
ccurDPRC_Set_Digital_Potentiometer_Test()
ccurDPRC_Set_Interrupt_Control()
ccurDPRC_Set_Interrupt_Status()
ccurDPRC_Set_Interrupt_Timeout_Seconds()
ccurDPRC_Set_IO_Control()
ccurDPRC_Set_Value()
ccurDPRC_VoltsToData()
ccurDPRC_VoltsToDataChanCal()
ccurDPRC_Wait_For_Interrupt()
ccurDPRC_Write()
ccurDPRC_Write_Serial_Prom()
ccurDPRC_Write_Serial_Prom_Item()
```

2.2.1 ccurDPRC_Abort_DMA()

This call will abort any DMA operation that is in progress. Normally, the user should not use this call unless they are providing their own DMA handling.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t ccurDPRC_Abort_DMA(void *Handle)

   Description: Abort any DMA in progress

   Input:   void *Handle                        (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION   (local region not present)
               # CCURDPRC_LIB_IOCTL_FAILED      (driver ioctl call failed)
*************************************************************************/
```

2.2.2 ccurDPRC_Activate_Board()

This call activates the ADC, Potentiometer and I/O in a specific sequence to ensure proper board activation. This is the preferred and recommended call to activate the card. Using the individual ADC and Potentiometer activation calls, not being performed in correct sequence could result in unpredictable behavior of the card.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t ccurDPRC_Activate_Board(void *Handle)


   Description: Activate ADC, Potentiometer and I/O Control module

   Input:   void                        *Handle       (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (device not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
               # CCURDPRC_LIB_ADC_FAILURE            (ADC failure)
               # CCURDPRC_LIB_ELECTRONIC_FUSE_TRIPPED (Electronic Fuse tripped)
*************************************************************************/
```

2.2.3 ccurDPRC_ADC_Activate()

This call gives the user the ability to activate, disable and get the current ADC state. 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 CCURDPRC_ADC_ALL_ENABLE_DO_NOT_CHANGE. If the ADC is already active and the user issues a CCURDPRC_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 CCURDPRC_ADC_ALL_DISABLE followed by CCURDPRC_ADC_ALL_ENABLE.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Activate (void                        *Handle,
                          _ccurdprc_adc_all_enable_t activate,
                          _ccurdprc_adc_all_enable_t *current_state)

   Description: Activate/DeActivate ADC module
```

```
   Input:   void                          *Handle     (Handle pointer)
            _ccurdprc_adc_all_enable_t    activate    (activate/deactivate)
               # CCURDPRC_ADC_ALL_DISABLE
               # CCURDPRC_ADC_ALL_ENABLE
               # CCURDPRC_ADC_ALL_ENABLE_DO_NOT_CHANGE
   Output:  _ccurdprc_adc_all_enable_t       *current_state (active/deactive)
               # CCURDPRC_ADC_ALL_DISABLE
               # CCURDPRC_ADC_ALL_ENABLE
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
*****************************************************************************/
```

2.2.4 ccurDPRC_ADC_Get_Negative_Cal()

This call returns the negative ADC calibration information to the user.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Get_Negative_Cal(void               *Handle,
                                 ccurdprc_adc_cal_t *cal)

   Description: Get the ADC Negative Calibration data.

   Input:   void                        *Handle  (handle pointer)
   Output:  ccurdprc_adc_cal_t          *cal     (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION   (local region not present)
*****************************************************************************/
```

2.2.5 ccurDPRC_ADC_Get_Offset_Cal()

This call returns the offset ADC calibration information to the user.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Get_Offset_Cal(void               *Handle,
                               ccurdprc_adc_cal_t *cal)

   Description: Get the ADC Offset Calibration data.

   Input:   void                        *Handle  (handle pointer)
   Output:  ccurdprc_adc_cal_t          *cal     (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION   (local region not present)
  *****************************************************************************/
```

2.2.6 ccurDPRC_ADC_Get_Positive_Cal()

This call returns the positive ADC calibration information to the user.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Get_Positive_Cal(void               *Handle,
                                 ccurdprc_adc_cal_t *cal)


   Description: Get the ADC Positive Calibration data.

   Input:   void                       *Handle  (handle pointer)
   Output:  ccurdprc_adc_cal_t         *cal     (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION   (local region not present)
 *************************************************************************/
```

2.2.7 ccurDPRC_ADC_Perform_Auto_Calibration()

This call performs a full ADC calibration.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Perform_Auto_Calibration(void *Handle)

   Description: Perform ADC Auto Calibration

   Input:   void   *Handle                          (handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (library not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE           (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR              (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
               # CCURDPRC_LIB_ADC_FAILURE           (ADC failure)
               # CCURDPRC_LIB_ELECTRONIC_FUSE_TRIPPED
                                                    (Electronic Fuse tripped)
 *************************************************************************/
```

2.2.8 ccurDPRC_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.

```
/*************************************************************************
  _ccurdprc_lib_error_number_t
  ccurDPRC_ADC_Perform_External_Negative_Calibration(void       *Handle,
                                        _ccurdprc_channel_t    chan_start,
                                        _ccurdprc_channel_t    chan_end,
                                        double                 ReferenceVoltage)

   Description: Perform ADC External Negative Calibration
```

```
   Input:   void                    *Handle          (handle pointer)
            _ccurdprc_channel_t chan_start           (start channel)
            _ccurdprc_channel_t chan_end             (end channel)
            double                  ReferenceVoltage (Reference Voltage)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
              # CCURDPRC_LIB_NO_ERROR                (successful)
              # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
              # CCURDPRC_LIB_NOT_OPEN                (library not open)
              # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
              # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
              # CCURDPRC_LIB_NO_RESOURCE             (no free PLL available)
              # CCURDPRC_LIB_IO_ERROR                (read error)
              # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE     (Clock is not active)
*****************************************************************************/
```

2.2.9 ccurDPRC_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.

```
/*****************************************************************************
  _ccurdprc_lib_error_number_t
  ccurDPRC_ADC_Perform_External_Offset_Calibration(void         *Handle,
                                        _ccurdprc_channel_t    chan_start,
                                        _ccurdprc_channel_t    chan_end)

   Description: Perform ADC External Offset Calibration

   Input:   void                    *Handle         (handle pointer)
            _ccurdprc_channel_t chan_start          (start channel)
            _ccurdprc_channel_t chan_end            (end channel)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (library not open)
               # CCURDPRC_LIB_INVALID_ARG           (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE           (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR              (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
*****************************************************************************/
```

2.2.10 ccurDPRC_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.

```
/*****************************************************************************
  _ccurdprc_lib_error_number_t
  ccurDPRC_ADC_Perform_External_Positive_Calibration(void       *Handle,
                                        _ccurdprc_channel_t    chan_start,
                                        _ccurdprc_channel_t    chan_end,
                                        double                 ReferenceVoltage)

   Description: Perform ADC External Positive Calibration
```

```
   Input:   void                    *Handle          (handle pointer)
            _ccurdprc_channel_t chan_start           (start channel)
            _ccurdprc_channel_t chan_end             (end channel)
            double                  ReferenceVoltage (Reference Voltage)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (library not open)
               # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE            (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR               (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE    (Clock is not active)
*****************************************************************************/
```

2.2.11 ccurDPRC_ADC_Perform_Negative_Calibration()

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

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Perform_Negative_Calibration(void *Handle)

   Description: Perform ADC Negative Calibration

   Input:   void   *Handle                          (handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (library not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE           (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR              (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
*****************************************************************************/
```

2.2.12 ccurDPRC_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.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Perform_Offset_Calibration(void *Handle)

   Description: Perform ADC Offset Calibration

   Input:   void   *Handle                          (handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (library not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE           (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR              (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
*****************************************************************************/
```

2.2.13 ccurDPRC_ADC_Perform_Positive_Calibration()

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

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Perform_Positive_Calibration(void *Handle)

   Description: Perform ADC Positive Calibration

   Input:   void   *Handle                          (handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (library not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE           (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR              (read error)
               # CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE   (Clock is not active)
 *****************************************************************************/
```

2.2.14 ccurDPRC_ADC_Read_Channels()

This call provides the user an easy method of reading the ADC channels. User can supply a channel mask.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Read_Channels(void                      *Handle,
                              _ccurdprc_channel_mask_t ChanMask,
                              ccurdprc_adc_volts_t     *adc_volts)

   Description: Read ADC Channels

   Input:   void                         *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t     ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
               # CCURDPRC_CHANNEL_MASK_2
               # CCURDPRC_CHANNEL_MASK_3
               # CCURDPRC_CHANNEL_MASK_4
               # CCURDPRC_CHANNEL_MASK_5
               # CCURDPRC_CHANNEL_MASK_6
               # CCURDPRC_CHANNEL_MASK_7
               # CCURDPRC_CHANNEL_MASK_8
               # CCURDPRC_CHANNEL_MASK_9
               # CCURDPRC_CHANNEL_MASK_10
               # CCURDPRC_CHANNEL_MASK_11
               # CCURDPRC_CHANNEL_MASK_12
               # CCURDPRC_CHANNEL_MASK_13
               # CCURDPRC_CHANNEL_MASK_14
               # CCURDPRC_CHANNEL_MASK_15
               # CCURDPRC_ALL_CHANNELS_MASK
   Output:  ccurdprc_adc_volts_t                *adc_volts (pointer to ADC volts)
                  uint    Raw[CCURDPRC_MAX_CHANNELS];
                  double  Float[CCURDPRC_MAX_CHANNELS];
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
 *****************************************************************************/
```

2.2.15 ccurDPRC_ADC_Read_Channels_Calibration()

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

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Read_Channels_Calibration(void *Handle,
                                          char *filename)

   Description: Read ADC Channels Calibration

   Input:    void   *Handle                       (handle pointer)
   Output:   char   *filename                     (pointer to filename)
   Return:   _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR            (successful)
               # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN            (library not open)
               # CCURDPRC_LIB_NO_LOCAL_REGION     (local region not present)
               # CCURDPRC_LIB_CANNOT_OPEN_FILE    (cannot open calib. file)
 *************************************************************************/
```

2.2.16 ccurDPRC_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. Users can supply CCURDPRC_DO_NOT_CHANGE as a gain for any channel that should not be changed. Additionally, this call will return the *RAW* value of the gain supplied that is written to the board.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Set_Negative_Cal(void                *Handle,
                                 ccurdprc_adc_cal_t *cal)

   Description: Set the ADC Negative Calibration data.

   Input:    void                        *Handle       (handle pointer)
             ccurdprc_adc_cal_t          *cal          (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Output:   none
   Return:   _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (library not open)
               # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
               # CCURDPRC_LIB_NO_RESOURCE            (no free PLL available)
               # CCURDPRC_LIB_IO_ERROR               (read error)
 *************************************************************************/
```

2.2.17 ccurDPRC_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. Users can supply CCURDPRC_DO_NOT_CHANGE as a gain for any channel that should not be changed. Additionally, this call will return the *Raw* value of the offset supplied that is written to the board.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Set_Offset_Cal(void               *Handle,
                               ccurdprc_adc_cal_t *cal)
```

```
   Description: Set the ADC Offset Calibration data.

   Input:    void                       *Handle       (handle pointer)
             ccurdprc_adc_cal_t         *cal          (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Output:   none
   Return:   _ccurdprc_lib_error_number_t
                # CCURDPRC_LIB_NO_ERROR               (successful)
                # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
                # CCURDPRC_LIB_NOT_OPEN               (library not open)
                # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
                # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
                # CCURDPRC_LIB_NO_RESOURCE            (no free PLL available)
                # CCURDPRC_LIB_IO_ERROR               (read error)
 ****************************************************************************/
```

2.2.18 ccurDPRC_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. Users can supply CCURDPRC_DO_NOT_CHANGE as a gain for any channel that should not be changed. Additionally, this call will return the *Raw* value of the gain supplied that is written to the board.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Set_Positive_Cal(void               *Handle,
                                 ccurdprc_adc_cal_t *cal)

   Description: Set the ADC Positive Calibration data.

   Input:    void                       *Handle       (handle pointer)
             ccurdprc_adc_cal_t         *cal          (pointer to board cal)
               uint    Raw[CCURDPRC_MAX_CHANNELS];
               double  Float[CCURDPRC_MAX_CHANNELS];
   Output:   none
   Return:   _ccurdprc_lib_error_number_t
                # CCURDPRC_LIB_NO_ERROR               (successful)
                # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
                # CCURDPRC_LIB_NOT_OPEN               (library not open)
                # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
                # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
                # CCURDPRC_LIB_NO_RESOURCE            (no free PLL available)
                # CCURDPRC_LIB_IO_ERROR               (read error)
****************************************************************************/
```

2.2.19 ccurDPRC_ADC_Write_Channels_Calibration()

This call allows the user to write the calibration registers from a user supplied calibration file.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_ADC_Write_Channels_Calibration(void *Handle,
                                           char *filename)

   Description: Write Channels Calibration

   Input:    void   *Handle                        (handle pointer)
   Output:   char   *filename                      (pointer to filename)
   Return:   _ccurdprc_lib_error_number_t
                # CCURDPRC_LIB_NO_ERROR            (successful)
                # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
                # CCURDPRC_LIB_NOT_OPEN            (library not open)
```

```
                # CCURDPRC_LIB_NO_LOCAL_REGION    (local region not present)
                # CCURDPRC_LIB_CANNOT_OPEN_FILE   (cannot open calib. file)
                # CCURDPRC_LIB_INVALID_ARG        (invalid argument)
 ***************************************************************************/
```

2.2.20 ccurDPRC_Add_Irq()

This call will add the driver interrupt handler if it has not been added. Normally, the user should not use this call unless they want to disable the interrupt handler and then re-enable it.

```
/***************************************************************************
   int ccurDPRC_Add_Irq(void *Handle)

   Description: By default, the driver assigns an interrupt handler to handle
                device interrupts. If the interrupt handler was removed using
                the ccurDPRC_Remove_Irq(), then this call adds it back.

   Input:   void *Handle                        (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (library not open)
               # CCURDPRC_LIB_IOCTL_FAILED      (driver ioctl call failed)
****************************************************************************/
```

2.2.21 ccurDPRC_Clear_Driver_Error()

This call resets the last driver error that was maintained internally by the driver to *CCURDPRC_SUCCESS*.

```
/***************************************************************************
   _ccurdprc_lib_error_number_t ccurDPRC_Clear_Driver_Error(void *Handle)

   Description: Clear any previously generated driver related error.

   Input:   void *Handle                        (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_IOCTL_FAILED      (driver ioctl call failed)
****************************************************************************/
```

2.2.22 ccurDPRC_Clear_Electronic_Fuse_Trip_Fault()

If an electronic fuse trip fault occured for a channel, that channel is no longer accessible to the user until the channel fuse trip fault is reset. Users need to query as to why the electronic fuse trip fault occurred, correct the condition that caused the fuse trip fault, and then clear the fuse trip fault for the channel with the help of this call. If the reason for the fuse trip fault is not cleared, the fault is cleared, it is likely that the electronic fuse trip will re-occur immediately. If multiple electronic fuse trip faults occur for a channel, the faults will be queued to a quantity of two deep. This call ensures that all queued faults for the requested channel are also cleared.

```
/***************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Clear_Electronic_Fuse_Trip_Fault (void                      *Handle,
                                              _ccurdprc_channel_mask_t ChanMask)

   Description: Clear Electronic Fuse Trip Fault
```

```
   Input:       void                          *Handle   (handle pointer)
                _ccurdprc_channel_mask_t      ChanMask  (specify channel mask)
                   # CCURDPRC_CHANNEL_MASK_0
                   # CCURDPRC_CHANNEL_MASK_1
                   # CCURDPRC_CHANNEL_MASK_2
                   # CCURDPRC_CHANNEL_MASK_3
                   # CCURDPRC_CHANNEL_MASK_4
                   # CCURDPRC_CHANNEL_MASK_5
                   # CCURDPRC_CHANNEL_MASK_6
                   # CCURDPRC_CHANNEL_MASK_7
                   # CCURDPRC_CHANNEL_MASK_8
                   # CCURDPRC_CHANNEL_MASK_9
                   # CCURDPRC_CHANNEL_MASK_10
                   # CCURDPRC_CHANNEL_MASK_11
                   # CCURDPRC_CHANNEL_MASK_12
                   # CCURDPRC_CHANNEL_MASK_13
                   # CCURDPRC_CHANNEL_MASK_14
                   # CCURDPRC_CHANNEL_MASK_15
                   # CCURDPRC_ALL_CHANNELS_MASK
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                    # CCURDPRC_LIB_IO_ERROR             (Channel Trip Fault not
                                                         clearing)
 ****************************************************************************/
```

2.2.23 ccurDPRC_Clear_Lib_Error()

This call resets the last library error that was maintained internally by the API.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t ccurDPRC_Clear_Lib_Error(void *Handle)

   Description: Clear any previously generated library related error.

   Input:   void *Handle                        (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
****************************************************************************/
```

2.2.24 ccurDPRC_Close()

This call is used to close an already opened device using the *ccurDPRC_Open()* call.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t ccurDPRC_Close(void *Handle)

   Description: Close a previously opened device.

   Input:   void *Handle                    (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
 ****************************************************************************/
```

2.2.25 ccurDPRC_DataToVolts()

This routine takes a raw analog input data value and converts it to a floating point.

```
/*************************************************************************
   double ccurDPRC_DataToVolts(int us_data)

   Description: Convert Data to volts

   Input:   int                                us_data       (data to convert)
   Output:  none
   Return:  double                             volts         (returned volts)
 *************************************************************************/
```

2.2.26 ccurDPRC_DigitalPotentiometerAndIo_Activate()

This call gives the user the ability to activate, disable and get the current ADC state.

```
/*************************************************************************
   ccurDPRC_DigitalPotentiometerAndIo_Activate()
   _ccurdprc_lib_error_number_t
   ccurDPRC_DigitalPotentiometerAndIo_Activate (void              *Handle,
                          _ccurdprc_digital_pot_and_io_enable_t activate,
                          _ccurdprc_digital_pot_and_io_enable_t *current_state)

   Description: Activate/DeActivate Digital Potentiometer and I/O  module

   Input:   void                                  *Handle   (Handle pointer)
            _ccurdprc_digital_pot_and_io_enable_t activate  (activate/deactivate)
               # CCURDPRC_DIGITAL_POT_AND_IO_DISABLE
               # CCURDPRC_DIGITAL_POT_AND_IO_ENABLE
               # CCURDPRC_DIGITAL_POT_AND_IO_ENABLE_DO_NOT_CHANGE
   Output:  _ccurdprc_digital_pot_and_io_enable_t *current_state
                                                              (active/deactive)
               # CCURDPRC_DIGITAL_POT_AND_IO_DISABLE
               # CCURDPRC_DIGITAL_POT_AND_IO_ENABLE
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
  *************************************************************************/
```

2.2.27 ccurDPRC_Disable_Pci_Interrupts()

The purpose of this call is to disable PCI interrupts. This call shouldn't be used during normal reads as calls could time out. The driver handles enabling and disabling interrupts during its normal course of operation.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Disable_Pci_Interrupts (void *Handle)


   Description: Disable interrupts being generated by the board.

   Input:       void *Handle                     (handle pointer)
   Output:      None
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR       (successful)
                    # CCURDPRC_LIB_BAD_HANDLE     (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN       (device not open)
                    # CCURDPRC_LIB_IOCTL_FAILED   (driver ioctl call failed)
```

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

2.2.28 ccurDPRC_Enable_Pci_Interrupts()

The purpose of this call is to enable PCI interrupts. This call shouldn't be used during normal reads as calls could time out. The driver handles enabling and disabling interrupts during its normal course of operation.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Enable_Pci_Interrupts (void *Handle, uint interrupt_mask)

   Description: Enable interrupts being generated by the board.

   Input:   void         *Handle            (Handle pointer)
            uint         interrupt_mask     (interrupt mask)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
               # CCURDPRC_LIB_IOCTL_FAILED  (driver ioctl call failed)
 *****************************************************************************/
```

2.2.29 ccurDPRC_Fast_Memcpy()

The purpose of this call is to provide a fast mechanism to copy between hardware and memory using programmed I/O. The library performs appropriate locking while the copying is taking place.

```
/*****************************************************************************
   ccurDPRC_Fast_Memcpy(void           *Handle,
                        volatile void  *Destination,
                        volatile void  *Source,
                        int            SizeInBytes)

   Description: Perform fast copy to/from buffer using Programmed I/O
                (WITH LOCKING)

   Input:   void         *Handle            (Handle pointer)
            volatile void *Source           (pointer to source buffer)
            int            SizeInBytes      (transfer size in bytes)
   Oupput:  volatile void *Destination      (pointer to destination buffer)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
 *****************************************************************************/
```

2.2.30 ccurDPRC_Fast_Memcpy_Unlocked()

The purpose of this call is to provide a fast mechanism to copy between hardware and memory using programmed I/O. The library does not perform any locking. User needs to provide external locking instead.

```
/*****************************************************************************
   void
   ccurDPRC_Fast_Memcpy_Unlocked(volatile void *Destination,
                                 volatile void *Source,
                                 int           SizeInBytes)

   Description: Perform fast copy to/from buffer using Programmed I/O
                (WITHOUT LOCKING)

   Input:   volatile void *Source               (pointer to source buffer)
            int            SizeInBytes          (transfer size in bytes)
```

```
   Oupput:  volatile void *Destination          (pointer to destination buffer)
   Return:  none
*****************************************************************************/
```

2.2.31 ccurDPRC_Fraction_To_Hex()

This converts a fractional decimal to a hexadecimal value.

```
/*****************************************************************************
   int
   ccurDPRC_Fraction_To_Hex (double Fraction,
                             uint   *value)

   Description: Convert Fractional Decimal to Hexadecimal

   Input:       double   Fraction       (fraction to convert)
   Output:      uint     *value         (converted hexadecimal value)
   Return:      1                       (call failed)
                0                       (good return)
 *****************************************************************************/
```

2.2.32 ccurDPRC_Get_Board_CSR()

This call returns information from the board status register.

```
/*****************************************************************************
  _ccurdprc_lib_error_number_t
  ccurDPRC_Get_Board_CSR (void                    *Handle,
                          ccurdprc_board_csr_t    *bcsr)

   Description: Get Board Control and Status information

   Input:   void                                *Handle    (Handle pointer)
   Output:  ccurdprc_board_csr_t                *bcsr      (pointer to board csr)
               _ccurdprc_bcsr_identify_board_t  identify_board
                  # CCURDPRC_BCSR_IDENTIFY_BOARD_DISABLE
                  # CCURDPRC_BCSR_IDENTIFY_BOARD_ENABLE
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
 *****************************************************************************/
```

2.2.33 ccurDPRC_Get_Board_Info()

This call returns the board id, the board type and the firmware revision level for the selected board. This board id is *0x9300* and board type is *0x1*.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Board_Info (void                  *Handle,
                            ccurdprc_board_info_t *binfo)

   Description: Get Board Information

   Input:       void                              *Handle (handle pointer)
   Output:      ccurdprc_board_info_t             *binfo  (pointer to board info)
                   int     board_id                      (board id)
                   int     board_type                    (board type)
                   int     firmware_rev                  (firmware revision)
                   ccurdprc_sprom_header_t sprom_header
```

```
                     u_int32_t   board_serial_number     (serial number)
                     u_short     sprom_revision          (serial prom revision)
                  int     number_of_channels     (number of hardware channels)
                  int     all_channels_mask              (all channels mask)
                  double  cal_ref_voltage        (calibration reference voltage)
                  double  voltage_range                  (maximum voltage range)
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR              (successful)
                    # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler
supplied)
                    # CCURDPRC_LIB_NOT_OPEN              (device not open)
                    # CCURDPRC_LIB_INVALID_ARG           (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION     (local region not present)
*****************************************************************************/
```

2.2.34 ccurDPRC_Get_CalibrationBus_Control()

This call returns the bus calibration control.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_CalibrationBus_Control (void                          *Handle,
                               _ccurdprc_calibration_bus_control_t *bus_control)

   Description: Get Calibration Bus Control

   Input:       void                                  *Handle (handle pointer)
   Output:      _ccurdprc_calibration_bus_control_t
                                  *bus_control  (pointer to control select)
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_OPEN
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_2_5_VOLTS
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_10_VOLTS
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_MINUS_10_VOLTS

                    # CCURDPRC_CALIBRATIONBUS_CONTROL_GROUND

                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_8_MILLIAMP
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_MINUS_8_MILLIAMP
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_16_MILLIAMP
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR           (successful)
                    # CCURDPRC_LIB_BAD_HANDLE         (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN           (device not open)
                    # CCURDPRC_LIB_INVALID_ARG        (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION    (local region not present)
 *****************************************************************************/
```

2.2.35 ccurDPRC_Get_Digital_Potentiometer()

This call returns to the user the raw and ohms value of the digital potentiometer for the selected channels.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Digital_Potentiometer(void                            *Handle,
                                   _ccurdprc_channel_mask_t         ChanMask,
                                   ccurdprc_digital_potentiometer_t *DPValue)

   Description: Get Digital Potentiometer Value

   Input:   void                          *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t    ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
```

```
                # CCURDPRC_CHANNEL_MASK_2
                # CCURDPRC_CHANNEL_MASK_3
                # CCURDPRC_CHANNEL_MASK_4
                # CCURDPRC_CHANNEL_MASK_5
                # CCURDPRC_CHANNEL_MASK_6
                # CCURDPRC_CHANNEL_MASK_7
                # CCURDPRC_CHANNEL_MASK_8
                # CCURDPRC_CHANNEL_MASK_9
                # CCURDPRC_CHANNEL_MASK_10
                # CCURDPRC_CHANNEL_MASK_11
                # CCURDPRC_CHANNEL_MASK_12
                # CCURDPRC_CHANNEL_MASK_13
                # CCURDPRC_CHANNEL_MASK_14
                # CCURDPRC_CHANNEL_MASK_15
                # CCURDPRC_ALL_CHANNELS_MASK
   Output:  ccurdprc_digital_potentiometer_t  *DPValue (pointer to Digital
                                                        Potentiometer Value)
                  uint     Raw[CCURDPRC_MAX_CHANNELS];
                  uint    Ohms[CCURDPRC_MAX_CHANNELS];
   Return:  _ccurdprc_lib_error_number_t
                # CCURDPRC_LIB_NO_ERROR                (successful)
                # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
                # CCURDPRC_LIB_NOT_OPEN                (device not open)
                # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
                # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
                # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                      (Potentiometer and I/O Control not active)
*****************************************************************************/
```

2.2.36 ccurDPRC_Get_Digital_Potentiometer_Test()

This call returns to the user the power-down and mode selection of the digital potentiometer for the selected channels.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Digital_Potentiometer_Test(void                          *Handle,
                                _ccurdprc_channel_mask_t                ChanMask,
                                ccurdprc_digital_potentiometer_test_t   *DPTest)

   Description: Get Digital Potentiometer Test

   Input:   void                        *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t    ChanMask      (specify channel mask)
                # CCURDPRC_CHANNEL_MASK_0
                # CCURDPRC_CHANNEL_MASK_1
                # CCURDPRC_CHANNEL_MASK_2
                # CCURDPRC_CHANNEL_MASK_3
                # CCURDPRC_CHANNEL_MASK_4
                # CCURDPRC_CHANNEL_MASK_5
                # CCURDPRC_CHANNEL_MASK_6
                # CCURDPRC_CHANNEL_MASK_7
                # CCURDPRC_CHANNEL_MASK_8
                # CCURDPRC_CHANNEL_MASK_9
                # CCURDPRC_CHANNEL_MASK_10
                # CCURDPRC_CHANNEL_MASK_11
                # CCURDPRC_CHANNEL_MASK_12
                # CCURDPRC_CHANNEL_MASK_13
                # CCURDPRC_CHANNEL_MASK_14
                # CCURDPRC_CHANNEL_MASK_15
                # CCURDPRC_ALL_CHANNELS_MASK
   Output:  ccurdprc_digital_potentiometer_test_t *DPTest (pointer to
```

```
                                                Digital Potentiometer Test)
                  _ccurdprc_digital_potentiometer_test_t
                                 DigitalPotTest[CCURDPRC_MAX_CHANNELS];
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT0
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT1
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT2
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT3
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_100K
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_FORCE_FAILURE
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_MODE_20K
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_MODE_100K
   Return:  _ccurdprc_lib_error_number_t
                     # CCURDPRC_LIB_NO_ERROR             (successful)
                     # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                     # CCURDPRC_LIB_NOT_OPEN             (device not open)
                     # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                     # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                     # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                    (Potentiometer and I/O Control not active)
 ****************************************************************************/
```

2.2.37 ccurDPRC_Get_Driver_Error()

This call returns the last error generated by the driver.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Driver_Error (void                    *Handle,
                              ccurdprc_user_error_t   *ret_err)

   Description: Get the last error generated by the driver.

   Input:   void                  *Handle             (Handle pointer)
   Output:  ccurdprc_user_error_t *ret_err            (error struct pointer)
               uint error                             (error number)
               char name[CCURDPRC_ERROR_NAME_SIZE]  (error name used in driver)
               char desc[CCURDPRC_ERROR_DESC_SIZE]  (error description)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (device not open)
               # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
               # CCURDPRC_LIB_IOCTL_FAILED           (driver ioctl call failed)
 ****************************************************************************/

#define CCURDPRC_ERROR_NAME_SIZE    64
#define CCURDPRC_ERROR_DESC_SIZE    128

typedef struct _ccurdprc_user_error_t
{
    uint error;                              /* error number */
    char name[CCURDPRC_ERROR_NAME_SIZE];     /* error name used in driver */
    char desc[CCURDPRC_ERROR_DESC_SIZE];     /* error description */
} ccurdprc_user_error_t;

enum
{
    CCURDPRC_SUCCESS = 0,
    CCURDPRC_INVALID_PARAMETER,
    CCURDPRC_DMA_TIMEOUT,
    CCURDPRC_OPERATION_CANCELLED,
    CCURDPRC_RESOURCE_ALLOCATION_ERROR,
    CCURDPRC_INVALID_REQUEST,
```

```
    CCURDPRC_FAULT_ERROR,
    CCURDPRC_BUSY,
    CCURDPRC_ADDRESS_IN_USE,
    CCURDPRC_USER_INTERRUPT_TIMEOUT,
    CCURDPRC_DMA_INCOMPLETE,
    CCURDPRC_DATA_UNDERFLOW,
    CCURDPRC_DATA_OVERFLOW,
    CCURDPRC_IO_FAILURE,
    CCURDPRC_PCI_ABORT_INTERRUPT_ACTIVE,
};
```

2.2.38 ccurDPRC_Get_Driver_Info()

This call returns internal information that is maintained by the driver.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Driver_Info (void                    *Handle,
                             ccurdprc_driver_info_t *info)

   Description: Get device information from driver.

   Input:       void                            *Handle (handle pointer)

   Output:      ccurdprc_driver_info_t          *info   (info struct pointer)
                   char                    version[12]
                   char                    built[32]
                   char                    module_name[16]
                   int                     board_index
                   char                    board_desc[32]
                   int                     bus
                   int                     slot
                   int                     func
                   int                     vendor_id
                   int                     sub_vendor_id
                   int                     board_id
                   int                     board_type
                   int                     sub_device_id
                   int                     board_info
                   int                     msi_support
                   int                     irqlevel
                   int                     firmware
                   int                     number_of_channels
                   int                     all_channels_mask
                   double                  cal_ref_voltage
                   int                     max_dma_samples
                   int                     dma_size
                   double                  voltage_range
                   ccurdprc_driver_int_t interrupt
                       unsigned long long count
                       u_int status
                       u_int mask
                       int timeout_seconds
                   int                     Ccurdprc_Max_Region
                   ccurdprc_dev_region_t mem_region[CCURDPRC_MAX_REGION]
                      uint physical_address
                      uint size
                      uint flags
                      uint *virtual_address
                   ccurdprc_sprom_header_t sprom_header
                      u_int32_t   board_serial_number
                      u_short     sprom_revision
   Return:      _ccurdprc_lib_error_number_t
```

```
                    # CCURDPRC_LIB_NO_ERROR            (successful)
                    # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN            (device not open)
                    # CCURDPRC_LIB_INVALID_ARG         (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION     (local region error)
                    # CCURDPRC_LIB_IOCTL_FAILED        (driver ioctl call failed)
*****************************************************************************/
```

2.2.39 ccurDPRC_Get_Driver_Read_Mode()

This call returns the driver read mode.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Driver_Read_Mode (void                        *Handle,
                                  _ccurdprc_driver_rw_mode_t *mode)

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

   Input:       void                           *Handle  (handle pointer)
   Output:      _ccurdprc_driver_rw_mode_t     *mode    (pointer to read mode)
                    # CCURDPRC_PIO_CHANNEL
                    # CCURDPRC_DMA_CHANNEL
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR            (successful)
                    # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN            (device not open)
                    # CCURDPRC_LIB_INVALID_ARG         (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION     (local region error)
                    # CCURDPRC_LIB_IOCTL_FAILED        (driver ioctl call failed)
*****************************************************************************/
```

2.2.40 ccurDPRC_Get_Driver_Write_Mode()

This call is currently not supported for driver writes.This call returns the driver write mode.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Driver_Write_Mode (void                        *Handle,
                                   _ccurdprc_driver_rw_mode_t *mode)


   Description: Get current write mode that will be selected by the 'write()'
                call

   Input:       void                           *Handle  (handle pointer)
   Output:      _ccurdprc_driver_rw_mode_t     *mode    (pointer to write mode)
                    # CCURDPRC_PIO_CHANNEL
                    # CCURDPRC_DMA_CHANNEL
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR            (successful)
                    # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN            (device not open)
                    # CCURDPRC_LIB_INVALID_ARG         (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION     (local region error)
                    # CCURDPRC_LIB_IOCTL_FAILED        (driver ioctl call failed)
*****************************************************************************/
```

2.2.41 ccurDPRC_Get_Electronic_Fuse_Base()

This call returns the Electronic Fuse Base for the selected channels. This value where the channel will fault for a short (10 ohm) resistance. This is for information only and must not be changed by the user, otherwise, it could result in damage to the board.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Base (void                              *Handle,
                                      _ccurdprc_channel_mask_t        ChanMask,
                                      ccurdprc_electronic_fuse_base_t Base)

   Description: Get Electronic Fuse Base information

   Input:       void                          *Handle   (handle pointer)
                _ccurdprc_channel_mask_t      ChanMask  (specify channel mask)
                   # CCURDPRC_CHANNEL_MASK_0
                   # CCURDPRC_CHANNEL_MASK_1
                   # CCURDPRC_CHANNEL_MASK_2
                   # CCURDPRC_CHANNEL_MASK_3
                   # CCURDPRC_CHANNEL_MASK_4
                   # CCURDPRC_CHANNEL_MASK_5
                   # CCURDPRC_CHANNEL_MASK_6
                   # CCURDPRC_CHANNEL_MASK_7
                   # CCURDPRC_CHANNEL_MASK_8
                   # CCURDPRC_CHANNEL_MASK_9
                   # CCURDPRC_CHANNEL_MASK_10
                   # CCURDPRC_CHANNEL_MASK_11
                   # CCURDPRC_CHANNEL_MASK_12
                   # CCURDPRC_CHANNEL_MASK_13
                   # CCURDPRC_CHANNEL_MASK_14
                   # CCURDPRC_CHANNEL_MASK_15
                   # CCURDPRC_ALL_CHANNELS_MASK
   Output:      ccurdprc_electronic_fuse_base_t  Base[CCURDPRC_MAX_CHANNELS]
                                  (pointer to electronic fuse base channel array)
                   _ccurdprc_electronic_fuse_base_t
                      int     base_raw;
                      double  base_volts;
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 *****************************************************************************/
```

2.2.42 ccurDPRC_Get_Electronic_Fuse_Internals()

This call returns the internal settings for the electronic fuse trip for the selected channels. This is for information only and must not be changed by the user, otherwise it could result in damage to the board.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Internals (void                   *Handle,
                           ccurdprc_electronic_fuse_internals_t *ElectronicFuse)

   Description: Get Electronic Fuse Internals information

   Input:       void                     *Handle   (handle pointer)
   Output:      ccurdprc_electronic_fuse_internals_t   ElectronicFuse (pointer to
electronic fuse internals)
                   int     electrical_short_raw;
                   double  electrical_short_volts;
                   int     delay;
                   int     count;
                   int     io_delay_raw;
                   double  io_delay_microseconds;
                   int     maximum_resistance_raw;
```

```
                  double  maximum_resistance_ohms;
                  int     maximum_voltage_raw;
                  double  maximum_voltage_volts;
                  int     voltage_fault_delay_raw;
                  int     voltage_fault_delay_microseconds;
   Return:      _ccurdprc_lib_error_number_t
                   # CCURDPRC_LIB_NO_ERROR             (successful)
                   # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                   # CCURDPRC_LIB_NOT_OPEN             (device not open)
                   # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                   # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 ************************************************************************/
```

2.2.43 ccurDPRC_Get_Electronic_Fuse_Multiplier()

This call returns the Electronic Fuse Multiplier for the selected channels. This is the value where the channel will fault for a resistance other than a short (10 ohm) resistance. This is for information only and must not be changed by the user, otherwise, it could result in damage to the board.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Multiplier (void                       *Handle,
                                _ccurdprc_channel_mask_t               ChanMask,
                               ccurdprc_electronic_fuse_multiplier_t Multiplier)

   Description: Get Electronic Fuse Multiplier information

   Input:       void                          *Handle   (handle pointer)
                _ccurdprc_channel_mask_t      ChanMask  (specify channel mask)
                   # CCURDPRC_CHANNEL_MASK_0
                   # CCURDPRC_CHANNEL_MASK_1
                   # CCURDPRC_CHANNEL_MASK_2
                   # CCURDPRC_CHANNEL_MASK_3
                   # CCURDPRC_CHANNEL_MASK_4
                   # CCURDPRC_CHANNEL_MASK_5
                   # CCURDPRC_CHANNEL_MASK_6
                   # CCURDPRC_CHANNEL_MASK_7
                   # CCURDPRC_CHANNEL_MASK_8
                   # CCURDPRC_CHANNEL_MASK_9
                   # CCURDPRC_CHANNEL_MASK_10
                   # CCURDPRC_CHANNEL_MASK_11
                   # CCURDPRC_CHANNEL_MASK_12
                   # CCURDPRC_CHANNEL_MASK_13
                   # CCURDPRC_CHANNEL_MASK_14
                   # CCURDPRC_CHANNEL_MASK_15
                   # CCURDPRC_ALL_CHANNELS_MASK
   Output:      ccurdprc_electronic_fuse_multiplier_t
Multiplier[CCURDPRC_MAX_CHANNELS]
                                                 (pointer to electronic fuse
multiplier channel array)
                  _ccurdprc_electronic_fuse_multiplier_t
                     int     multiplier_raw;
                     double  multiplier_volts;
   Return:      _ccurdprc_lib_error_number_t
                   # CCURDPRC_LIB_NO_ERROR             (successful)
                   # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                   # CCURDPRC_LIB_NOT_OPEN             (device not open)
                   # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                   # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 ************************************************************************/
```

2.2.44 ccurDPRC_Get_Electronic_Fuse_Status()

This call returns the various Electronic Fuse Trip Status for the board.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Status (void                          *Handle,
                                        ccurdprc_electronic_fuse_status_t *Status)

   Description: Get Electronic Fuse Status information

   Input:       void                                  *Handle (handle pointer)
   Output:      ccurdprc_electronic_fuse_status_t   Status  (pointer to
electronic fuse status)
               _ccurdprc_efstat_fuse_tripped_t          any_fuse_tripped
                      # CCURDPRC_EFSTAT_FUSE_NOT_TRIPPED
                      # CCURDPRC_EFSTAT_FUSE_TRIPPED
                   _ccurdprc_efstat_fuse_adc_failed_t       adc_1_fuse_failed
                      # CCURDPRC_EFSTAT_FUSE_ADC_NOT_FAILED
                      # CCURDPRC_EFSTAT_FUSE_ADC_FAILED
                   _ccurdprc_efstat_fuse_adc_failed_t       adc_0_fuse_failed
                      # CCURDPRC_EFSTAT_FUSE_ADC_NOT_FAILED
                      # CCURDPRC_EFSTAT_FUSE_ADC_FAILED
                    _ccurdprc_efstat_fuse_tripped_t
channel_fuse_tripped[CCURDPRC_MAX_CHANNELS]
                      # CCURDPRC_EFSTAT_FUSE_NOT_TRIPPED
                      # CCURDPRC_EFSTAT_FUSE_TRIPPED
                   int
channel_fuse_tripped_mask
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 ************************************************************************/
```

2.2.45 ccurDPRC_Get_Electronic_Fuse_Threshold()

This call returns the Electronic Fuse Threshold for selected channels. This is for information only and must not be changed by the user, otherwise, it could result in damage to the board.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Threshold (void                          *Handle,
                                  _ccurdprc_channel_mask_t            ChanMask,
                                ccurdprc_electronic_fuse_threshold_t Threshold)

   Description: Get Electronic Fuse Threshold information

   Input:       void                          *Handle   (handle pointer)
                _ccurdprc_channel_mask_t    ChanMask  (specify channel mask)
                   # CCURDPRC_CHANNEL_MASK_0
                   # CCURDPRC_CHANNEL_MASK_1
                   # CCURDPRC_CHANNEL_MASK_2
                   # CCURDPRC_CHANNEL_MASK_3
                   # CCURDPRC_CHANNEL_MASK_4
                   # CCURDPRC_CHANNEL_MASK_5
                   # CCURDPRC_CHANNEL_MASK_6
                   # CCURDPRC_CHANNEL_MASK_7
                   # CCURDPRC_CHANNEL_MASK_8
                   # CCURDPRC_CHANNEL_MASK_9
                   # CCURDPRC_CHANNEL_MASK_10
```

```
                  # CCURDPRC_CHANNEL_MASK_11
                  # CCURDPRC_CHANNEL_MASK_12
                  # CCURDPRC_CHANNEL_MASK_13
                  # CCURDPRC_CHANNEL_MASK_14
                  # CCURDPRC_CHANNEL_MASK_15
                  # CCURDPRC_ALL_CHANNELS_MASK
   Output:      ccurdprc_electronic_fuse_threshold_t
Threshold[CCURDPRC_MAX_CHANNELS]
                                          (pointer to electronic fuse
threshold channel array)
                  _ccurdprc_electronic_fuse_threshold_t
                     int     threshold_raw;
                     double  threshold_volts;
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 ***************************************************************************/
```

2.2.46 ccurDPRC_Get_Electronic_Fuse_Trip()

This call returns the Electronic Fuse Trip for selected channels. This is for information only and must not be changed by the user, otherwise, it could result in damage to the board.

```
/***************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Electronic_Fuse_Trip (void                                *Handle,
                              _ccurdprc_channel_mask_t                 ChanMask,
                              ccurdprc_electronic_fuse_channel_trip_t Trip)

   Description: Get Electronic Fuse Trip information

   Input:       void                         *Handle   (handle pointer)
                _ccurdprc_channel_mask_t     ChanMask  (specify channel mask)
                   # CCURDPRC_CHANNEL_MASK_0
                   # CCURDPRC_CHANNEL_MASK_1
                   # CCURDPRC_CHANNEL_MASK_2
                   # CCURDPRC_CHANNEL_MASK_3
                   # CCURDPRC_CHANNEL_MASK_4
                   # CCURDPRC_CHANNEL_MASK_5
                   # CCURDPRC_CHANNEL_MASK_6
                   # CCURDPRC_CHANNEL_MASK_7
                   # CCURDPRC_CHANNEL_MASK_8
                   # CCURDPRC_CHANNEL_MASK_9
                   # CCURDPRC_CHANNEL_MASK_10
                   # CCURDPRC_CHANNEL_MASK_11
                   # CCURDPRC_CHANNEL_MASK_12
                   # CCURDPRC_CHANNEL_MASK_13
                   # CCURDPRC_CHANNEL_MASK_14
                   # CCURDPRC_CHANNEL_MASK_15
                   # CCURDPRC_ALL_CHANNELS_MASK
   Output:    ccurdprc_electronic_fuse_channel_trip_t Trip[CCURDPRC_MAX_CHANNELS]
                                (pointer to electronic fuse trip channel array)
                 _ccurdprc_electronic_fuse_channel_trip_t
                    _ccurdprc_eft_fuse_tripped_t                fuse_trip_status
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
                    _ccurdprc_eft_invalid_calibration_fuse_trip_t
                                                    invalid_calibration_trip
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
```

```
                    _ccurdprc_eft_potentiometer_failure_fuse_trip_t
                                                    potentiometer_failure_trip
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
                    _ccurdprc_eft_adc_failure_fuse_trip_t        adc_failure_trip
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
                    _ccurdprc_eft_switch_voltage_fuse_trip_t   witch_voltage_trip
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
                    _ccurdprc_eft_adc_compare_fuse_trip_t        adc_compare_trip
                       # CCURDPRC_EFT_FUSE_NOT_TRIPPED
                       # CCURDPRC_EFT_FUSE_TRIPPED
                      uint                                    last_adc_tripped_raw
                      double                                 last_adc_tripped_volts
   Return:       _ccurdprc_lib_error_number_t
                     # CCURDPRC_LIB_NO_ERROR             (successful)
                     # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                     # CCURDPRC_LIB_NOT_OPEN             (device not open)
                     # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                     # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 *************************************************************************/
```

2.2.47 ccurDPRC_Get_Interrupt_Control()

This call returns the interrupt control information.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Interrupt_Control (void                 *Handle,
                                   ccurdprc_interrupt_t *intr)

   Description: Get Interrupt Control information

   Input:       void                       *Handle  (handle pointer)
   Output:      ccurdprc_interrupt_t       *intr    (pointer to interrupt control)
                   int global_int
                   int plx_local_int
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR        (successful)
                    # CCURDPRC_LIB_BAD_HANDLE      (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN        (device not open)
                    # CCURDPRC_LIB_INVALID_ARG     (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION (local region error)
 *************************************************************************/
```

2.2.48 ccurDPRC_Get_Interrupt_Status()

This call returns the current status of the PLX interrupt.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Interrupt_Status (void                 *Handle,
                                  ccurdprc_interrupt_t *intr)

   Description: Get Interrupt Status information

   Input:       void                        *Handle  (handle pointer)
   Output:      ccurdprc_interrupt_t        *intr    (pointer to interrupt status)
                   int plx_local_int
                       # CCURDPRC_ISR_LOCAL_PLX_NONE
                       # CCURDPRC_ISR_LOCAL_PLX_OCCURRED
   Return:      _ccurdprc_lib_error_number_t
```

```
                    # CCURDPRC_LIB_NO_ERROR         (successful)
                    # CCURDPRC_LIB_BAD_HANDLE       (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN         (device not open)
                    # CCURDPRC_LIB_INVALID_ARG      (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION  (local region error)
 ************************************************************************/
```

2.2.49 ccurDPRC_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.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Interrupt_Timeout_Seconds (void    *Handle,
                                           int     *int_timeout_secs)

   Description: Get Interrupt Timeout Seconds

   Input:   void            *Handle              (Handle pointer)
   Output:  int             *int_timeout_secs    (pointer to int tout secs)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR           (successful)
               # CCURDPRC_LIB_BAD_HANDLE         (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN           (device not open)
               # CCURDPRC_LIB_INVALID_ARG        (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION    (local region not present)
               # CCURDPRC_LIB_IOCTL_FAILED       (ioctl error)
 ************************************************************************/
```

2.2.50 ccurDPRC_Get_IO_Control()

This call returns the IO Control settomg for the selected channels.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_IO_Control(void                       *Handle,
                           _ccurdprc_channel_mask_t ChanMask,
                           ccurdprc_io_control_t    *IoControl)

   Description: Get I/O Control

   Input:   void                          *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t      ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
               # CCURDPRC_CHANNEL_MASK_2
               # CCURDPRC_CHANNEL_MASK_3
               # CCURDPRC_CHANNEL_MASK_4
               # CCURDPRC_CHANNEL_MASK_5
               # CCURDPRC_CHANNEL_MASK_6
               # CCURDPRC_CHANNEL_MASK_7
               # CCURDPRC_CHANNEL_MASK_8
               # CCURDPRC_CHANNEL_MASK_9
               # CCURDPRC_CHANNEL_MASK_10
               # CCURDPRC_CHANNEL_MASK_11
               # CCURDPRC_CHANNEL_MASK_12
               # CCURDPRC_CHANNEL_MASK_13
               # CCURDPRC_CHANNEL_MASK_14
               # CCURDPRC_CHANNEL_MASK_15
               # CCURDPRC_ALL_CHANNELS_MASK
```

```
   Output:  ccurdprc_io_control_t                    *IoControl (pointer to I/O
control)
                  _ccurdprc_io_control_t
IoSignal[CCURDPRC_MAX_CHANNELS];
                     # CCURDPRC_IO_CONTROL_OPEN
                     # CCURDPRC_IO_CONTROL_EXTERNAL
                     # CCURDPRC_IO_CONTROL_TEST_BUS
                     # CCURDPRC_IO_CONTROL_RA_GROUND_FAULT
                     # CCURDPRC_IO_CONTROL_RB_GROUND_FAULT
                     # CCURDPRC_IO_CONTROL_RA_RB_GROUND_FAULT
                     # CCURDPRC_IO_CONTROL_RA_V_PLUS_FAULT
                     # CCURDPRC_IO_CONTROL_RB_V_PLUS_FAULT
                     # CCURDPRC_IO_CONTROL_RA_RB_V_PLUS_FAULT
                     # CCURDPRC_IO_CONTROL_RA_V_PLUS_RB_GROUND_FAULT_SWITCH
                     # CCURDPRC_IO_CONTROL_RA_GROUND_RB_V_PLUS_FAULT_SWITCH
   Return:  _ccurdprc_lib_error_number_t
                     # CCURDPRC_LIB_NO_ERROR             (successful)
                     # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                     # CCURDPRC_LIB_NOT_OPEN             (device not open)
                     # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                     # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                     # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                    (Potentiometer and I/O Control not active)
 ****************************************************************************/
```

2.2.51 ccurDPRC_Get_Lib_Error_Description()

This call returns the library error name and description for the supplied error number.

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

   ccurDPRC_Get_Lib_Error_Description()

   Description: Get Error Description of supplied error number.

   Input:   int                                ErrorNumber     (Library error number)
   Output:  ccurdprc_lib_error_description_t   *lib_error_desc (error description struct pointer)
              -- int  found
              -- char name[CCURDPRC_LIB_ERROR_NAME_SIZE] (last library error name)
              -- char desc[CCURDPRC_LIB_ERROR_DESC_SIZE] (last library error description)
   Return:  none

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

2.2.52 ccurDPRC_Get_Lib_Error()

This call provides detailed information about the last library error that was maintained by the API.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Lib_Error (void                    *Handle,
                           ccurdprc_lib_error_t    *lib_error)

   Description: Get last error generated by the library.

   Input:   void                 *Handle                  (Handle pointer)
   Output:  ccurdprc_lib_error_t *lib_error               (error struct pointer)
              -- uint error                               (last library error number)
              -- char name[CCURDPRC_LIB_ERROR_NAME_SIZE]  (last library error name)
              -- char desc[CCURDPRC_LIB_ERROR_DESC_SIZE]  (last library error
                                                           description)
              -- int  line_number                         (last library error line number
                                                           in lib)
```

```
                -- char function[CCURDPRC_LIB_ERROR_FUNC_SIZE]
                                                        (library function in error)
                -- ccurdprc_lib_error_backtrace_t BT[CCURDPRC_BACK_TRACE_DEPTH]
                                                        (backtrace of errors)
                   -- int  line_number                  (line number in library)
                   -- char function[CCURDPRC_LIB_ERROR_FUNC_SIZE]
                                                        (library function)
   Return:  _ccurdprc_lib_error_number_t
                   # CCURDPRC_LIB_NO_ERROR             (successful)
                   # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                   # CCURDPRC_LIB_NOT_OPEN             (device not open)
 ************************************************************************/
typedef struct
{
    int  line_number;                           /* line number in library */
    char function[CCURDPRC_LIB_ERROR_FUNC_SIZE]; /* library function */
} ccurdprc_lib_error_backtrace_t;

typedef struct
{
    uint error;                                 /* last library error number */
    char name[CCURDPRC_LIB_ERROR_NAME_SIZE];    /* last library error name */
    char desc[CCURDPRC_LIB_ERROR_DESC_SIZE];    /* last libarary error description */
    int  line_number;                           /* last library  error line number in
                                                   lib */
    char function[CCURDPRC_LIB_ERROR_FUNC_SIZE]; /* library function in error */
    ccurdprc_lib_error_backtrace_t BT[CCURDPRC_BACK_TRACE_DEPTH];
                                                /* backtrace of errors */
} ccurdprc_lib_error_t;
```

Possible library errors:

```
CCURDPRC_LIB_NO_ERROR                   =  0,  /* successful */
CCURDPRC_LIB_INVALID_ARG                = -1,  /* invalid argument */
CCURDPRC_LIB_ALREADY_OPEN               = -2,  /* already open */
CCURDPRC_LIB_OPEN_FAILED                = -3,  /* open failed */
CCURDPRC_LIB_BAD_HANDLE                 = -4,  /* bad handle */
CCURDPRC_LIB_NOT_OPEN                   = -5,  /* device not opened */
CCURDPRC_LIB_MMAP_SELECT_FAILED         = -6,  /* mmap selection failed */
CCURDPRC_LIB_MMAP_FAILED                = -7,  /* mmap failed */
CCURDPRC_LIB_MUNMAP_FAILED              = -8,  /* munmap failed */
CCURDPRC_LIB_NOT_MAPPED                 = -9,  /* not mapped */
CCURDPRC_LIB_ALREADY_MAPPED             = -10, /* already mapped */
CCURDPRC_LIB_IOCTL_FAILED               = -11, /* driver ioctl failed */
CCURDPRC_LIB_IO_ERROR                   = -12, /* i/o error */
CCURDPRC_LIB_INTERNAL_ERROR             = -13, /* internal library error */
CCURDPRC_LIB_NOT_IMPLEMENTED            = -14, /* call not implemented */
CCURDPRC_LIB_LOCK_FAILED                = -15, /* failed to get lib lock */
CCURDPRC_LIB_NO_LOCAL_REGION            = -16, /* local region not present */
CCURDPRC_LIB_NO_CONFIG_REGION           = -17, /* config region not present */
CCURDPRC_LIB_NO_SOLUTION_FOUND          = -18, /* no solution found */
CCURDPRC_LIB_CONVERTER_RESET            = -19, /* converter not active */
CCURDPRC_LIB_NO_RESOURCE                = -20, /* resource not available */
CCURDPRC_LIB_CALIBRATION_RANGE_ERROR    = -21, /* calibration voltage out of range */
CCURDPRC_LIB_CANNOT_OPEN_FILE           = -23, /* cannot open file */
CCURDPRC_LIB_BAD_DATA_IN_CAL_FILE       = -24, /* bad date in calibration file */
CCURDPRC_LIB_UNUSED_25                  = -25, /* UNUSED */
CCURDPRC_LIB_SERIAL_PROM_BUSY           = -26, /* serial prom busy */
CCURDPRC_LIB_SERIAL_PROM_FAILURE        = -27, /* serial prom failure - malfunction
occurred */
CCURDPRC_LIB_INVALID_CRC                = -28, /* invalid CRC read */
CCURDPRC_LIB_SERIAL_PROM_WRITE_PROTECTED = -29, /* serial prom is write protected */
CCURDPRC_LIB_ADC_IS_NOT_ACTIVE          = -30, /* ADC is not active */
```

```
CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                        = -31, /* Digital Potentiometer & I/O is not
                                                  active */
CCURDPRC_LIB_CLOCK_IS_NOT_ACTIVE        =  32, /* ADC Clock is not active */
CCURDPRC_LIB_ADC_FAILURE                = -33, /* ADC Failure */
CCURDPRC_LIB_ELECTRONIC_FUSE_TRIPPED    = -34, /* Electronic Fuse Tripped */
```

2.2.53 ccurDPRC_Get_Mapped_Config_Ptr()

If the user wishes to bypass the API and communicate directly with the board configuration registers, then they can use this call to acquire a pointer to these registers. Please note that any type of access (read or write) by bypassing the API could compromise the API and results could be unpredictable. It is recommended that only advanced users should use this call and with extreme care and intimate knowledge of the hardware programming registers before attempting to access these registers. For information on the registers, refer to the *ccurdprc_user.h* include file that is supplied with the driver.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Mapped_Config_Ptr (void                            *Handle,
                                   ccurdprc_config_local_data_t    **config_ptr)

   Description: Get mapped configuration pointer.

   Input:   void                           *Handle         (Handle pointer)
   Output:  ccurdprc_config_local_data_t   **config_ptr    (config struct ptr)
               -- structure in ccurdprc_user.h
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (device not open)
               # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
               # CCURDPRC_LIB_NO_CONFIG_REGION       (config region not present)
************************************************************************/
```

2.2.54 ccurDPRC_Get_Mapped_Driver_Library_Ptr()

The driver and library share a common structure. This call returns a pointer to the shared driver/library structure.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Mapped_Driver_Library_Ptr (void                *Handle,
                            ccurdprc_driver_library_common_t   **driver_lib_ptr)

   Description: Get mapped Driver/Library structure pointer.

   Input:   void                              *Handle  (Handle pointer)
   Output:  ccurdprc_driver_library_common_t  **driver_lib_ptr
                                                       (driver_lib struct ptr)
               uint dma_abort_count                    (DMA abort count)
               ccurdprc_sprom_header_t sprom_header
                  u_int32_t   board_serial_number      (serial number)
                  u_short     sprom_revision           (serial revision)
               uint library_needs_initialization
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR               (successful)
               # CCURDPRC_LIB_BAD_HANDLE             (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN               (device not open)
               # CCURDPRC_LIB_INVALID_ARG            (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION        (local region not present)
************************************************************************/
```

2.2.55 ccurDPRC_Get_Mapped_Local_Ptr()

If the user wishes to bypass the API and communicate directly with the board control and data registers, then they can use this call to acquire a pointer to these registers. Please note that any type of access (read or write) by bypassing the API could compromise the API and results could be unpredictable. It is recommended that only advanced users should use this call and with extreme care and intimate knowledge of the hardware programming registers before attempting to access these registers. For information on the registers, refer to the *ccurdprc_user.h* include file that is supplied with the driver.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Mapped_Local_Ptr (void                          *Handle,
                                  ccurdprc_local_ctrl_data_t   **local_ptr)

   Description: Get mapped local pointer.

   Input:   void                         *Handle     (Handle pointer)
   Output:  ccurdprc_local_ctrl_data_t  **local_ptr (local struct ptr)
               -- structure in ccurdprc_user.h
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR              (successful)
               # CCURDPRC_LIB_BAD_HANDLE            (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN              (device not open)
               # CCURDPRC_LIB_INVALID_ARG           (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION       (local region not present)
 *****************************************************************************/
```

2.2.56 ccurDPRC_Get_Open_File_Descriptor()

When the library *ccurDPRC_Open()* call is successfully invoked, the board is opened using the system call *open(2)*. The file descriptor associated with this board is returned to the user with this call. This call allows advanced users to bypass the library and communicate directly with the driver with calls like *read(2), ioctl(2),* etc. Normally, this is not recommended as internal checking and locking is bypassed and the library calls can no longer maintain integrity of the functions. This is only provided for advanced users who want more control and are aware of the implications.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Open_File_Descriptor (void *Handle,
                                      int  *fd)

   Description: Get Open File Descriptor

   Input:   void                   *Handle    (Handle pointer)
   Output:  int                    *fd        (open file descriptor)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR        (successful)
               # CCURDPRC_LIB_BAD_HANDLE      (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN        (device not open)
               # CCURDPRC_LIB_INVALID_ARG     (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION (local region not present)

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

2.2.57 ccurDPRC_Get_Physical_Memory()

This call returns to the user the physical memory pointer and size that was previously allocated by the *ccurDPRC_Mmap_Physical_Memory()* call. The physical memory is allocated by the user when they wish to perform their own DMA and bypass the API. Once again, this call is only useful for advanced users.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
```

```
   ccurDPRC_Get_Physical_Memory (void                *Handle,
                                 ccurdprc_phys_mem_t *phys_mem)

   Description: Get previously mmapped() physical memory address and size

   Input:       void                *Handle             (handle pointer)
   Output:      ccurdprc_phys_mem_t *phys_mem           (mem struct pointer)
                   void *phys_mem
                   uint  phys_mem_size
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
*****************************************************************************/
```

2.2.58 ccurDPRC_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.

```
/*****************************************************************************
   ccurDPRC_Get_Value()
   _ccurdprc_lib_error_number_t
   ccurDPRC_Get_Value (void             *Handle,
                       CCURDPRC_CONTROL cmd,
                       void             *value)

   Description: Return the value of the specified board register.

   Input:       void              *Handle               (handle pointer)
                CCURDPRC_CONTROL  cmd                   (register definition)
                    # CCURDPRC_CONTROL_BOARD_INFORMATION
                    # CCURDPRC_CONTROL_BOARD_CSR
                    # CCURDPRC_CONTROL_INTERRUPT_CONTROL
                    # CCURDPRC_CONTROL_INTERRUPT_STATUS
                    # CCURDPRC_CONTROL_CALIBRATION_BUS_CONTROL
                    # CCURDPRC_CONTROL_FIRMWARE_SPI_COUNTER_STATUS
                    # CCURDPRC_CONTROL_ADC_ENABLE
                    # CCURDPRC_CONTROL_DIGITAL_POTENTIOMETER_AND_IO_ENABLE

                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_0
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_1
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_2
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_3
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_4
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_5
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_6
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_7
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_8
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_9
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_10
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_11
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_12
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_13
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_14
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_15

                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_0
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_1
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_2
```

```
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_3
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_4
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_5
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_6
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_7
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_8
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_9
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_10
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_11
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_12
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_13
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_14
# CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_15

# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_0
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_1
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_2
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_3
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_4
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_5
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_6
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_7
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_8
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_9
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_10
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_11
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_12
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_13
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_14
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_15

# CCURDPRC_ADC_DATA_CHANNEL_0
# CCURDPRC_ADC_DATA_CHANNEL_1
# CCURDPRC_ADC_DATA_CHANNEL_2
# CCURDPRC_ADC_DATA_CHANNEL_3
# CCURDPRC_ADC_DATA_CHANNEL_4
# CCURDPRC_ADC_DATA_CHANNEL_5
# CCURDPRC_ADC_DATA_CHANNEL_6
# CCURDPRC_ADC_DATA_CHANNEL_7
# CCURDPRC_ADC_DATA_CHANNEL_8
# CCURDPRC_ADC_DATA_CHANNEL_9
# CCURDPRC_ADC_DATA_CHANNEL_10
# CCURDPRC_ADC_DATA_CHANNEL_11
# CCURDPRC_ADC_DATA_CHANNEL_12
# CCURDPRC_ADC_DATA_CHANNEL_13
# CCURDPRC_ADC_DATA_CHANNEL_14
# CCURDPRC_ADC_DATA_CHANNEL_15

# CCURDPRC_CONTROL_SPROM_STAT_ADDR_WRITE_DATA

# CCURDPRC_IO_CONTROL_CHANNEL_0
# CCURDPRC_IO_CONTROL_CHANNEL_1
# CCURDPRC_IO_CONTROL_CHANNEL_2
# CCURDPRC_IO_CONTROL_CHANNEL_3
# CCURDPRC_IO_CONTROL_CHANNEL_4
# CCURDPRC_IO_CONTROL_CHANNEL_5
# CCURDPRC_IO_CONTROL_CHANNEL_6
# CCURDPRC_IO_CONTROL_CHANNEL_7
# CCURDPRC_IO_CONTROL_CHANNEL_8
# CCURDPRC_IO_CONTROL_CHANNEL_9
# CCURDPRC_IO_CONTROL_CHANNEL_10
# CCURDPRC_IO_CONTROL_CHANNEL_11
# CCURDPRC_IO_CONTROL_CHANNEL_12
```

```
# CCURDPRC_IO_CONTROL_CHANNEL_13
# CCURDPRC_IO_CONTROL_CHANNEL_14
# CCURDPRC_IO_CONTROL_CHANNEL_15

# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_0
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_1
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_2
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_3
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_4
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_5
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_6
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_7
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_8
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_9
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_10
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_11
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_12
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_13
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_14
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_15

# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_0
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_1
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_2
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_3
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_4
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_5
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_6
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_7
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_8
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_9
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_10
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_11
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_12
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_13
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_14
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_15

# CCURDPRC_CONTROL_SPROM_READ_DATA

# CCURDPRC_ELECTRONIC_FUSE_STATUS

# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_0
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_1
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_2
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_3
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_4
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_5
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_6
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_7
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_8
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_9
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_10
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_11
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_12
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_13
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_14
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_15

# CCURDPRC_ELECTRONIC_FUSE_ELECTRICAL_SHORT_VALUE
# CCURDPRC_ELECTRONIC_FUSE_DELAY_VALUE
# CCURDPRC_ELECTRONIC_FUSE_COUNT_VALUE
# CCURDPRC_ELECTRONIC_FUSE_IO_DELAY_VALUE
```

```
                    # CCURDPRC_ELECTRONIC_FUSE_MAXIMUM_RESISTANCE
                    # CCURDPRC_ELECTRONIC_FUSE_MAXIMUM_VOLTAGE
                    # CCURDPRC_ELECTRONIC_FUSE_VOLTAGE_FAULT_DELAY

                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_0
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_1
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_2
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_3
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_4
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_5
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_6
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_7
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_8
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_9
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_10
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_11
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_12
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_13
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_14
                    # CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_15

                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_0
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_1
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_2
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_3
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_4
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_5
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_6
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_7
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_8
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_9
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_10
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_11
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_12
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_13
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_14
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_15

                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_0
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_1
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_2
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_3
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_4
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_5
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_6
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_7
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_8
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_9
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_10
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_11
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_12
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_13
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_14
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_15

                    # CCURDPRC_CONTROL_SPI_RAM
Output:      void              *value;              (pointer to value)
Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR            (successful)
                    # CCURDPRC_LIB_BAD_HANDLE          (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN            (device not open)
                    # CCURDPRC_LIB_INVALID_ARG         (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION     (local region error)
```

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

2.2.59 ccurDPRC_Hex_To_Fraction()

This call converts a hexadecimal value to a fractional decimal.

```
/****************************************************************************
   double
   ccurDPRC_Hex_To_Fraction (uint value)

   Description: Convert Hexadecimal to Fractional Decimal

   Input:       uint     value                  (hexadecimal to convert)
   Output:      none
   Return:      double   Fraction               (converted fractional value)
 ****************************************************************************/
```

2.2.60 ccurDPRC_Identify_Board()

This call is useful in identifying a physical board via software control. It causes the front LED to either flash or stay steady. Users can also specify the number of seconds they wish to flash the LED.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Identify_Board (void                 *Handle,
                            _ccurdprc_identify_t Identify)

   Description: Identify the board by setting the front LED

   Input:   void                 *Handle              (Handle pointer)
            _ccurdprc_identify_t Identify             (Identify board settings)
               # CCURDPRC_IDENTIFY_OFF                (turn off flashing)
               # CCURDPRC_IDENTIFY_ON                 (turn on flashing)
               # Number of seconds to flash           (flash for number of seconds)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
 ****************************************************************************/
```

2.2.61 ccurDPRC_Initialize_Board()

This call resets the board to a default initial state. This call is currently identical to the *ccurDPRC_Reset_Board()* call.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Initialize_Board (void *Handle)

   Description: Initialize the board.

   Input:   void           *Handle             (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_IOCTL_FAILED      (driver ioctl call failed)
               # CCURDPRC_LIB_NO_LOCAL_REGION   (local region not present)
 ****************************************************************************/
```

2.2.62 ccurDPRC_MMap_Physical_Memory()

This call is provided for advanced users to create a physical memory of specified size that can be used for DMA. The allocated DMA memory is rounded to a page size. If a physical memory is not available, this call will fail, at which point the user will need to issue the *ccurDPRC_Munmap_Physical_Memory()* API call to remove the previously allocated physical memory.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_MMap_Physical_Memory (void *Handle,
                                  int  size,
                                  void **mem_ptr)

   Description: Allocate a physical DMA memory for size bytes.

   Input:       void        *Handle                     (handle pointer)
                int         size                        (size in bytes)
   Output:      void        **mem_ptr                   (mapped memory pointer)
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_MMAP_SELECT_FAILED   (mmap selection failed)
                    # CCURDPRC_LIB_MMAP_FAILED          (mmap failed)
*****************************************************************************/
```

2.2.63 ccurDPRC_Munmap_Physical_Memory()

This call simply removes a physical memory that was previously allocated by the *ccurDPRC_MMap_Physical_Memory()* API call.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Munmap_Physical_Memory (void *Handle)


   Description: Unmap a previously mapped physical DMA memory.

   Input:       void        *Handle                     (handle pointer)
   Output:      None
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_MMAP_SELECT_FAILED   (mmap selection failed)
                    # CCURDPRC_LIB_MMAP_FAILED          (mmap failed)
*****************************************************************************/
```

2.2.64 ccurDPRC_NanoDelay()

This call goes into a tight loop spinning for the requested nano seconds specified by the user.

```
/*****************************************************************************
   void
   ccurDPRC_NanoDelay (unsigned long long NanoDelay)

   Description: Delay (loop) for user specified nano-seconds

   Input:   unsigned long long NanoDelay       (number of nano-secs to delay)
   Output:  none
   Return:  none
```

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

2.2.65 ccurDPRC_Open()

This is the first call that needs to be issued by a user to open a device and access the board through the rest of the API calls. What is returned is a handle to a *void pointer* that is supplied as an argument to the other API calls. The *Board_Number* is a valid board number [0..9] that is associated with a physical card. There must exist a character special file */dev/ccurdprc<Board_Number>* for the call to be successful. One character special file is created for each board found when the driver is successfully loaded.

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.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Open (void     **My_Handle,
                  int      Board_Number,
                  int      oflag)

   Description: Open a device.

   Input:   void            **Handle               (Handle pointer to pointer)
            int             Board_Number           (0-9 board number)
            int             oflag                  (open flags)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR             (successful)
               # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
               # CCURDPRC_LIB_ALREADY_OPEN         (device already opened)
               # CCURDPRC_LIB_OPEN_FAILED          (device open failed)
               # CCURDPRC_LIB_ALREADY_MAPPED       (memory already mmapped)
               # CCURDPRC_LIB_MMAP_SELECT_FAILED   (mmap selection failed)
               # CCURDPRC_LIB_MMAP_FAILED          (mmap failed)
 ****************************************************************************/
```

2.2.66 ccurDPRC_Read()

Currently, this call is not supported. It basically calls the *read(2)* system call with the exception that it performs necessary *locking* and returns the *errno* returned from the system call in the pointer to the *error* variable.

For specific information about the data being returned for the various read modes, refer to the *read(2)* system call description the *Driver Direct Access* section.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Read (void     *Handle,
                  void     *buf,
                  int      size,
                  int      *bytes_read,
```

```
                int      *error)

   Description: Perform a read operation.

   Input:   void     *Handle                (Handle pointer)
            int      size                   (size of buffer in bytes)
   Output:  void     *buf                   (pointer to buffer)
            int      *bytes_read            (bytes read)
            int      *error                 (returned errno)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
               # CCURDPRC_LIB_IO_ERROR      (read failed)
 ****************************************************************************/
```

2.2.67 ccurDPRC_Read_Serial_Prom()

This is a basic call to read short word entries from the serial prom. The user specifies a word offset within the serial prom and a word count, and the call returns the data read in a pointer to short words.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Read_Serial_Prom(void                *Handle,
                             ccurdprc_sprom_rw_t *spr)

   Description: Read Serial Prom for specified number of words

   Input:       void                    *Handle          (handle pointer)
                ccurdprc_sprom_rw_t     *spr             (pointer to struct)
                   u_short word_offset
                   u_short num_words
   Output:      ccurdprc_sprom_rw_t     *spr             (pointer to struct)
                   u_short *data_ptr
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                    # CCURDPRC_LIB_SERIAL_PROM_BUSY     (serial prom busy)
                    # CCURDPRC_LIB_SERIAL_PROM_FAILURE  (serial prom failure)

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

2.2.68 ccurDPRC_Read_Serial_Prom_Item()

This call is used to read well defined sections in the serial prom. The user supplies the serial prom section that needs to be read and the data is returned in a section specific structure.

```
/****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Read_Serial_Prom_Item(void                       *Handle,
                                  _ccurdprc_sprom_access_t item,
                                  void                       *item_ptr)

   Description: Read Serial Prom for specified item

   Input:       void                    *Handle          (handle pointer)
                _ccurdprc_sprom_access_t item            (select item)
                   CCURDPRC_SPROM_HEADER
   Output:      ccurdprc_sprom_header_t sprom_header     (pinter to item struct)
                   u_int32_t   board_serial_number
```

```
                    u_short     sprom_revision
   Return:       _ccurdprc_lib_error_number_t
                     # CCURDPRC_LIB_NO_ERROR             (successful)
                     # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                     # CCURDPRC_LIB_NOT_OPEN             (device not open)
                     # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                     # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                     # CCURDPRC_LIB_SERIAL_PROM_BUSY     (serial prom busy)
                     # CCURDPRC_LIB_SERIAL_PROM_FAILURE  (serial prom failure)
*****************************************************************************/
```

2.2.69 ccurDPRC_Remove_Irq()

The purpose of this call is to remove the interrupt handler that was previously set up. The interrupt handler is managed internally by the driver and the library. The user should not issue this call, otherwise reads will time out.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Remove_Irq (void *Handle)

   Description: By default, the driver sets up a shared IRQ interrupt handler
                when the device is opened. Now if for any reason, another
                device is sharing the same IRQ as this driver, the interrupt
                handler will also be entered every time the other shared
                device generates an interrupt. There are times that a user,
                for performance reasons may wish to run the board without
                interrupts enabled. In that case, they can issue this ioctl
                to remove the interrupt handling capability from the driver.

   Input:   void *Handle                    (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
               # CCURDPRC_LIB_IOCTL_FAILED  (driver ioctl call failed)
 *****************************************************************************/
```

2.2.70 ccurDPRC_Reset_Board()

This call resets the board to a known initial default state. This call is currently identical to the *ccurDPRC_Initialize_Board()* call.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Reset_Board (void *Handle)

   Description: Reset the board.

   Input:   void *Handle                      (Handle pointer)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR        (successful)
               # CCURDPRC_LIB_BAD_HANDLE      (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN        (device not open)
               # CCURDPRC_LIB_IOCTL_FAILED    (driver ioctl call failed)
               # CCURDPRC_LIB_NO_LOCAL_REGION (local region not present)
 *****************************************************************************/
```

2.2.71 ccurDPRC_Select_Driver_Read_Mode()

This call can be used to select the driver read mode.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Select_Driver_Read_Mode (void                        *Handle,
                                     _ccurdprc_driver_rw_mode_t mode)

   Description: Select Driver Read Mode

   Input:       void                        *Handle      (handle pointer)
                _ccurdprc_driver_rw_mode_t mode         (select read mode)
                    # CCURDPRC_PIO_CHANNEL
                    # CCURDPRC_DMA_CHANNEL              (CURRENTLY NOT SUPPORTED)
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
*****************************************************************************/
```

2.2.72 ccurDPRC_Select_Driver_Write_Mode()

This call is currently not supported for driver writes.This call can be used to select the driver write mode.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Select_Driver_Write_Mode (void                        *Handle,
                                      _ccurdprc_driver_rw_mode_t mode)

   Description: Select Driver Write Mode

   Input:       void                        *Handle      (handle pointer)
                _ccurdprc_driver_rw_mode_t mode         (select write mode)
                    # CCURDPRC_PIO_CHANNEL
                    # CCURDPRC_DMA_CHANNEL
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
*****************************************************************************/
```

2.2.73 ccurDPRC_Serial_Prom_Write_Override()

The serial prom is non-volatile and its information is preserved during a power cycle. It contains useful information and settings that the customer could lose if they were to inadvertently overwrite. For this reason, all calls that write to the serial proms will fail with a write protect error, unless this write protect override API is invoked prior to writing to the serial proms. Once the Write Override is enabled, it will stay in effect until the user closes the device or re-issues this call to disable writes to the serial prom.
The calls that will fail unless the write protect is disabled are:

- ccurDPRC_Write_Serial_Prom()
- ccurDPRC_Write_Serial_Prom_Item()

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Serial_Prom_Write_Override (void *Handle,
                                        int  action)
```

```
   Description: Set Serial Prom Write Override

   Input:       void                *Handle             (handle pointer)
                _ccurdprc_bool_t    action              (override action)
                    # CCURDPRC_TRUE
                    # CCURDPRC_FALSE
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
*****************************************************************************/
```

2.2.74 ccurDPRC_Set_Board_CSR()

This call is used to set the board control register.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Board_CSR (void                    *Handle,
                           ccurdprc_board_csr_t    *bcsr)

   Description: Set Board Control and Status information

   Input:   void                                *Handle (Handle pointer)
            ccurdprc_board_csr_t                *bcsr   (pointer to board csr)
               _ccurdprc_bcsr_identify_board_t identify_board
                  # CCURDPRC_BCSR_IDENTIFY_BOARD_DISABLE
                  # CCURDPRC_BCSR_IDENTIFY_BOARD_ENABLE
                  # CCURDPRC_BCSR_IDENTIFY_BOARD_ENABLE_DO_NOT_CHANGE
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR                (successful)
               # CCURDPRC_LIB_BAD_HANDLE              (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN                (device not open)
               # CCURDPRC_LIB_INVALID_ARG             (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION         (local region not present)
 *****************************************************************************/
```

2.2.75 ccurDPRC_Set_CalibrationBus_Control()

This call sets the calibration bus control.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_CalibrationBus_Control (void                          *Handle,
                                _ccurdprc_calibration_bus_control_t bus_control)

   Description: Set Calibration Bus Control

   Input:       void                *Handle             (handle pointer)
                _ccurdprc_calibration_bus_control_t
                                    bus_control         (control set)
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_OPEN
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_2_5_VOLTS
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_10_VOLTS
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_MINUS_10_VOLTS

                    # CCURDPRC_CALIBRATIONBUS_CONTROL_GROUND

                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_8_MILLIAMP
```

```
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_MINUS_8_MILLIAMP
                    # CCURDPRC_CALIBRATIONBUS_CONTROL_PLUS_16_MILLIAMP
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
 ************************************************************************/
```

2.2.76 ccurDPRC_Set_Digital_Potentiometer()

This call is used for setting the digital potentiometer resistance in ohms for the selected channels. Users need to supply valid resistances for channels in ohms and the call will set them accordingly and return the raw value supplied to the hardware. Valid resistances are 10 Ohms and 45 to 1000,000 Ohms in approximately 5 Ohms increments. If an exact resistance value is not supplied to the call, it will fail, unless the user tags the resistance with CCURDPRC_POTENTIOMETER_AUTOCORRECT_OHM_TAG flag. In this case, the call will attempt to get the nearest programmable resistance value and return this value back to the user after setting it.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Digital_Potentiometer_Test(void                          *Handle,
                            _ccurdprc_channel_mask_t                     ChanMask,
                            ccurdprc_digital_potentiometer_test_t        *DPTest)

   Description: Set Digital Potentiometer

   Input:   void                         *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t     ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
               # CCURDPRC_CHANNEL_MASK_2
               # CCURDPRC_CHANNEL_MASK_3
               # CCURDPRC_CHANNEL_MASK_4
               # CCURDPRC_CHANNEL_MASK_5
               # CCURDPRC_CHANNEL_MASK_6
               # CCURDPRC_CHANNEL_MASK_7
               # CCURDPRC_CHANNEL_MASK_8
               # CCURDPRC_CHANNEL_MASK_9
               # CCURDPRC_CHANNEL_MASK_10
               # CCURDPRC_CHANNEL_MASK_11
               # CCURDPRC_CHANNEL_MASK_12
               # CCURDPRC_CHANNEL_MASK_13
               # CCURDPRC_CHANNEL_MASK_14
               # CCURDPRC_CHANNEL_MASK_15
               # CCURDPRC_ALL_CHANNELS_MASK
            ccurdprc_digital_potentiometer_t  *DPValue (pointer to Digital
                                                        Potentiometer Value)
                 uint     Raw[CCURDPRC_MAX_CHANNELS];
                 uint    Ohms[CCURDPRC_MAX_CHANNELS];
   Output:  None
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR             (successful)
               # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN             (device not open)
               # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
               # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                  (Potentiometer and I/O Control not active)
 ************************************************************************/
```

2.2.77 ccurDPRC_Set_Digital_Potentiometer_Test()

This call sets the digital potentiometer test power down and modes for selected channels.

```
/*************************************************************************
   ccurDPRC_Set_Digital_Potentiometer_Test()

   Description: Set Digital Potentiometer Test

   Input:   void                         *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t     ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
               # CCURDPRC_CHANNEL_MASK_2
               # CCURDPRC_CHANNEL_MASK_3
               # CCURDPRC_CHANNEL_MASK_4
               # CCURDPRC_CHANNEL_MASK_5
               # CCURDPRC_CHANNEL_MASK_6
               # CCURDPRC_CHANNEL_MASK_7
               # CCURDPRC_CHANNEL_MASK_8
               # CCURDPRC_CHANNEL_MASK_9
               # CCURDPRC_CHANNEL_MASK_10
               # CCURDPRC_CHANNEL_MASK_11
               # CCURDPRC_CHANNEL_MASK_12
               # CCURDPRC_CHANNEL_MASK_13
               # CCURDPRC_CHANNEL_MASK_14
               # CCURDPRC_CHANNEL_MASK_15
               # CCURDPRC_ALL_CHANNELS_MASK
            ccurdprc_digital_potentiometer_test_t *DPTest (pointer to Digital
                                                           Potentiometer Test)
                  _ccurdprc_digital_potentiometer_test_t
DigitalPotTest[CCURDPRC_MAX_CHANNELS];
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT0
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT1
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT2
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_20K_POT3
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_PWRDWN_100K
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_FORCE_FAILURE
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_MODE_20K
                     # CCURDPRC_DIGITAL_POTENTIOMETER_TEST_MODE_100K
   Output:  None
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR             (successful)
               # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN             (device not open)
               # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
               # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
               # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                    (Potentiometer and I/O Control not active)
 *************************************************************************/
```

2.2.78 ccurDPRC_Set_Interrupt_Control()

This call sets the interrupt control.

```
/*************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Interrupt_Control (void                  *Handle,
                                   ccurdprc_interrupt_t *intr)

   Description: Set Interrupt Control information

   Input:       void                    *Handle     (handle pointer)
```

```
                ccurdprc_interrupt_t   *intr       (pointer to interrupt control)
                   int global_int
                       # CCURDPRC_ICSR_GLOBAL_DISABLE
                       # CCURDPRC_ICSR_GLOBAL_ENABLE
                       # CCURDPRC_DO_NOT_CHANGE
                   int plx_local_int
                       # CCURDPRC_ICSR_LOCAL_PLX_DISABLE
                       # CCURDPRC_ICSR_LOCAL_PLX_ENABLE
                       # CCURDPRC_DO_NOT_CHANGE
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR          (successful)
                    # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN          (device not open)
                    # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION   (local region error)
 ************************************************************************/
```

2.2.79 ccurDPRC_Set_Interrupt_Status()

This call sets/clears the PLX interrupt.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Interrupt_Status (void                  *Handle,
                                  ccurdprc_interrupt_t *intr)

   Description: Set Interrupt Status information

   Input:       void                    *Handle      (handle pointer)
                ccurdprc_interrupt_t    *intr        (pointer to interrupt status)
                   int plx_local_int
                       # CCURDPRC_INTSTAT_LOCAL_PLX_NONE
                       # CCURDPRC_INTSTAT_LOCAL_PLX_RESET
                       # CCURDPRC_DO_NOT_CHANGE
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR          (successful)
                    # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN          (device not open)
                    # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION   (local region error)
************************************************************************/
```

2.2.80 ccurDPRC_Set_Interrupt_Timeout_Seconds()

This call sets the read *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.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Interrupt_Timeout_Seconds (void    *Handle,
                                           int     timeout_secs)

   Description: Set Interrupt Timeout Seconds

   Input:   void         *Handle            (Handle pointer)
            int          timeout_secs       (interrupt tout secs)
   Output:  none
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR      (successful)
```

```
               # CCURDPRC_LIB_BAD_HANDLE    (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN      (device not open)
               # CCURDPRC_LIB_INVALID_ARG   (invalid argument)
 ************************************************************************/
```

2.2.81 ccurDPRC_Set_IO_Control()

This call sets the I/O Controls for the selected channels.

```
/************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_IO_Control(void                    *Handle,
                           _ccurdprc_channel_mask_t ChanMask,
                           ccurdprc_io_control_t    *IoControl)

   Description: Set I/O Control

   Input:   void                        *Handle       (Handle pointer)
            _ccurdprc_channel_mask_t    ChanMask      (specify channel mask)
               # CCURDPRC_CHANNEL_MASK_0
               # CCURDPRC_CHANNEL_MASK_1
               # CCURDPRC_CHANNEL_MASK_2
               # CCURDPRC_CHANNEL_MASK_3
               # CCURDPRC_CHANNEL_MASK_4
               # CCURDPRC_CHANNEL_MASK_5
               # CCURDPRC_CHANNEL_MASK_6
               # CCURDPRC_CHANNEL_MASK_7
               # CCURDPRC_CHANNEL_MASK_8
               # CCURDPRC_CHANNEL_MASK_9
               # CCURDPRC_CHANNEL_MASK_10
               # CCURDPRC_CHANNEL_MASK_11
               # CCURDPRC_CHANNEL_MASK_12
               # CCURDPRC_CHANNEL_MASK_13
               # CCURDPRC_CHANNEL_MASK_14
               # CCURDPRC_CHANNEL_MASK_15
               # CCURDPRC_ALL_CHANNELS_MASK
            ccurdprc_io_control_t             *IoControl (pointer to I/O control)
                 _ccurdprc_io_control_t
IoSignal[CCURDPRC_MAX_CHANNELS];
                  # CCURDPRC_IO_CONTROL_OPEN
                  # CCURDPRC_IO_CONTROL_EXTERNAL
                  # CCURDPRC_IO_CONTROL_TEST_BUS
                  # CCURDPRC_IO_CONTROL_RA_GROUND_FAULT
                  # CCURDPRC_IO_CONTROL_RB_GROUND_FAULT
                  # CCURDPRC_IO_CONTROL_RA_RB_GROUND_FAULT
                  # CCURDPRC_IO_CONTROL_RA_V_PLUS_FAULT
                  # CCURDPRC_IO_CONTROL_RB_V_PLUS_FAULT
                  # CCURDPRC_IO_CONTROL_RA_RB_V_PLUS_FAULT
                  # CCURDPRC_IO_CONTROL_RA_V_PLUS_RB_GROUND_FAULT_SWITCH
                  # CCURDPRC_IO_CONTROL_RA_GROUND_RB_V_PLUS_FAULT_SWITCH
   Output:  None
   Return:  _ccurdprc_lib_error_number_t
                  # CCURDPRC_LIB_NO_ERROR             (successful)
                  # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                  # CCURDPRC_LIB_NOT_OPEN             (device not open)
                  # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                  # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                  # CCURDPRC_LIB_DIGITAL_POT_AND_IO_IS_NOT_ACTIVE
                                    (Potentiometer and I/O Control not active)
                  # CCURDPRC_LIB_ADC_FAILURE          (ADC failure)
                  # CCURDPRC_LIB_ELECTRONIC_FUSE_TRIPPED
                                                      (Electronic Fuse tripped)
 ************************************************************************/
```

2.2.82 ccurDPRC_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.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Set_Value (void             *Handle,
                       CCURDPRC_CONTROL cmd,
                       void             *value)

   Description: Set the value of the specified board register.

   Input:       void             *Handle                (handle pointer)
                CCURDPRC_CONTROL cmd                    (register definition)
                    # CCURDPRC_CONTROL_BOARD_CSR
                    # CCURDPRC_CONTROL_INTERRUPT_CONTROL
                    # CCURDPRC_CONTROL_INTERRUPT_STATUS
                    # CCURDPRC_CONTROL_CALIBRATION_BUS_CONTROL
                    # CCURDPRC_CONTROL_FIRMWARE_SPI_COUNTER_STATUS
                    # CCURDPRC_CONTROL_ADC_ENABLE
                    # CCURDPRC_CONTROL_DIGITAL_POTENTIOMETER_AND_IO_ENABLE

                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_0
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_1
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_2
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_3
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_4
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_5
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_6
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_7
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_8
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_9
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_10
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_11
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_12
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_13
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_14
                    # CCURDPRC_ADC_POSITIVE_CAL_CHANNEL_15

                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_0
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_1
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_2
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_3
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_4
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_5
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_6
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_7
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_8
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_9
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_10
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_11
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_12
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_13
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_14
                    # CCURDPRC_ADC_NEGATIVE_CAL_CHANNEL_15
```

```
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_0
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_1
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_2
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_3
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_4
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_5
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_6
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_7
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_8
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_9
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_10
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_11
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_12
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_13
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_14
# CCURDPRC_ADC_OFFSET_CAL_CHANNEL_15

# CCURDPRC_CONTROL_SPROM_STAT_ADDR_WRITE_DATA

# CCURDPRC_IO_CONTROL_CHANNEL_0
# CCURDPRC_IO_CONTROL_CHANNEL_1
# CCURDPRC_IO_CONTROL_CHANNEL_2
# CCURDPRC_IO_CONTROL_CHANNEL_3
# CCURDPRC_IO_CONTROL_CHANNEL_4
# CCURDPRC_IO_CONTROL_CHANNEL_5
# CCURDPRC_IO_CONTROL_CHANNEL_6
# CCURDPRC_IO_CONTROL_CHANNEL_7
# CCURDPRC_IO_CONTROL_CHANNEL_8
# CCURDPRC_IO_CONTROL_CHANNEL_9
# CCURDPRC_IO_CONTROL_CHANNEL_10
# CCURDPRC_IO_CONTROL_CHANNEL_11
# CCURDPRC_IO_CONTROL_CHANNEL_12
# CCURDPRC_IO_CONTROL_CHANNEL_13
# CCURDPRC_IO_CONTROL_CHANNEL_14
# CCURDPRC_IO_CONTROL_CHANNEL_15

# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_0
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_1
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_2
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_3
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_4
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_5
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_6
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_7
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_8
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_9
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_10
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_11
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_12
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_13
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_14
# CCURDPRC_DIGITAL_POT_VALUE_CHANNEL_15

# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_0
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_1
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_2
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_3
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_4
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_5
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_6
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_7
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_8
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_9
```

```
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_10
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_11
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_12
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_13
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_14
# CCURDPRC_DIGITAL_POT_TEST_CHANNEL_15

# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_0
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_1
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_2
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_3
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_4
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_5
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_6
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_7
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_8
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_9
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_10
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_11
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_12
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_13
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_14
# CCURDPRC_ELECTRONIC_FUSE_TRIP_CHANNEL_15

# CCURDPRC_ELECTRONIC_FUSE_ELECTRICAL_SHORT_VALUE
# CCURDPRC_ELECTRONIC_FUSE_DELAY_VALUE
# CCURDPRC_ELECTRONIC_FUSE_COUNT_VALUE
# CCURDPRC_ELECTRONIC_FUSE_IO_DELAY_VALUE
# CCURDPRC_ELECTRONIC_FUSE_MAXIMUM_RESISTANCE
# CCURDPRC_ELECTRONIC_FUSE_MAXIMUM_VOLTAGE
# CCURDPRC_ELECTRONIC_FUSE_VOLTAGE_FAULT_DELAY

# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_0
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_1
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_2
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_3
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_4
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_5
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_6
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_7
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_8
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_9
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_10
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_11
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_12
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_13
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_14
# CCURDPRC_ELECTRONIC_FUSE_BASE_CHANNEL_15

# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_0
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_1
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_2
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_3
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_4
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_5
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_6
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_7
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_8
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_9
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_10
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_11
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_12
# CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_13
```

```
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_14
                    # CCURDPRC_ELECTRONIC_FUSE_MULTIPLIER_CHANNEL_15

                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_0
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_1
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_2
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_3
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_4
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_5
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_6
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_7
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_8
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_9
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_10
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_11
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_12
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_13
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_14
                    # CCURDPRC_ELECTRONIC_FUSE_THRESHOLD_CHANNEL_15

                    # CCURDPRC_CONTROL_SPI_RAM
                void             *value              (pointer to value to be set)
   Output:      None
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR          (successful)
                    # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN          (device not open)
                    # CCURDPRC_LIB_INVALID_ARG       (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION   (local region error)
 ****************************************************************************/
```

2.2.83 ccurDPRC_VoltsToData()

This call converts user supplied volts to raw data.

```
/****************************************************************************
   uint
   ccurDPRC_VoltsToData (double volts)

   Description: Convert Volts to data

   Input:       double   volts                 (volts to convert)
   Output:      none
   Return:      uint     data                  (returned data)
 ****************************************************************************/
```

2.2.84 ccurDPRC_VoltsToDataChanCal()

This call converts user supplied volts to raw data for calibration registers.

```
/****************************************************************************
   uint
   ccurDPRC_VoltsToDataChanCal (double volts)

   Description: Convert Volts to Data (for Channel Calibration)

   Input:       double   volts                 (volts to convert)
   Output:      none
   Return:      uint     data                  (returned data)
****************************************************************************/
```

2.2.85 ccurDPRC_Wait_For_Interrupt()

This call is made available to advanced users to bypass the API and perform their own interrupt handling. If a time out value greater than zero is specified, the call will time out after the specified seconds, otherwise it will not time out.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Wait_For_Interrupt (void                  *Handle,
                                ccurdprc_driver_int_t *drv_int)

   Description: Wait For Interrupt

   Input:       void                    *Handle        (handle pointer)
   Output:      ccurdprc_driver_int_t  *drv_int       (pointer to drv_int struct)
                   unsigned long long count
                   u_int status
                   u_int mask
                        # CCURDPRC_INTSTAT_LOCAL_PLX_MASK
                   int timeout_seconds
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR           (successful)
                    # CCURDPRC_LIB_BAD_HANDLE         (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN           (device not open)
                    # CCURDPRC_LIB_INVALID_ARG        (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION    (local region error)
*****************************************************************************/
```

2.2.86 ccurDPRC_Write()

This call is currently not supported.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Write (void    *Handle,
                   void    *buf,
                   int     size,
                   int     *bytes_written,
                   int     *error)

   Description: Perform a write operation.

   Input:   void     *Handle                    (Handle pointer)
            int      size                       (number of bytes to write)
   Output:  void     *buf                       (pointer to buffer)
            int      *bytes_written             (bytes written)
            int      *error                     (returned errno)
   Return:  _ccurdprc_lib_error_number_t
               # CCURDPRC_LIB_NO_ERROR          (successful)
               # CCURDPRC_LIB_BAD_HANDLE        (no/bad handler supplied)
               # CCURDPRC_LIB_NOT_OPEN          (device not open)
               # CCURDPRC_LIB_IO_ERROR          (write failed)
               # CCURDPRC_LIB_NOT_IMPLEMENTED   (call not implemented)
 *****************************************************************************/
```

2.2.87 ccurDPRC_Write_Serial_Prom()

This is a basic call to write short word entries to the serial prom. The user specifies a word offset within the serial prom and a word count, and the call writes the data pointed to by the *spw* pointer, in short words.

Prior to using this call, the user will need to issue the *ccurDPRC_Serial_Prom_Write_Override()* to allowing writing to the serial prom.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Write_Serial_Prom(void                *Handle,
                              ccurdprc_sprom_rw_t *spw)

   Description: Write data to Serial Prom for specified number of words

   Input:       void                   *Handle          (handle pointer)
                ccurdprc_sprom_rw_t    *spw             (pointer to struct)
                   # u_short word_offset
                   # u_short num_words
                   # u_short *data_ptr
   Output:      none
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                    # CCURDPRC_LIB_SERIAL_PROM_BUSY     (serial prom busy)
                    # CCURDPRC_LIB_SERIAL_PROM_FAILURE  (serial prom failure)
*****************************************************************************/
```

2.2.88 ccurDPRC_Write_Serial_Prom_Item()

This call is used to write well defined sections in the serial prom. The user supplies the serial prom section that needs to be written and the data points to the section specific structure. This call should normally not be used by the user.

Prior to using this call, the user will need to issue the *ccurDPRC_Serial_Prom_Write_Override()* to allowing writing to the serial prom.

```
/*****************************************************************************
   _ccurdprc_lib_error_number_t
   ccurDPRC_Write_Serial_Prom_Item(void                      *Handle,
                                   _ccurdprc_sprom_access_t item,
                                   void                      *item_ptr)

   Description: Write Serial Prom with specified item

   Input:       void                   *Handle          (handle pointer)
                _ccurdprc_sprom_access_t item           (select item)
                    # CCURDPRC_SPROM_HEADER
   Output:      ccurdprc_sprom_header_t sprom_header    (pinter to item struct)
                   u_int32_t   board_serial_number
                   u_short     sprom_revision
   Return:      _ccurdprc_lib_error_number_t
                    # CCURDPRC_LIB_NO_ERROR             (successful)
                    # CCURDPRC_LIB_BAD_HANDLE           (no/bad handler supplied)
                    # CCURDPRC_LIB_NOT_OPEN             (device not open)
                    # CCURDPRC_LIB_INVALID_ARG          (invalid argument)
                    # CCURDPRC_LIB_NO_LOCAL_REGION      (local region error)
                    # CCURDPRC_LIB_SERIAL_PROM_BUSY     (serial prom busy)
                    # CCURDPRC_LIB_SERIAL_PROM_FAILURE  (serial prom failure)
*****************************************************************************/
```

3. Test Programs

This driver and API are accompanied with an extensive set of test examples. Examples under the *Direct Driver Access* do not use the API, while those under *Application Program Interface Access* use the API.

3.1 Direct Driver Access Example Tests

These set of tests are located in the *.../test* directory and do not use the API. They communicate directly with the driver. Users should be extremely familiar with both the driver and the hardware registers if they wish to communicate directly with the hardware.

3.1.1 ccurdprc_dump

This test is for debugging purpose. It dumps all the hardware registers.

Usage: ccurdprc_dump [-b board]
 -b board: board number -- default board is 0

Example display:

./ccurdprc_dump

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

LOCAL Register 0x7ffff7ff6000 Offset=0x0
CONFIG Register 0x7ffff7ff5000 Offset=0x0

======= LOCAL BOARD REGISTERS =========
LBR: @0x0000 --> 0x93100102
LBR: @0x0004 --> 0x00000000
LBR: @0x0008 --> 0x00000000
LBR: @0x000c --> 0x00000000
LBR: @0x0010 --> 0x00000000
LBR: @0x0014 --> 0x00000001
LBR: @0x0018 --> 0x00000001
LBR: @0x001c --> 0x00000001
LBR: @0x0020 --> 0x00000001
LBR: @0x0024 --> 0x00000001
LBR: @0x0028 --> 0x00000001
LBR: @0x002c --> 0x00000001
LBR: @0x0030 --> 0x00000001
.
.
.
LBR: @0x07c0 --> 0x00000000
LBR: @0x07c4 --> 0x00000000
LBR: @0x07c8 --> 0x00000000
LBR: @0x07cc --> 0x00000000
LBR: @0x07d0 --> 0x00000000
LBR: @0x07d4 --> 0x00000000
LBR: @0x07d8 --> 0x00000000
LBR: @0x07dc --> 0x00000000
LBR: @0x07e0 --> 0x00000000
LBR: @0x07e4 --> 0x00000000
LBR: @0x07e8 --> 0x00000000
LBR: @0x07ec --> 0x00000000
LBR: @0x07f0 --> 0x00000000
```

```
LBR: @0x07f4 --> 0x00000000
LBR: @0x07f8 --> 0x00000000
LBR: @0x07fc --> 0x00000000

======= LOCAL CONFIG REGISTERS =========
LCR: @0x0000 --> 0xfffff800
LCR: @0x0004 --> 0x00000001
LCR: @0x0008 --> 0x00200000
LCR: @0x000c --> 0x00300400
LCR: @0x0010 --> 0x00000000
LCR: @0x0014 --> 0x00000000
LCR: @0x0018 --> 0x42430343
LCR: @0x001c --> 0x00000000
LCR: @0x0020 --> 0x00000000
LCR: @0x0024 --> 0x00000000
LCR: @0x0028 --> 0x00000000
LCR: @0x002c --> 0x00000000
LCR: @0x0030 --> 0x00000000
.
.
.
LCR: @0x00c0 --> 0x00000002
LCR: @0x00c4 --> 0x00000000
LCR: @0x00c8 --> 0x00000000
LCR: @0x00cc --> 0x00000000
LCR: @0x00d0 --> 0x00000000
LCR: @0x00d4 --> 0x00000000
LCR: @0x00d8 --> 0x00000000
LCR: @0x00dc --> 0x00000000
LCR: @0x00e0 --> 0x00000000
LCR: @0x00e4 --> 0x00000000
LCR: @0x00e8 --> 0x00000050
LCR: @0x00ec --> 0x00000000
LCR: @0x00f0 --> 0x00000000
LCR: @0x00f4 --> 0x00000000
LCR: @0x00f8 --> 0x00000043
LCR: @0x00fc --> 0x00000000
LCR: @0x0100 --> 0x00000000
LCR: @0x0104 --> 0x00000000

======= PCI CONFIG REG ADDR MAPPING =========
PCR: @0x0000 --> 0x93101542
PCR: @0x0004 --> 0x02b00017
PCR: @0x0008 --> 0x08800001
PCR: @0x000c --> 0x00006008
PCR: @0x0010 --> 0xc4c01000
PCR: @0x0014 --> 0x00000000
PCR: @0x0018 --> 0xc4c00000
PCR: @0x001c --> 0x00000000
PCR: @0x0020 --> 0x00000000
PCR: @0x0024 --> 0x00000000
PCR: @0x0028 --> 0x00000000
PCR: @0x002c --> 0x905610b5
PCR: @0x0030 --> 0x00000000
PCR: @0x0034 --> 0x00000040
PCR: @0x0038 --> 0x00000000
PCR: @0x003c --> 0x0000010b
```

```
PCR: @0x0040 --> 0x00024801
PCR: @0x0044 --> 0x00000000
PCR: @0x0048 --> 0x00004c00
PCR: @0x004c --> 0x00000003
PCR: @0x0050 --> 0x00000000

======= PCI BRIDGE REGISTERS =========
PBR: @0x0000 --> 0x811110b5
PBR: @0x0004 --> 0x00100417
PBR: @0x0008 --> 0x06040021
PBR: @0x000c --> 0x00010010
PBR: @0x0010 --> 0xc490000c
PBR: @0x0014 --> 0x00000000
PBR: @0x0018 --> 0x00080807
PBR: @0x001c --> 0x220000f0
PBR: @0x0020 --> 0xc4c0c4c0
PBR: @0x0024 --> 0x0000fff0
PBR: @0x0028 --> 0x00000000
PBR: @0x002c --> 0x00000000
PBR: @0x0030 --> 0x00000000
.
.
.
PBR: @0x00d0 --> 0x00000000
PBR: @0x00d4 --> 0x00000000
PBR: @0x00d8 --> 0x00000000
PBR: @0x00dc --> 0x00000000
PBR: @0x00e0 --> 0x00000000
PBR: @0x00e4 --> 0x00000000
PBR: @0x00e8 --> 0x00000000
PBR: @0x00ec --> 0x00000000
PBR: @0x00f0 --> 0x00000000
PBR: @0x00f4 --> 0x00000000
PBR: @0x00f8 --> 0x00000000
PBR: @0x00fc --> 0x00000000
PBR: @0x0100 --> 0x00010004
PBR: @0x0104 --> 0x00000000
PBR: @0x0108 --> 0x00000000
PBR: @0x010c --> 0x00000000
PBR: @0x0110 --> 0x00000000
PBR: @0x0114 --> 0x00000000
PBR: @0x0118 --> 0x00000000

======= MAIN CONTROL REGISTERS =========
MCR: @0x0000 --> 0x00000033
MCR: @0x0004 --> 0x8000ff00
MCR: @0x0008 --> 0x00000000
MCR: @0x000c --> 0x1b008090
MCR: @0x0010 --> 0x80000002
MCR: @0x0014 --> 0x00000000
MCR: @0x0018 --> 0x00000000
MCR: @0x001c --> 0x00000000
MCR: @0x0020 --> 0x0000141f
MCR: @0x0024 --> 0x00000000
MCR: @0x0028 --> 0x00000000
MCR: @0x002c --> 0x00000000
MCR: @0x0030 --> 0xfeedface
```

```
MCR: @0x0034 --> 0x00000000
MCR: @0x0038 --> 0x00000000
MCR: @0x003c --> 0x00000000
MCR: @0x0040 --> 0x00000201
MCR: @0x0044 --> 0x00000000
MCR: @0x0048 --> 0x00810a20
MCR: @0x004c --> 0x000000d4
MCR: @0x0050 --> 0x00010700
MCR: @0x0054 --> 0x00000000
MCR: @0x0058 --> 0x080a2c2a
MCR: @0x005c --> 0x0000029a
MCR: @0x0060 --> 0x00000019
MCR: @0x0064 --> 0x00000000
```

3.1.2 ccurdprc_rdreg

This is a simple program that returns the local register value for a given offset.

```
Usage: ./ccurdprc_rdreg [-b board] [-o offset] [-s size]
 -b board : board number -- default board is 0
 -o offset: hex offset to read from -- default offset is 0x0
 -s size  : number of bytes to read -- default size is 0x4
```

Example display:
./ccurdprc_rdreg –s64

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

#### LOCAL REGS #### (length=100)
+LCL+        0   93100102  00000000  00000000  00000000 *...3............*
+LCL+     0x10   00000000  00000001  00000001  00000001 *................*
+LCL+     0x20   00000001  00000001  00000001  00000001 *................*
+LCL+     0x30   00000001  00000001  00000001  00000001 *................*
+LCL+     0x40   00000001  00000001  00000001  00000001 *................*
+LCL+     0x50   00000001  00000001  00000001  00000001 *................*
+LCL+     0x60   0000000
```

3.1.3 ccurdprc_reg

This call displays all the boards local and configuration registers.

```
Usage: ./ccurdprc_reg [-b board]
 -b board: Board number -- default board is 0
```

Example display:

./ccurdprc_reg

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

LOCAL Register 0x7ffff7ff6000 Offset=0x0

#### LOCAL REGS #### (length=2048)
+LCL+        0   93100102  00000000  00000000  00000000 *...3............*
+LCL+     0x10   00000000  00000001  00000001  00000001 *................*
+LCL+     0x20   00000001  00000001  00000001  00000001 *................*
+LCL+     0x30   00000001  00000001  00000001  00000001 *................*
+LCL+     0x40   00000001  00000001  00000001  00000001 *................*
+LCL+     0x50   00000001  00000001  00000001  00000001 *................*
```

```
+LCL+     0x60   00000001  00000001  00000001  00000001 *................*
+LCL+     0x70   00000001  00000001  00000001  00000001 *................*
+LCL+     0x80   00000001  00000001  00000001  00000001 *................*
+LCL+     0x90   00000001  00000001  00000001  00000001 *................*
+LCL+     0xa0   00000001  00000001  00000001  00000001 *................*
+LCL+     0xb0   00000000  00000001  00000001  00000001 *................*
+LCL+     0xc0   00000001  00000001  00000001  00000001 *................*
+LCL+     0xd0   00000001  00000001  00000001  00000001 *................*
+LCL+     0xe0   00000001  00000001  00000001  00000001 *................*
+LCL+     0xf0   00000000  00000000  00000001  00000001 *................*
+LCL+    0x100   00000000  00000001  00000001  00000001 *................*
+LCL+    0x110   80000000  80000000  80000000  80000000 *................*
+LCL+    0x120   80000000  80000000  80000000  80000000 *................*
+LCL+    0x130   80000000  80000000  80000000  80000000 *................*
+LCL+    0x140   80000000  80000000  80000000  80000000 *................*
+LCL+    0x150   80000000  80000000  80000000  80000000 *................*
+LCL+    0x160   80000000  80000000  80000000  80000000 *................*
+LCL+    0x170   80000000  80000000  80000000  80000000 *................*
+LCL+    0x180   80000000  80000000  80000000  80000000 *................*
+LCL+    0x190   00000000  00000000  00000000  00000000 *................*
+LCL+    0x1a0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x1b0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x1c0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x1d0   00000001  00000001  00000001  00000001 *................*
+LCL+    0x1e0   00000001  00000001  00000001  00000001 *................*
+LCL+    0x1f0   00000001  00000001  00000001  00000001 *................*
+LCL+    0x200   00000001  00000001  00000001  00000001 *................*
.
.
.
+LCL+    0x700   00000000  00000000  00000000  00000000 *................*
+LCL+    0x710   00000000  00000000  00000000  00000000 *................*
+LCL+    0x720   00000000  00000000  00000000  00000000 *................*
+LCL+    0x730   00000000  00000000  00000000  00000000 *................*
+LCL+    0x740   00000000  00000000  00000000  00000000 *................*
+LCL+    0x750   00000000  00000000  00000000  00000000 *................*
+LCL+    0x760   00000000  00000000  00000000  00000000 *................*
+LCL+    0x770   00000000  00000000  00000000  00000000 *................*
+LCL+    0x780   00000000  00000000  00000000  00000000 *................*
+LCL+    0x790   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7a0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7b0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7c0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7d0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7e0   00000000  00000000  00000000  00000000 *................*
+LCL+    0x7f0   00000000  00000000  00000000  00000000 *................*

CONFIG Register 0x7ffff7ff5000 Offset=0x0

#### CONFIG REGS #### (length=264)
+CFG+       0   fffff800  00000001  00200000  00300400 *........ ...0..*
+CFG+    0x10   00000000  00000000  42430343  00000000 *........BC.C....*
+CFG+    0x20   00000000  00000000  00000000  00000000 *................*
+CFG+    0x30   00000000  00000008  00000000  00000000 *................*
+CFG+    0x40   00000000  00000000  00000000  00000000 *................*
+CFG+    0x50   00000000  00000000  00000000  00000000 *................*
+CFG+    0x60   00000000  00000000  0f000080  100f767c *..............v|*
+CFG+    0x70   905610b5  000000ba  00000000  00000000 *.V..............*
+CFG+    0x80   00000003  00000000  00000000  00000000 *................*
+CFG+    0x90   00000000  00000003  00000000  00000000 *................*
+CFG+    0xa0   00000000  00000000  00001010  00200000 *.............  ..*
+CFG+    0xb0   00000000  00000000  00000000  00000000 *................*
+CFG+    0xc0   00000002  00000000  00000000  00000000 *................*
```

```
+CFG+     0xd0   00000000  00000000  00000000  00000000 *................*
+CFG+     0xe0   00000000  00000000  00000050  00000000 *...........P....*
+CFG+     0xf0   00000000  00000000  00000043  00000000 *...........C....*
+CFG+    0x100   00000000  00000000                     *........        *

======= CONFIG REGISTERS =========
    las0rr                                          =0xfffff800     @0x00000000
    las0ba                                          =0x00000001     @0x00000004
    marbr                                           =0x00200000     @0x00000008
    bigend                                          =0x00300400     @0x0000000c
    eromrr                                          =0x00000000     @0x00000010
    eromba                                          =0x00000000     @0x00000014
    lbrd0                                           =0x42430343     @0x00000018
    dmrr                                            =0x00000000     @0x0000001c
    dmlbam                                          =0x00000000     @0x00000020
    dmlbai                                          =0x00000000     @0x00000024
    dmpbam                                          =0x00000000     @0x00000028
    dmcfga                                          =0x00000000     @0x0000002c
    oplfis                                          =0x00000000     @0x00000030
    oplfim                                          =0x00000008     @0x00000034
    mbox0                                           =0x00000000     @0x00000040
    mbox1                                           =0x00000000     @0x00000044
    mbox2                                           =0x00000000     @0x00000048
    mbox3                                           =0x00000000     @0x0000004c
    mbox4                                           =0x00000000     @0x00000050
    mbox5                                           =0x00000000     @0x00000054
    mbox6                                           =0x00000000     @0x00000058
    mbox7                                           =0x00000000     @0x0000005c
    p2ldbell                                        =0x00000000     @0x00000060
    l2pdbell                                        =0x00000000     @0x00000064
    intcsr                                          =0x0f000080     @0x00000068
    cntrl                                           =0x100f767c     @0x0000006c
    pcihidr                                         =0x905610b5     @0x00000070
    pcihrev                                         =0x000000ba     @0x00000074
    dmamode0                                        =0x00000003     @0x00000080
    dmapadr0                                        =0x00000000     @0x00000084
    dmaladr0                                        =0x00000000     @0x00000088
    dmasiz0                                         =0x00000000     @0x0000008c
    dmadpr0                                         =0x00000000     @0x00000090
    dmamode1                                        =0x00000003     @0x00000094
    dmapadr1                                        =0x00000000     @0x00000098
    dmaladr1                                        =0x00000000     @0x0000009c
    dmasiz1                                         =0x00000000     @0x000000a0
    dmadpr1                                         =0x00000000     @0x000000a4
    dmacsr0                                         =0x00000010     @0x000000a8
    dmacsr1                                         =0x00000010     @0x000000a9
    dmaarb                                          =0x00200000     @0x000000ac
    dmathr                                          =0x00000000     @0x000000b0
    dmadac0                                         =0x00000000     @0x000000b4
    dmadac1                                         =0x00000000     @0x000000b8
    las1rr                                          =0x00000000     @0x000000f0
    las1ba                                          =0x00000000     @0x000000f4
    lbrd1                                           =0x00000043     @0x000000f8
    dmdac                                           =0x00000000     @0x000000fc
    pciarb                                          =0x00000000     @0x00000100
    pabtadr                                         =0x00000000     @0x00000104
======= LOCAL REGISTERS =========
    board_info                                      =0x93100102     @0x00000000
    board_csr                                       =0x00000000     @0x00000004
    interrupt_control                               =0x00000000     @0x00000008
    interrupt_status                                =0x00000000     @0x0000000c
    calib_bus_control                               =0x00000000     @0x000000b0
    spi_counter_status                              =0x00000000     @0x000000f0
```

```
ADC_Enable                                          =0x00000000     @0x00000100
ADC_PositiveCalibration[CCURDPRC_CHANNEL_0]         =0x80000000     @0x00000110
ADC_PositiveCalibration[CCURDPRC_CHANNEL_1]         =0x80000000     @0x00000114
ADC_PositiveCalibration[CCURDPRC_CHANNEL_2]         =0x80000000     @0x00000118
ADC_PositiveCalibration[CCURDPRC_CHANNEL_3]         =0x80000000     @0x0000011c
ADC_PositiveCalibration[CCURDPRC_CHANNEL_4]         =0x80000000     @0x00000120
ADC_PositiveCalibration[CCURDPRC_CHANNEL_5]         =0x80000000     @0x00000124
ADC_PositiveCalibration[CCURDPRC_CHANNEL_6]         =0x80000000     @0x00000128
ADC_PositiveCalibration[CCURDPRC_CHANNEL_7]         =0x80000000     @0x0000012c
ADC_PositiveCalibration[CCURDPRC_CHANNEL_8]         =0x80000000     @0x00000130
ADC_PositiveCalibration[CCURDPRC_CHANNEL_9]         =0x80000000     @0x00000134
ADC_PositiveCalibration[CCURDPRC_CHANNEL_10]        =0x80000000     @0x00000138
ADC_PositiveCalibration[CCURDPRC_CHANNEL_11]        =0x80000000     @0x0000013c
ADC_PositiveCalibration[CCURDPRC_CHANNEL_12]        =0x80000000     @0x00000140
ADC_PositiveCalibration[CCURDPRC_CHANNEL_13]        =0x80000000     @0x00000144
ADC_PositiveCalibration[CCURDPRC_CHANNEL_14]        =0x80000000     @0x00000148
ADC_PositiveCalibration[CCURDPRC_CHANNEL_15]        =0x80000000     @0x0000014c
ADC_NegativeCalibration[CCURDPRC_CHANNEL_0]         =0x80000000     @0x00000150
ADC_NegativeCalibration[CCURDPRC_CHANNEL_1]         =0x80000000     @0x00000154
ADC_NegativeCalibration[CCURDPRC_CHANNEL_2]         =0x80000000     @0x00000158
ADC_NegativeCalibration[CCURDPRC_CHANNEL_3]         =0x80000000     @0x0000015c
ADC_NegativeCalibration[CCURDPRC_CHANNEL_4]         =0x80000000     @0x00000160
ADC_NegativeCalibration[CCURDPRC_CHANNEL_5]         =0x80000000     @0x00000164
ADC_NegativeCalibration[CCURDPRC_CHANNEL_6]         =0x80000000     @0x00000168
ADC_NegativeCalibration[CCURDPRC_CHANNEL_7]         =0x80000000     @0x0000016c
ADC_NegativeCalibration[CCURDPRC_CHANNEL_8]         =0x80000000     @0x00000170
ADC_NegativeCalibration[CCURDPRC_CHANNEL_9]         =0x80000000     @0x00000174
ADC_NegativeCalibration[CCURDPRC_CHANNEL_10]        =0x80000000     @0x00000178
ADC_NegativeCalibration[CCURDPRC_CHANNEL_11]        =0x80000000     @0x0000017c
ADC_NegativeCalibration[CCURDPRC_CHANNEL_12]        =0x80000000     @0x00000180
ADC_NegativeCalibration[CCURDPRC_CHANNEL_13]        =0x80000000     @0x00000184
ADC_NegativeCalibration[CCURDPRC_CHANNEL_14]        =0x80000000     @0x00000188
ADC_NegativeCalibration[CCURDPRC_CHANNEL_15]        =0x80000000     @0x0000018c
ADC_OffsetCalibration[CCURDPRC_CHANNEL_0]           =0x00000000     @0x00000190
ADC_OffsetCalibration[CCURDPRC_CHANNEL_1]           =0x00000000     @0x00000194
ADC_OffsetCalibration[CCURDPRC_CHANNEL_2]           =0x00000000     @0x00000198
ADC_OffsetCalibration[CCURDPRC_CHANNEL_3]           =0x00000000     @0x0000019c
ADC_OffsetCalibration[CCURDPRC_CHANNEL_4]           =0x00000000     @0x000001a0
ADC_OffsetCalibration[CCURDPRC_CHANNEL_5]           =0x00000000     @0x000001a4
ADC_OffsetCalibration[CCURDPRC_CHANNEL_6]           =0x00000000     @0x000001a8
ADC_OffsetCalibration[CCURDPRC_CHANNEL_7]           =0x00000000     @0x000001ac
ADC_OffsetCalibration[CCURDPRC_CHANNEL_8]           =0x00000000     @0x000001b0
ADC_OffsetCalibration[CCURDPRC_CHANNEL_9]           =0x00000000     @0x000001b4
ADC_OffsetCalibration[CCURDPRC_CHANNEL_10]          =0x00000000     @0x000001b8
ADC_OffsetCalibration[CCURDPRC_CHANNEL_11]          =0x00000000     @0x000001bc
ADC_OffsetCalibration[CCURDPRC_CHANNEL_12]          =0x00000000     @0x000001c0
ADC_OffsetCalibration[CCURDPRC_CHANNEL_13]          =0x00000000     @0x000001c4
ADC_OffsetCalibration[CCURDPRC_CHANNEL_14]          =0x00000000     @0x000001c8
ADC_OffsetCalibration[CCURDPRC_CHANNEL_15]          =0x00000000     @0x000001cc
ADC_Data[CCURDPRC_CHANNEL_0]                        =0x00000000     @0x00000280
ADC_Data[CCURDPRC_CHANNEL_1]                        =0x00000000     @0x00000284
ADC_Data[CCURDPRC_CHANNEL_2]                        =0x00000000     @0x00000288
ADC_Data[CCURDPRC_CHANNEL_3]                        =0x00000000     @0x0000028c
ADC_Data[CCURDPRC_CHANNEL_4]                        =0x00000000     @0x00000290
ADC_Data[CCURDPRC_CHANNEL_5]                        =0x00000000     @0x00000294
ADC_Data[CCURDPRC_CHANNEL_6]                        =0x00000000     @0x00000298
ADC_Data[CCURDPRC_CHANNEL_7]                        =0x00000000     @0x0000029c
ADC_Data[CCURDPRC_CHANNEL_8]                        =0x00000000     @0x000002a0
ADC_Data[CCURDPRC_CHANNEL_9]                        =0x00000000     @0x000002a4
ADC_Data[CCURDPRC_CHANNEL_10]                       =0x00000000     @0x000002a8
ADC_Data[CCURDPRC_CHANNEL_11]                       =0x00000000     @0x000002ac
ADC_Data[CCURDPRC_CHANNEL_12]                       =0x00000000     @0x000002b0
ADC_Data[CCURDPRC_CHANNEL_13]                       =0x00000000     @0x000002b4
```

```
ADC_Data[CCURDPRC_CHANNEL_14]                    =0x00000000     @0x000002b8
ADC_Data[CCURDPRC_CHANNEL_15]                    =0x00000000     @0x000002bc
Digital_Potentiomenter_and_IO_Enable             =0x00000000     @0x00000300
IO_Signal_Select[CCURDPRC_CHANNEL_0]             =0x00000000     @0x00000310
IO_Signal_Select[CCURDPRC_CHANNEL_1]             =0x00000000     @0x00000314
IO_Signal_Select[CCURDPRC_CHANNEL_2]             =0x00000000     @0x00000318
IO_Signal_Select[CCURDPRC_CHANNEL_3]             =0x00000000     @0x0000031c
IO_Signal_Select[CCURDPRC_CHANNEL_4]             =0x00000000     @0x00000320
IO_Signal_Select[CCURDPRC_CHANNEL_5]             =0x00000000     @0x00000324
IO_Signal_Select[CCURDPRC_CHANNEL_6]             =0x00000000     @0x00000328
IO_Signal_Select[CCURDPRC_CHANNEL_7]             =0x00000000     @0x0000032c
IO_Signal_Select[CCURDPRC_CHANNEL_8]             =0x00000000     @0x00000330
IO_Signal_Select[CCURDPRC_CHANNEL_9]             =0x00000000     @0x00000334
IO_Signal_Select[CCURDPRC_CHANNEL_10]            =0x00000000     @0x00000338
IO_Signal_Select[CCURDPRC_CHANNEL_11]            =0x00000000     @0x0000033c
IO_Signal_Select[CCURDPRC_CHANNEL_12]            =0x00000000     @0x00000340
IO_Signal_Select[CCURDPRC_CHANNEL_13]            =0x00000000     @0x00000344
IO_Signal_Select[CCURDPRC_CHANNEL_14]            =0x00000000     @0x00000348
IO_Signal_Select[CCURDPRC_CHANNEL_15]            =0x00000000     @0x0000034c
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_0] =0x00032000     @0x00000400
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_1] =0x00032000     @0x00000404
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_2] =0x00032000     @0x00000408
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_3] =0x00032000     @0x0000040c
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_4] =0x00032000     @0x00000410
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_5] =0x00032000     @0x00000414
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_6] =0x00032000     @0x00000418
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_7] =0x00032000     @0x0000041c
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_8] =0x00032000     @0x00000420
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_9] =0x00032000     @0x00000424
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_10]=0x00032000     @0x00000428
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_11]=0x00032000     @0x0000042c
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_12]=0x00032000     @0x00000430
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_13]=0x00032000     @0x00000434
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_14]=0x00032000     @0x00000438
Digital_Potentiometer_Value[CCURDPRC_CHANNEL_15]=0x00032000     @0x0000043c
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_0]  =0x00000100     @0x00000480
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_1]  =0x00000100     @0x00000484
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_2]  =0x00000100     @0x00000488
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_3]  =0x00000100     @0x0000048c
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_4]  =0x00000100     @0x00000490
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_5]  =0x00000100     @0x00000494
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_6]  =0x00000100     @0x00000498
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_7]  =0x00000100     @0x0000049c
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_8]  =0x00000100     @0x000004a0
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_9]  =0x00000100     @0x000004a4
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_10] =0x00000100     @0x000004a8
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_11] =0x00000100     @0x000004ac
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_12] =0x00000100     @0x000004b0
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_13] =0x00000100     @0x000004b4
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_14] =0x00000100     @0x000004b8
Digital_Potentiometer_Test[CCURDPRC_CHANNEL_15] =0x00000100     @0x000004bc
sprom_stat_addr_write_data                       =0x00000000     @0x00000500
sprom_read_data                                  =0x00000000     @0x00000504
Electronic_Fuse_Status                           =0x00030000     @0x00000520
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_0]         =0x00000000     @0x00000530
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_1]         =0x00000000     @0x00000534
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_2]         =0x00000000     @0x00000538
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_3]         =0x00000000     @0x0000053c
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_4]         =0x00000000     @0x00000540
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_5]         =0x00000000     @0x00000544
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_6]         =0x00000000     @0x00000548
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_7]         =0x00000000     @0x0000054c
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_8]         =0x00000000     @0x00000550
```

```
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_9]        =0x00000000     @0x00000554
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_10]       =0x00000000     @0x00000558
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_11]       =0x00000000     @0x0000055c
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_12]       =0x00000000     @0x00000560
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_13]       =0x00000000     @0x00000564
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_14]       =0x00000000     @0x00000568
Electronic_Fuse_Trip[CCURDPRC_CHANNEL_15]       =0x00000000     @0x0000056c
Electronic_Fuse_Electrical_Short_Value          =0x000000b4     @0x000005b0
Electronic_Fuse_Delay_Value                     =0x00000002     @0x000005b4
Electronic_Fuse_Count_Value                     =0x00000001     @0x000005b8
Electronic_Fuse_IO_Delay_Value                  =0x00000800     @0x000005bc
Electronic_Fuse_Maximum_Resistance              =0x000000fd     @0x000005c0
Electronic_Fuse_Maximum_Voltage                 =0x00005ae0     @0x000005c4
Electronic_Fuse_Voltage_Fault_Delay             =0x00000002     @0x000005c8
Electronic_Fuse_Base[CCURDPRC_CHANNEL_0]        =0x00000241     @0x00000600
Electronic_Fuse_Base[CCURDPRC_CHANNEL_1]        =0x00000241     @0x00000604
Electronic_Fuse_Base[CCURDPRC_CHANNEL_2]        =0x00000241     @0x00000608
Electronic_Fuse_Base[CCURDPRC_CHANNEL_3]        =0x00000241     @0x0000060c
Electronic_Fuse_Base[CCURDPRC_CHANNEL_4]        =0x00000241     @0x00000610
Electronic_Fuse_Base[CCURDPRC_CHANNEL_5]        =0x00000241     @0x00000614
Electronic_Fuse_Base[CCURDPRC_CHANNEL_6]        =0x00000241     @0x00000618
Electronic_Fuse_Base[CCURDPRC_CHANNEL_7]        =0x00000241     @0x0000061c
Electronic_Fuse_Base[CCURDPRC_CHANNEL_8]        =0x00000241     @0x00000620
Electronic_Fuse_Base[CCURDPRC_CHANNEL_9]        =0x00000241     @0x00000624
Electronic_Fuse_Base[CCURDPRC_CHANNEL_10]       =0x00000241     @0x00000628
Electronic_Fuse_Base[CCURDPRC_CHANNEL_11]       =0x00000241     @0x0000062c
Electronic_Fuse_Base[CCURDPRC_CHANNEL_12]       =0x00000241     @0x00000630
Electronic_Fuse_Base[CCURDPRC_CHANNEL_13]       =0x00000241     @0x00000634
Electronic_Fuse_Base[CCURDPRC_CHANNEL_14]       =0x00000241     @0x00000638
Electronic_Fuse_Base[CCURDPRC_CHANNEL_15]       =0x00000241     @0x0000063c
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_0]  =0x0000005a     @0x00000640
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_1]  =0x0000005a     @0x00000644
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_2]  =0x0000005a     @0x00000648
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_3]  =0x0000005a     @0x0000064c
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_4]  =0x0000005a     @0x00000650
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_5]  =0x0000005a     @0x00000654
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_6]  =0x0000005a     @0x00000658
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_7]  =0x0000005a     @0x0000065c
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_8]  =0x0000005a     @0x00000660
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_9]  =0x0000005a     @0x00000664
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_10] =0x0000005a     @0x00000668
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_11] =0x0000005a     @0x0000066c
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_12] =0x0000005a     @0x00000670
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_13] =0x0000005a     @0x00000674
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_14] =0x0000005a     @0x00000678
Electronic_Fuse_Multiplier[CCURDPRC_CHANNEL_15] =0x0000005a     @0x0000067c
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_0]   =0x00005ae0     @0x00000680
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_1]   =0x00005ae0     @0x00000684
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_2]   =0x00005ae0     @0x00000688
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_3]   =0x00005ae0     @0x0000068c
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_4]   =0x00005ae0     @0x00000690
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_5]   =0x00005ae0     @0x00000694
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_6]   =0x00005ae0     @0x00000698
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_7]   =0x00005ae0     @0x0000069c
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_8]   =0x00005ae0     @0x000006a0
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_9]   =0x00005ae0     @0x000006a4
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_10]  =0x00005ae0     @0x000006a8
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_11]  =0x00005ae0     @0x000006ac
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_12]  =0x00005ae0     @0x000006b0
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_13]  =0x00005ae0     @0x000006b4
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_14]  =0x00005ae0     @0x000006b8
Electronic_Fuse_Threshold[CCURDPRC_CHANNEL_15]  =0x00005ae0     @0x000006bc
```

```
        spi_ram[0..63]
@0x0700 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x0720 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x0740 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x0760 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x0780 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x07a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x07c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
@0x07e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

3.1.4 ccurdprc_regedit

This is an interactive test to display and write to local, configuration and physical memory.

Usage: ./ccurdprc_regedit [-b board]
-b board: Board number -- default board is 0

Example display:

./ccurdprc_regedit

```
Device Name     : /dev/ccurdprc0
Board Serial No : 680593 (0x000a6291)
Initialize_Board: Firmware Rev. 0x01 successful

Virtual Address: 0x7ffff7ff6000
   1 = Create Physical Memory         2 = Destroy Physical memory
   3 = Display Driver Information     4 = Display Firmware RAM
   5 = Display Physical Memory Info   6 = Display Registers (CONFIG)
   7 = Display Registers (LOCAL)      8 = Dump Physical Memory
   9 = Reset Board                   10 = Write Register (LOCAL)
  11 = Write Register (CONFIG)       12 = Write Physical Memory

Main Selection ('h'=display menu, 'q'=quit)->
```

3.1.5 ccurdprc_tst

This is an interactive test to exercise some of the driver features.

Usage: ./ccurdprc_tst [-b board]
-b board: Board number -- default board is 0

Example display:

./ccurdprc_tst

```
Device Name     : /dev/ccurdprc0
Board Serial No : 680593 (0x000a6291)
Initialize_Board: Firmware Rev. 0x01 successful

  01 = add irq                       02 = disable pci interrupts
  03 = enable pci interrupts         04 = get device error
  05 = get driver info               06 = get physical mem
  07 = init board                    08 = mmap select
  09 = mmap(CONFIG registers)        10 = mmap(LOCAL registers)
  11 = mmap(physical memory)         12 = munmap(physical memory)
  13 = no command                    14 = read operation
  15 = remove irq                    16 = reset board
  17 = write operation

Main Selection ('h'=display menu, 'q'=quit)->
```

3.1.6 ccurdprc_wreg

This is a simple test to write to the local registers at the user specified offset.

```
Usage: ./ccurdprc_wreg [-b board] [-o offset] [-s size] [-v value] [-x]
 -b board : board selection --  default board is 0
 -o offset: hex offset to write to -- default offset is 0x0
 -s size  : number of bytes to write -- default size is 0x4
 -v value : hex value to write at offset -- default value is 0x0
 -x       : Do not read back just written values -- default read back values
```

Example display:

./ccurdprc_wreg -v12345678 -o0x700 –s100

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

Writing 0x12345678 to offset 0x0700 for 256 bytes

#### LOCAL REGS #### (length=256)
+LCL+    0x700   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x710   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x720   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x730   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x740   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x750   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x760   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x770   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x780   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x790   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7a0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7b0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7c0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7d0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7e0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
+LCL+    0x7f0   12345678  12345678  12345678  12345678 *.4Vx.4Vx.4Vx.4Vx*
```

3.1.7 Flash/ccurdprc_flash

This program is used to burn new firmware. This must only be done at the direction of Concurrent Real-Time support team; otherwise, they could render the board useless.

```
./ccurdprc_flash -[rw] -b[board] -q -s[start] -e[end] file_name
-b [board]        : board number. Default=-1
-e [end address]  : Default=0x408c83
-q                : Quite (non-interactive) mode
-r                : Read Flash and write to output file created by
./ccurdprc_flash
-s [start address]: Default=0x0
-w                : Read input file and Flash the board
Use either -r or -w to read or write the ccurdprc spi flash
The file_name is required

e.g. ./ccurdprc_flash -w -s 0x0 -e 0x408C89 -b0 FIRMWARE/CCURDPRC.bin
     ./ccurdprc_flash -r -s 0x0 -e 0x408c89 -b0 /tmp/CCURDPRC.out
```

3.1.8 Flash/ccurdprc_fwreload

This program reloads the firmware. This is normally performed after a new firmware is burnt.

```
./ccurdprc_fwreload -b[board]
-b [board]        : board number. Default=-1
```

```
e.g. ./ccurdprc_fwreload -b0
```

3.1.9 Eeprom/ccurdprc_eeprom

This program is used to burn new eeprom. This must only be done at the direction of Concurrent Real-Time support team; otherwise, they could render the board useless.

```
./ccurdprc_eeprom -b[board]
-b [board]        : board number. Default=-1

e.g. ./ccurdprc_eeprom -b0
```

Example display:

./ccurdprc_eeprom –b0

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

Dumping EEPROM: (0x00 - 0x3f)
@0x00:  9310 1542 0880 0001 0000 0100 0000 0000
@0x08:  0000 0000 ffff f800 0000 0001 0020 0000
@0x10:  0030 0400 0000 0000 0000 0000 4243 0343
@0x18:  0000 0000 0000 0000 0000 0000 0000 0000
@0x20:  0000 0000 9056 10b5 0000 0000 0000 0000
@0x28:  0000 0043 0000 4c00 0000 0000 0002 0000
@0x30:  0000 0000 0000 0000 0000 0000 0000 0000
@0x38:  0000 0000 0000 0000 0000 0000 0000 0100

device id           = 0x9310
vendor id           = 0x1542
subsystem device id = 0x9056
subsystem vendor id = 0x10b5
eeprom revision     = 0x0100
eeprom size         = 128 bytes
eeprom crc16        = 0x0000

  d = Dump EEPROM                   p = Pattern Fill EEPROM
  r = Restore EEPROM to default     w = Write EEPROM

Main Selection ('h'=display menu, 'q'=quit)->
```

3.2 Application Program Interface (API) Access Example Tests

These set of tests are located in the *.../test/lib* directory and use the API.

3.2.1 lib/ccurdprc_adc_calibrate

This utility can be used to perform AutoCalibration. Additionally, they can use this utility can also be used to display or write calibration information.

```
Usage: ./ccurdprc_adc_calibrate [-A] [-A!] [-b board] [-i inCalFile]
                                [-o outCalFile] [-R] [-W]
 -A                    (perform Auto Calibration)
 -A !                  (perform Auto Calibration only if any channel not
                        calibrated)
 -b <board>            (board #, default = 0)
 -i <In Cal File>      (input calibration file [input->board_reg])
 -o <Out Cal File>     (output calibration file [board_reg->output])
 -R                    (reset calibration registers)
 -W                    (wait for busy to clear before autocal - approx 60
```

```
                        seconds)
```

Example display:

./ccurdprc_adc_calibrate -A

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)
Auto Calibration started...done. (3.742 seconds)

===> Dump to 'stdout'
#Date           : Fri Jan 05 07:49:27 2018

#Chan   Negative                Offset                  Positive
#====   ========                ======                  ========
 ch00:  0.99923127004876732826  0.00061035156250000000  0.99922021990641951561
 ch01:  0.99947279226034879684  0.00000000000000000000  0.99952901387587189674
 ch02:  0.99920631432905793190  0.00000000000000000000  0.99925487162545323372
 ch03:  0.99905325518921017647  0.00000000000000000000  0.99904842255637049675
 ch04:  0.99838953465223312378 -0.00061035156250000000  0.99847391713410615921
 ch05:  0.99927610578015446663  0.00000000000000000000  0.99934815429151058197
 ch06:  0.99905750295147299767 -0.00122070312500000000  0.99922219337895512581
 ch07:  0.99851436214521527290  0.00000000000000000000  0.99861047510057687759
 ch08:  1.00050740223377943039  0.00061035156250000000  1.00050500780344009399
 ch09:  1.00066073611378669739  0.00000000000000000000  1.00072780018672347069
 ch10:  0.99857268156483769417  0.00000000000000000000  0.99864626117050647736
 ch11:  0.99877768149599432945 -0.00061035156250000000  0.99878046847879886627
 ch12:  0.99823356745764613152  0.00000000000000000000  0.99832826759666204453
 ch13:  0.99831963051110506058  0.00061035156250000000  0.99833270628005266190
 ch14:  1.00009105587378144264 -0.00122070312500000000  1.00013939756900072098
 ch15:  0.99985559144988656044  0.00000000000000000000  0.99989806674420833588
```

./ccurdprc_adc_calibrate -o OUTFILE

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)

===> Dump of 'OUTFILE' file
#Date           : Fri Jan 05 07:50:16 2018

#Chan   Negative                Offset                  Positive
#====   ========                ======                  ========
 ch00:  0.99923127004876732826  0.00061035156250000000  0.99922021990641951561
 ch01:  0.99947279226034879684  0.00000000000000000000  0.99952901387587189674
 ch02:  0.99920631432905793190  0.00000000000000000000  0.99925487162545323372
 ch03:  0.99905325518921017647  0.00000000000000000000  0.99904842255637049675
 ch04:  0.99838953465223312378 -0.00061035156250000000  0.99847391713410615921
 ch05:  0.99927610578015446663  0.00000000000000000000  0.99934815429151058197
 ch06:  0.99905750295147299767 -0.00122070312500000000  0.99922219337895512581
 ch07:  0.99851436214521527290  0.00000000000000000000  0.99861047510057687759
 ch08:  1.00050740223377943039  0.00061035156250000000  1.00050500780344009399
 ch09:  1.00066073611378669739  0.00000000000000000000  1.00072780018672347069
 ch10:  0.99857268156483769417  0.00000000000000000000  0.99864626117050647736
 ch11:  0.99877768149599432945 -0.00061035156250000000  0.99878046847879886627
 ch12:  0.99823356745764613152  0.00000000000000000000  0.99832826759666204453
 ch13:  0.99831963051110506058  0.00061035156250000000  0.99833270628005266190
 ch14:  1.00009105587378144264 -0.00122070312500000000  1.00013939756900072098
 ch15:  0.99985559144988656044  0.00000000000000000000  0.99989806674420833588

===> Board calibration data written to 'OUTFILE' file
```

./ccurdprc_adc_calibrate -i INFILE

```
Device Name    : /dev/ccurdprc0
Board Serial No: 680593 (0x000a6291)
===> Calibration data from 'INFILE' file written to board

===> Dump of 'INFILE' file
#Date           : Fri Jan 05 07:50:16 2018

#Chan   Negative                Offset                  Positive
#====   ========                ======                  ========
 ch00:  0.99923127004876732826  0.00061035156250000000  0.99922021990641951561
 ch01:  0.99947279226034879684  0.00000000000000000000  0.99952901387587189674
 ch02:  0.99920631432905793190  0.00000000000000000000  0.99925487162545323372
 ch03:  0.99905325518921017647  0.00000000000000000000  0.99904842255637049675
 ch04:  0.99838953465223312378 -0.00061035156250000000  0.99847391713410615921
 ch05:  0.99927610578015446663  0.00000000000000000000  0.99934815429151058197
 ch06:  0.99905750295147299767 -0.00122070312500000000  0.99922219337895512581
 ch07:  0.99851436214521527290  0.00000000000000000000  0.99861047510057687759
 ch08:  1.00050740223377943039  0.00061035156250000000  1.00050500780344009399
 ch09:  1.00066073611378669739  0.00000000000000000000  1.00072780018672347069
 ch10:  0.99857268156483769417  0.00000000000000000000  0.99864626117050647736
 ch11:  0.99877768149599432945 -0.00061035156250000000  0.99878046847879886627
 ch12:  0.99823356745764613152  0.00000000000000000000  0.99832826759666204453
 ch13:  0.99831963051110506058  0.00061035156250000000  0.99833270628005266190
 ch14:  1.00009105587378144264 -0.00122070312500000000  1.00013939756900072098
 ch15:  0.99985559144988656044  0.00000000000000000000  0.99989806674420833588

===> Dump to 'stdout'
#Date           : Fri Jan 05 07:51:26 2018

#Chan   Negative                Offset                  Positive
#====   ========                ======                  ========
 ch00:  0.99923127004876732826  0.00061035156250000000  0.99922021990641951561
 ch01:  0.99947279226034879684  0.00000000000000000000  0.99952901387587189674
 ch02:  0.99920631432905793190  0.00000000000000000000  0.99925487162545323372
 ch03:  0.99905325518921017647  0.00000000000000000000  0.99904842255637049675
 ch04:  0.99838953465223312378 -0.00061035156250000000  0.99847391713410615921
 ch05:  0.99927610578015446663  0.00000000000000000000  0.99934815429151058197
 ch06:  0.99905750295147299767 -0.00122070312500000000  0.99922219337895512581
 ch07:  0.99851436214521527290  0.00000000000000000000  0.99861047510057687759
 ch08:  1.00050740223377943039  0.00061035156250000000  1.00050500780344009399
 ch09:  1.00066073611378669739  0.00000000000000000000  1.00072780018672347069
 ch10:  0.99857268156483769417  0.00000000000000000000  0.99864626117050647736
 ch11:  0.99877768149599432945 -0.00061035156250000000  0.99878046847879886627
 ch12:  0.99823356745764613152  0.00000000000000000000  0.99832826759666204453
 ch13:  0.99831963051110506058  0.00061035156250000000  0.99833270628005266190
 ch14:  1.00009105587378144264 -0.00122070312500000000  1.00013939756900072098
 ch15:  0.99985559144988656044  0.00000000000000000000  0.99989806674420833588
```

3.2.2 lib/ccurdprc_disp

This is a powerful utility to not only display the various registers but control and test the board.

```
Usage: ./ccurdprc_disp [-a RollingAve] [-A] [-b BoardNo] [-c Chan] [-d Delay]
                       [-E ExpInpVolts] [-F DebugFile] [-l LoopCnt] [-r resistance]
                       [-s InputSignal] [-t Resistance] [-X]
  -a RollingAve           (Rolling average -- default =1000)
  -A                      (Perform Auto Calibration using reference voltage first)
  -b BoardNo              (select specific board, default = 0)
  -c ChanNo               (select specific channel, default = ALL CHANNELS)
  -d Delay                (Delay between screen refresh in milli-seconds -- default is 10)
  -E <ExpInpVolts>@<Tol>  (Expected Input Volts@Tolerance -- default Tol=0.006000)
     +@<Tol>              (Positive Calibration Ref Volt@Tolerance)
```

```
      -@<Tol>              (Negative Calibration Ref Volt@Tolerance)
      r@<Tol>              (Variable expected voltage based on selected resistance in '-t'
                            option)
   -f FaultGeneration      (select fault 'o','RaG','RbG','RaRbG','RaV','RbV','RaRbV',
                            'RaVRbG','RaGRbV')
      -f o                 (Ch0..15[ 0]=Open [fault or disabled])
      -f rag               (Ch0..15[17]=RA Ground Fault)
      -f RBg               (Ch0..15[18]=RB Ground Fault)
      -f rarbg             (Ch0..15[19]=RA & RB Ground Fault)
      -f raV               (Ch0..15[20]=RA Voltage Plus Fault)
      -f RBV               (Ch0..15[21]=RB Voltage Plus Fault)
      -f rarbV             (Ch0..15[22]=RA & RB Voltage Plus Fault)
      -f RavRbg            (Ch0..15[33]=RA Voltage Plus & RB Ground Fault Switch Test)
      -f RaGRbV            (Ch0..15[34]=RA Ground & RB Voltage Plus Fault Switch Test)
   -F DebugFile            (Menu display and write to debug file)
      @DebugFile           (No menu display. Only write to debug file)
      @                    (No menu display or write to debug file, only summary to stderr)
   -l LoopCnt              (Loop count -- default is 0)
   -r <Resistance>         (Program Potentiometer resistance (Range: 10-1000000)
   -s InputSignal          (select input signal, 'e', 'g', '+', '-', 't')
      -s e                 (Ch0..15=External input)
      -s g                 (Ch0..15=Ground Reference)
      -s t                 (Ch0..15=Positive 2.5 Volts Reference)
      -s +                 (Ch0..15=Positive 10 Volts Reference)
      -s -                 (Ch0..15=Negative 10 Volts Reference)
   -t <Resistance>@<Err>   (Perform Potentiometer Test using supplied resistance
                            (Range: 10-1000000) -- default Err=3.0%
   -X                      (Adjusted Measured Ohms for programmed Ohms between
                            45 and 1000000)

 Notes: Fault generation option 'f' and input signal option 's' not allowed together
        Potentiometer test option 't' and program resistance option 'r' not allowed together
        Potentiometer test option 't' and input signal option 's' not allowed together
        Potentiometer test option 't' and fault generation option 'f' not allowed together
        Option 't' % tolerance is with respect to the user supplied resistance

  e.g. ./ccurdprc_disp -t1240           (set all channels resistance to 1240 and display)
       ./ccurdprc_disp -t1240 -c4       (set channel 4 resistance to 1240 and display)
       ./ccurdprc_disp -t40 -Er         (set all channels resistance to 40 and validate
                                         expected voltage)
       ./ccurdprc_disp -t12345 -FDebug  (set all channels resistance to 12345 and output to
                                         Debug file)
       ./ccurdprc_disp -t1240 -X        (set all channels resistance to 1240 and display
                                         adjusted measured ohms)
       ./ccurdprc_disp -s+ -E10         (set all channels to 10V and validate expected
                                         voltage)
       ./ccurdprc_disp -se -r12345      (set all channels to external and program all
                                         channels to 12345 ohms)
       ./ccurdprc_disp -se -r12345 -c5  (set channel 5 to external and program channel 5 to
                                         12345 ohms)
```

Example display:

./ccurdprc_disp -A

```
Auto Calibration started...done. (4.773 seconds)

Specific Channel Selected[-c]: === All Channels Selected ===
Delay                  [-d]: 10 milli-seconds
Expected Input Volts   [-E]: === Not Specified ===
Loop Count             [-l]: ***Forever***
Calibration Bus Control    : 0 (Open)

                           : c00 c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 c11 c12 c13 c14 c15
                           : === === === === === === === === === === === === === === === ===
Digital Pot. Test (HexMask): 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
I/O Control (chan00..15)   :   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Scan Count                 : 2161
Read Duration (microsecs)  : TotalDelta:    33.127 (min=  32.827/max=  41.054/ave=  33.434)
```

```
    ##### ADC Channels (Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]    0000    FFFF    FFFF    0000    FFFF    FFFE    FFFE    FFFF    FFFF    FFFE
[1]    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF

    ##### ADC Channels (Volts) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] +0.0000 -0.0006 -0.0006 +0.0000 -0.0006 -0.0012 -0.0012 -0.0006 -0.0006 -0.0012
[1] -0.0006 -0.0006 -0.0006 -0.0006 -0.0006 -0.0006

    ------------------------------------------------------------------------------

    ##### Digital Potentiometer (Programmed Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]   32000   32000   32000   32000   32000   32000   32000   32000   32000   32000
[1]   32000   32000   32000   32000   32000   32000

    ##### Digital Potentiometer (Programmed Ohms) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000
[1] 1000000 1000000 1000000 1000000 1000000 1000000

            Expected Input Volts: === Not Specified ===

================== [ volts ] ====================
Chan  Min       Max       Ave      TolerExeededCnt
====  =======   =======   =======  ===============
 00   -0.0012    0.0006   -0.0003         -
 01   -0.0024    0.0000   -0.0006         -
 02   -0.0031    0.0000   -0.0006         -
 03   -0.0024    0.0000   -0.0005         -
 04   -0.0018    0.0000   -0.0004         -
 05   -0.0024    0.0000   -0.0006         -
 06   -0.0024    0.0006   -0.0007         -
 07   -0.0031    0.0000   -0.0006         -
 08   -0.0037    0.0012   -0.0008         -
 09   -0.0031    0.0012   -0.0009         -
 10   -0.0031    0.0006   -0.0004         -
 11   -0.0031    0.0012   -0.0004         -
 12   -0.0024    0.0006   -0.0006         -
 13   -0.0018    0.0000   -0.0005         -
 14   -0.0031    0.0012   -0.0007         -
 15   -0.0031    0.0012   -0.0006         -
================================================
```

./ccurdprc_disp -t12345 -X

```
Selected Ohms=12345, Actual Ohms=12344, Expected ADC Volts=9.081813
Computing Channel Voltage Error at 1 MegaOhm....done

==== Voltage Error Adjustment at 1 MegaOhm Resistance (for high ohms adjustment)
  Chan= 0 Error=+0.000386 volts
  Chan= 1 Error=+0.000514 volts
  Chan= 2 Error=+0.000489 volts
  Chan= 3 Error=+0.000666 volts
  Chan= 4 Error=+0.001069 volts
  Chan= 5 Error=+0.001197 volts
  Chan= 6 Error=+0.000978 volts
  Chan= 7 Error=+0.001002 volts
  Chan= 8 Error=+0.000630 volts
  Chan= 9 Error=+0.000343 volts
  Chan=10 Error=+0.000489 volts
  Chan=11 Error=+0.000801 volts
  Chan=12 Error=+0.000941 volts
  Chan=13 Error=+0.000727 volts
```

```
  Chan=14 Error=+0.000862 volts
  Chan=15 Error=+0.000849 volts
Computing Channel Ohms where expected=programmed....done

==== Point where Programmed Equal to Measured Ohm (for low ohms adjustment)
  Chan= 0 Ohm=960
  Chan= 1 Ohm=1023
  Chan= 2 Ohm=1333
  Chan= 3 Ohm=994
  Chan= 4 Ohm=558
  Chan= 5 Ohm=1115
  Chan= 6 Ohm=713
  Chan= 7 Ohm=481
  Chan= 8 Ohm=1207
  Chan= 9 Ohm=2573
  Chan=10 Ohm=960
  Chan=11 Ohm=1866
  Chan=12 Ohm=1517
  Chan=13 Ohm=1115
  Chan=14 Ohm=650
  Chan=15 Ohm=1333

 Specific Channel Selected[-c]: === All Channels Selected ===
 Delay                     [-d]: 10 milli-seconds
 Expected Input Volts      [-E]: === Not Specified ===
 Loop Count                [-l]: ***Forever***
 Potentiometer Test        [-t]: supplied 12345 Ohms (actual=12344) (Tolerance 3.000000% error
                                 wrt actual ohms)
 Adjusted Measured Ohms    [-X]: === Enabled ===)
 Calibration Bus Control       : 8 (+8 Milli-Ampere Current)

                               : c00 c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 c11 c12 c13 c14 c15
                               : === === === === === === === === === === === === === === === ===
 Digital Pot. Test (HexMask)   : 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
 I/O Control (chan00..15)      : (*** all 15 channels being set to test bus '7' in turn ***)

 Scan Count                    : 456
 Ohms Tolerance Exceeded Count: 0 (=== Passed ===)
 Read Duration (microsecs)     : TotalDelta: 55161.612
                                 (min=55105.278/max=55184.692/ave=55041.208)

    ##### ADC Channels (Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]    3A27    3A2A    3A28    3A2A    3A29    3A26    3A28    3A2A    3A27    3A25
[1]    3A2A    3A20    3A27    3A26    3A28    3A22

    ##### ADC Channels (Volts) ##### (Expected ADC Volts for Attached Resistance is 9.0818)

        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] +9.0867 +9.0883 +9.0875 +9.0886 +9.0880 +9.0860 +9.0874 +9.0887 +9.0869 +9.0852
[1] +9.0885 +9.0826 +9.0864 +9.0861 +9.0873 +9.0834

    --------------------------------------------------------------------------------

    ##### Digital Potentiometer (Programmed Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]   009E0   009E0   009E0   009E0   009E0   009E0   009E0   009E0   009E0   009E0
[1]   009E0   009E0   009E0   009E0   009E0   009E0

    ##### Digital Potentiometer (Programmed Ohms) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]   12344   12344   12344   12344   12344   12344   12344   12344   12344   12344
[1]   12344   12344   12344   12344   12344   12344

    ##### Digital Potentiometer (Adjusted Internal Measured Ohms) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
```

```
[0]   12410   12433   12421   12435   12420   12388   12411   12431   12410   12388
[1]   12436   12343   12398   12397   12413   12354

           Expected Input Volts: === Not Specified ===

=================== [ volts ] ====================
Chan  Min       Max       Ave      TolerExeededCnt
====  =======   =======   =======  ===============
 00    9.0865    9.0868    9.0867         -
 01    9.0882    9.0885    9.0883         -
 02    9.0873    9.0877    9.0875         -
 03    9.0884    9.0887    9.0886         -
 04    9.0879    9.0881    9.0880         -
 05    9.0858    9.0861    9.0859         -
 06    9.0872    9.0875    9.0873         -
 07    9.0885    9.0888    9.0887         -
 08    9.0867    9.0870    9.0869         -
 09    9.0850    9.0853    9.0852         -
 10    9.0883    9.0887    9.0885         -
 11    9.0825    9.0828    9.0827         -
 12    9.0862    9.0866    9.0864         -
 13    9.0858    9.0862    9.0860         -
 14    9.0871    9.0875    9.0873         -
 15    9.0833    9.0836    9.0834         -
================================================

          Potentiometer Resistance: supplied 12345 ohms (actual=12344) (Tolerance 3.000000 %
                                    error wrt actual ohms)
  Resistance Tolerence Exceed Count: 0
============================ [ resistance ] ============================
      <--------- (Ohms) -------->     <-- (% error wrt supplied ohms ) ->
Chan  Min       Max       Ave          Min    Max    Ave   TolerExeededCnt
====  =======   =======   =======      ====   ====   ====  ===============
 00     12408     12412     12409     0.52   0.55   0.53         -
 01     12430     12435     12432     0.70   0.74   0.71         -
 02     12419     12424     12420     0.61   0.65   0.62         -
 03     12432     12436     12434     0.71   0.75   0.73         -
 04     12418     12422     12420     0.60   0.63   0.62         -
 05     12385     12389     12387     0.33   0.36   0.35         -
 06     12409     12413     12410     0.53   0.56   0.53         -
 07     12428     12432     12430     0.68   0.71   0.70         -
 08     12407     12412     12409     0.51   0.55   0.53         -
 09     12386     12390     12388     0.34   0.37   0.36         -
 10     12433     12439     12436     0.72   0.77   0.75         -
 11     12341     12347     12344     0.02   0.02   0.00         -
 12     12395     12400     12397     0.41   0.45   0.43         -
 13     12393     12399     12395     0.40   0.45   0.41         -
 14     12410     12415     12412     0.53   0.58   0.55         -
 15     12352     12357     12354     0.06   0.11   0.08         -
================================================================================
```

./ccurdprc_disp -s+ -a0

```
 Rolling Average Count    [-a]: 1000
 Specific Channel Selected[-c]: === All Channels Selected ===
 Delay                    [-d]: 10 milli-seconds
 Expected Input Volts     [-E]: === Not Specified ===
 Loop Count               [-l]: ***Forever***
 Calibration Bus Control      : 2 (+10 Volts Reference)

                              : c00 c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 c11 c12 c13 c14 c15
                              : === === === === === === === === === === === === === === === ===
 Digital Pot. Test (HexMask)  : 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
 I/O Control (chan00..15)     : (*** all 15 channels being set to test bus '7' in turn ***)

 Scan Count                   : 1250
 Read Duration (microsecs)    : TotalDelta: 55191.640
(min=55083.646/max=55212.522/ave=55140.996)

    ##### ADC Channels (Raw Hex Data) (Rolling Average Count [1000/1000])  #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
```

```
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]    3FFF    3FFF    3FFF    3FFE    3FFF    3FFF    3FFF    3FFF    3FFE    3FFF
[1]    3FFF    3FFF    3FFF    3FFF    4000    3FFF

    ##### ADC Channels (Volts) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] +9.9997 +9.9998 +9.9998 +9.9994 +9.9997 +9.9999 +9.9998 +9.9997 +9.9994 +9.9997
[1] +9.9998 +9.9999 +9.9998 +9.9997+10.0000+10.0000

    ------------------------------------------------------------------------------

    ##### Digital Potentiometer (Programmed Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]   32000   32000   32000   32000   32000   32000   32000   32000   32000   32000
[1]   32000   32000   32000   32000   32000   32000

    ##### Digital Potentiometer (Programmed Ohms) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000
[1] 1000000 1000000 1000000 1000000 1000000 1000000

            Expected Input Volts: === Not Specified ===

=================== [ volts ] ====================
Chan  Min       Max       Ave      TolerExeededCnt
====  =======   =======   =======  ===============
 00    9.9998    9.9999    9.9999         -
 01    9.9998    9.9999    9.9998         -
 02    9.9995    9.9996    9.9996         -
 03    9.9997    9.9998    9.9998         -
 04    9.9999   10.0000   10.0000         -
 05    9.9997    9.9998    9.9998         -
 06    9.9998    9.9999    9.9998         -
 07    9.9998    9.9999    9.9999         -
 08    9.9998    9.9999    9.9999         -
 09    9.9994    9.9995    9.9995         -
 10    9.9995    9.9995    9.9995         -
 11    9.9999   10.0000   10.0000         -
 12    9.9996    9.9997    9.9997         -
 13    9.9996    9.9997    9.9996         -
 14    9.9998    9.9999    9.9998         -
 15    9.9998    9.9999    9.9998         -
====================================================
```

./ccurdprc_disp -f RavRbG

```
 Specific Channel Selected[-c]: === All Channels Selected ===
 Delay                    [-d]: 10 milli-seconds
 Expected Input Volts     [-E]: === Not Specified ===
 Loop Count               [-l]: ***Forever***
 Calibration Bus Control      : 0 (Open)

                              : c00 c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 c11 c12 c13 c14 c15
                              : === === === === === === === === === === === === === === === ===
 Digital Pot. Test (HexMask)  : 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
 I/O Control (chan00..15)     :  33  33  33  33  33  33  33  33  33  33  33  33  33  33  33  33

 Scan Count                   : 3515
 Read Duration (microsecs)    : TotalDelta:    33.184 (min=  32.809/max=  41.355/ave=  33.440)

    ##### ADC Channels (Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]    5EF4    5EF7    5F06    5ED9    5EEA    5ED9    5EDC    5EC8    5EE6    5F02
[1]    5EC6    5EF4    5F0B    5EFA    5EF3    5F0A

    ##### ADC Channels (Volts) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
```

```
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]+14.8364+14.8383+14.8474+14.8199+14.8303+14.8199+14.8218+14.8096+14.8279+14.8450
[1]+14.8083+14.8364+14.8505+14.8401+14.8358+14.8499

    ------------------------------------------------------------------------------

    ##### Digital Potentiometer (Programmed Raw Hex Data) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0]   32000   32000   32000   32000   32000   32000   32000   32000   32000   32000
[1]   32000   32000   32000   32000   32000   32000

    ##### Digital Potentiometer (Programmed Ohms) #####
        [0]     [1]     [2]     [3]     [4]     [5]     [6]     [7]     [8]     [9]
    ======= ======= ======= ======= ======= ======= ======= ======= ======= =======
[0] 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 1000000
[1] 1000000 1000000 1000000 1000000 1000000 1000000

            Expected Input Volts: === Not Specified ===

=================== [ volts ] ====================
Chan  Min       Max       Ave      TolerExeededCnt
====  =======   =======   =======  ===============
 00   14.8236   14.8370   14.8360         -
 01   14.8248   14.8395   14.8378         -
 02   14.8352   14.8486   14.8467         -
 03   14.8071   14.8218   14.8203         -
 04   14.8175   14.8322   14.8303         -
 05   14.8065   14.8218   14.8202         -
 06   14.8083   14.8230   14.8218         -
 07   14.7955   14.8108   14.8093         -
 08   14.8157   14.8303   14.8282         -
 09   14.8328   14.8462   14.8446         -
 10   14.7955   14.8108   14.8087         -
 11   14.8242   14.8389   14.8364         -
 12   14.8376   14.8511   14.8491         -
 13   14.8273   14.8413   14.8397         -
 14   14.8242   14.8413   14.8369         -
 15   14.8376   14.8529   14.8494         -
================================================
```

3.2.3 lib/ccurdprc_fault_protection

This utility is useful in displaying any Electronic Fuse Trip faults that may have occurred.

```
Usage: ./ccurdprc_fault_protection [-A] [-b BoardNo] [-c ChannelMask] [-d Delay]
                                   [-F DebugFile] [-l LoopCnt] [-R]
  -A                        (perform Auto Calibration)
  -b BoardNo                (select specific board, default = 0)
  -c ChannelMask            (Channel mask, default = 0xffff)
  -d Delay                  (Delay between screen refresh in milli-seconds -
                             default is 10)
  -F DebugFile              (Menu display and write to debug file)
     @DebugFile             (No menu display. Only write to debug file)
     @                      (No menu display or write to debug file, only summary
                             to stderr)
  -l LoopCnt                (Loop count -- default is 0)
  -R                        (Clear Fault Condition)
```

Example display:

./ccurdprc_fault_protection

```
 Channel Mask         [-c]: 0xffff
 Delay                [-d]: 10 milli-seconds
 Loop Count           [-l]: ***Forever***
```

```
Scan Count              : 332
Read Duration (microsecs): TotalDelta:   204.982 (min= 203.346/max= 215.011/ave= 206.281)

         Any Fuse Tripped: 0 (=== No ===)
             ADC 0 Failed: 0 (=== No ===)
             ADC 1 Failed: 0 (=== No ===)
Channel Fuse Tripped Mask: 0x0000
          Electrical Short: 0x00b4 (0.109863 volts)
                     Delay: 0x0002 (2)
                     Count: 0x0001 (1)
               I/O Delay: 0x0800 (31.000000 microseconds)
       Maximum Resistance: 0x00fd (1266.000000 ohms)
          Maximum Voltage: 0x5ae0 (14.199219 volts)
      Voltage Fault Delay: 0x0002 (8 microseconds)

           Fuse_Base        Fuse_Multiplier  Potentiometer    Threshold         IOS
Ch  Trip?  hex   volts      hex   volts      hex    ohms      hex   volts       hex
==  =====  ==============  ==============  ==============  ================  ===
 0   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 1   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 2   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 3   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 4   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 5   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 6   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 7   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 8   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
 9   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
10   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
11   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
12   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
13   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
14   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21
15   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)  5ae0 (14.199219)  21

Ch AnyTrip? CalNotValid? PotFail? ADCFail? VoltTrip? ADCComp? ADCValue? (hex/volts)
== ======== ============ ======== ======== ========= ======== ======================
 0   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 1   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 2   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 3   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 4   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 5   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 6   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 7   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 8   ---        ---        ---      ---      ---       ---    0000 (0.000000)
 9   ---        ---        ---      ---      ---       ---    0000 (0.000000)
10   ---        ---        ---      ---      ---       ---    0000 (0.000000)
11   ---        ---        ---      ---      ---       ---    0000 (0.000000)
12   ---        ---        ---      ---      ---       ---    0000 (0.000000)
13   ---        ---        ---      ---      ---       ---    0000 (0.000000)
14   ---        ---        ---      ---      ---       ---    0000 (0.000000)
15   ---        ---        ---      ---      ---       ---    0000 (0.000000)
```

3.2.4 lib/ccurdprc_fault_trip_test

This utility is only for validating the boards handling of Enetronic Fuse Trip handling.

```
Usage: ./ccurdprc_fault_trip_test [-A] [-b BoardNo] [-c ChannelMask]
                                  [-F DebugFile] [-l LoopCnt] [-R] [-t TestRun]
  -A                      (perform Auto Calibration)
  -b BoardNo              (select specific board, default = 0)
  -c ChannelMask          (Channel mask, default = 0xffff)
  -F DebugFile            (Menu display and write to debug file)
     @DebugFile           (No menu display. Only write to debug file)
     @                    (No menu display or write to debug file, only summary
                           to stderr)
  -l LoopCnt              (Loop count -- default is 1)
  -R                      (Clear Fault Condition)
```

```
  -t TestRun              (Test Run Option -- default is all options)
                          ( -t1 Calibration Not Valid Trip)
                          ( -t2 ADC Failure Trip)
                          ( -t3 ADC Compare Short Failure Trip)
                          ( -t4 ADC Compare Low Resistance Failure Trip)
                          ( -t5 Potentiometer Failure Trip)
                          ( -t6 Switch Voltage Failure Trip)
```

Example display:

```
############################ Loop    1 ############################
 1) Testing tripping condition for "Calibration Not Valid"
      Activating ADC and Potentiometer.....................passed
      Preserving Calibration for selected channels..........passed

      Case (A)
      ========
      Clearing Calibration for selected channels............passed
      Clearing Faults for selected channels      [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (B)
      ========
      Writing Potentiometer to selected Channels [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Writing Potentiometer to selected Channels [0x0002]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Writing Potentiometer to selected Channels [0x0004]...passed
       .
       .
       .
      Case (C)
      ========
      Restoring Calibration for selected channels...........passed
      Clearing Faults for selected channels      [0x0001]...passed
      Writing Potentiometer to selected Channels [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Clearing Faults for selected channels      [0x0002]...passed
      Writing Potentiometer to selected Channels [0x0002]...passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Restoring Calibration for selected channels...........passed
    ==== PASSED ====
 2) Test tripping condition for "ADC Failure"..............
      Activating ADC and Potentiometer.....................passed

      Case (A)
      ========
      Disabling both ADCs..................................passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (B)
      ========
      Writing Potentiometer to selected Channels [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Writing Potentiometer to selected Channels [0x0002]...passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Case (C)
```

```
      ========
      Enabling all ADCs.....................................passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (D)
      ========
      Clearing Faults for selected channels      [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Clearing Faults for selected channels      [0x0002]...passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Clearing Faults for selected channels      [0x8000]...passed
      Validating Faults for selected channels    [0xffff]...passed
    ==== PASSED ====
 3) Test tripping condition for "ADC Compare Short Failure".
      Activating ADC and Potentiometer......................passed

      Case (A) - Calibration Control is Open
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Setting Electronic Fuse Short Value to 43.............passed
      Setting Electronic Fuse Short Value to 42.............passed
.
.
.
      Setting Electronic Fuse Short Value to  2.............passed
      Setting Electronic Fuse Short Value to  1.............passed
      Tripped Channels Mask                      [0x0b00]...passed
      Setting Electronic Fuse Short Value to  0.............passed
      Tripped Channels Mask                      [0xffff]...passed
        1: Tripped Channels Mask                 [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Activating ADC and Potentiometer......................passed

      Case (B) - Calibration Control is Minus 8 Milli-Amps
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Tripped Channels Mask                      [0xff7f]...passed
      Setting Electronic Fuse Short Value to 43.............passed
      Tripped Channels Mask                      [0xff7f]...passed
.
.
.
      Setting Electronic Fuse Short Value to 17.............passed
      Tripped Channels Mask                      [0xffff]...passed
        1: Tripped Channels Mask                 [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Activating ADC and Potentiometer......................passed

      Case (C) - Calibration Control is Plus 8 Milli-Amps
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Tripped Channels Mask                      [0xffff]...passed
        1: Tripped Channels Mask                 [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Activating ADC and Potentiometer......................passed

      Case (D) - Calibration Control is Plus 2.5 Volts
```

```
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Setting Electronic Fuse Short Value to 43.............passed
      Setting Electronic Fuse Short Value to 42.............passed
.
.
.
      Setting Electronic Fuse Short Value to  2.............passed
      Setting Electronic Fuse Short Value to  1.............passed
      Tripped Channels Mask                       [0x4300]...passed
      Setting Electronic Fuse Short Value to  0.............passed
      Tripped Channels Mask                       [0xffff]...passed
        1: Tripped Channels Mask                  [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Activating ADC and Potentiometer.......................passed

      Case (E) - Calibration Control is Minus 10 Volts
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Tripped Channels Mask                       [0xffff]...passed
        1: Tripped Channels Mask                  [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Activating ADC and Potentiometer.......................passed

      Case (F) - Calibration Control is Plus 10 Volts
      ========
      Writing 10 Ohms (short) to all Channels    [0xffff]...passed
      Setting Electronic Fuse Short Value to 44.............passed
      Tripped Channels Mask                       [0xffff]...passed
        1: Tripped Channels Mask                  [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed
    ==== PASSED ====
 4) Test tripping condition for "ADC Compare Low Resistance Failure".
      Activating ADC and Potentiometer.......................passed

      Case (A) - Low Resistance   45 Ohms
      ========
      Writing   45 Ohms (short) to all Channels  [0xffff]...passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (B) - Low Resistance   45 Ohms
      ========
      Setting Fuse Base to 0x0192 for channel  0.............passed
      Validating Faults for selected channels    [0xffff]...passed
      Setting Fuse Base to 0x0191 for channel  1.............passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Setting Fuse Base to 0x0195 for channel 15.............passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (C) - Low Resistance   45 Ohms
      ========
      Clearing Faults for selected channels      [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Clearing Faults for selected channels      [0x8000]...passed
      Validating Faults for selected channels    [0xffff]...passed
```

```
Case (D) - Low Resistance   45 Ohms
========
Writing   45 Ohms (short) to Channel       [0x0001]...passed
Validating Faults for selected channels    [0xffff]...passed
 .
 .
 .
Writing   45 Ohms (short) to Channel       [0x8000]...passed
Validating Faults for selected channels    [0xffff]...passed

Case (E) - Low Resistance   45 Ohms
========
Clearing Faults for selected channels      [0x0001]...passed
Setting Fuse Base/Multiplier defaults for channel  0..passed
Writing   45 Ohms (short) to Channel       [0x0001]...passed
Validating Faults for selected channels    [0xffff]...passed
 .
 .
 .
Case (A) - Low Resistance 1265 Ohms
========
Writing 1265 Ohms (short) to all Channels  [0xffff]...passed
Validating Faults for selected channels    [0xffff]...passed

Case (B) - Low Resistance 1265 Ohms
========
Setting Fuse Multiplier to 0x001e for channel  0......passed
Validating Faults for selected channels    [0xffff]...passed
Setting Fuse Multiplier to 0x001e for channel  1......passed
Validating Faults for selected channels    [0xffff]...passed
 .
 .
 .
Setting Fuse Multiplier to 0x001e for channel 15......passed
Validating Faults for selected channels    [0xffff]...passed

Case (C) - Low Resistance 1265 Ohms
========
Clearing Faults for selected channels      [0x0001]...passed
Validating Faults for selected channels    [0xffff]...passed
 .
 .
 .
Clearing Faults for selected channels      [0x8000]...passed
Validating Faults for selected channels    [0xffff]...passed

Case (D) - Low Resistance 1265 Ohms
========
Writing 1265 Ohms (short) to Channel       [0x0001]...passed
Validating Faults for selected channels    [0xffff]...passed
 .
 .
 .
Writing 1265 Ohms (short) to Channel       [0x8000]...passed
Validating Faults for selected channels    [0xffff]...passed

Case (E) - Low Resistance 1265 Ohms
========
Clearing Faults for selected channels      [0x0001]...passed
Setting Fuse Base/Multiplier defaults for channel  0..passed
Writing 1265 Ohms (short) to Channel       [0x0001]...passed
Validating Faults for selected channels    [0xffff]...passed
```

```
       .
       .
       .
      Setting Fuse Base/Multiplier defaults for channel 15..passed
      Writing 1265 Ohms (short) to Channel       [0x8000]...passed
      Validating Faults for selected channels    [0xffff]...passed
    ==== PASSED ====
 5) Test tripping condition for "Potentiometer Failure" ....
      Activating ADC and Potentiometer.....................passed

      Case (A)
      ========
      Force Potentiometer Failure on all channels...........passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (B)
      ========
      Writing Potentiometer to selected Channels [0x0001]...passed
      Validating Faults for selected channels    [0xffff]...passed
      Writing Potentiometer to selected Channels [0x0002]...passed
      Validating Faults for selected channels    [0xffff]...passed
       .
       .
       .
      Writing Potentiometer to selected Channels [0x8000]...passed
      Validating Faults for selected channels    [0xffff]...passed

      Case (C)
      ========
      Activating ADC and Potentiometer.....................passed
      Validating Faults for selected channels    [0xffff]...passed
    ==== PASSED ====
 6) Test tripping condition for "Switch Voltage Failure"....(ToBeCoded)
```

3.2.5 lib/ccurdprc_identify

This test is useful in identifying a card by displaying its LED.

```
Usage: ./ccurdprc_identify -[absx]
          -a                    (Identify all cards through a light sequence)
          -b <board>            (board #, default = 0)
          -s <seconds)          (Identify Board: ENABLED for number of seconds,
                                 default = 10)
          -s 0                  (Identify Board: DISABLED)
          -s <negative value>   (Identify Board: ENABLED forever)
          -x                    (silent)

If the '-a' option is selected, all other options are ignored. This option will
sequence through all the cards found in turn as follows:
  1) The first device number will flash its LED for 10 seconds
  2) The remaining devices numbers will be selected sequentially and flash their
LEDs for 3 seconds
```

Example display:

./ccurdprc_identify

```
Device Name    : /dev/ccurdprc0
Board ID       : 9310
Board Type     : 01
Board NumChans : 16
Board Cal Volts: 10.000000
Board Serial No: 680584 (0x000a6288)
```

```
Identify ENABLED on board 0 (LED should start flashing for 10 seconds)
Sleeping for 10 seconds...
Identify DISABLED on board 0 (LED should stop flashing)
```

3.2.6 lib/ccurdprc_info

This test is useful in getting information for all the *ccurdprc* devices in the system.

```
Usage: ./ccurdprc_info -[bpv]
          -b <board>   (board #, default = 0)
          -p <a|d>     (Program Activate(a)/Disable(d) All, default = no program)
          -v           (Verbose, default = no verbose)
```

Example display:

./ccurdprc_info

```
                       Version:  23.1.1
                         Build:  Apr 14 2020, 12:33:40
                        Module:  ccurdprc
                   Board Index:  0 (PLX-CCURDPRC)
                Board Serial No:  680593 (0x000a6291)
                Serial Prom Rev:  0x0000
                           Bus:  8
                          Slot:  4
                          Func:  0
                     Vendor ID:  0x1542
                   Sub-Vendor ID:  0x10b5
                        Board ID:  0x9310
                     Board Type:  0x0001
                   Sub-Device ID:  0x9056
                      Board Info:  0x93100102
                     MSI Support:  Enabled
                       IRQ Level:  55
                        Firmware:  0x0001
                Interrupt Count:  0
               Interrupt Status:  0x0000
             Number of Channels:  16
               All Channel Mask:  0xffff
 Calibration Reference Voltage:  10.000000 volts
                  Voltage Range:  0.000000 volts
                       Region 0:  Addr=0xc4c01000  Size=512   (0x200)
                       Region 2:  Addr=0xc4c00000  Size=2048  (0x800)
       Calibration Bus Control:  00 (0x0) Bus Open
                    I/O Control:  === Disabled ===
                  Potentiometer:  === Disabled ===
             Potentiometer Test:  === Disabled ===
   Analog to Digital Converter:  === Disabled ===
               Any Fuse Tripped:  === No ===
                   ADC 0 Failed:  ### Yes ###
                   ADC 1 Failed:  ### Yes ###
               Electrical Short:  0x00b4 (0.109863 volts)
                          Delay:  0x0002 (2)
                          Count:  0x0001 (1)
                      I/O Delay:  0x0800 (31.000000 volts)
             Maximum Resistance:  0x00fd (1266.000000 ohms)
                Maximum voltage:  0x5ae0 (14.199219 volts)
            Voltage Fault Delay:  0x0002 (8 microseconds)
```

./ccurdprc_info -v

```
                       Version:  23.1.1
                         Build:  Apr 14 2020, 12:33:40
                        Module:  ccurdprc
                   Board Index:  0 (PLX-CCURDPRC)
                Board Serial No:  680593 (0x000a6291)
                Serial Prom Rev:  0x0000
                           Bus:  8
                          Slot:  4
```

```
                          Func:  0
                     Vendor ID:  0x1542
                Sub-Vendor ID:  0x10b5
                     Board ID:  0x9310
                   Board Type:  0x0001
                Sub-Device ID:  0x9056
                   Board Info:  0x93100102
                  MSI Support:  Enabled
                    IRQ Level:  55
                     Firmware:  0x0001
              Interrupt Count:  0
            Interrupt Status:  0x0000
          Number of Channels:  16
            All Channel Mask:  0xffff
 Calibration Reference Voltage:  10.000000 volts
               Voltage Range:  0.000000 volts
                     Region 0:  Addr=0xc4c01000  Size=512   (0x200)
                     Region 2:  Addr=0xc4c00000  Size=2048  (0x800)
       Calibration Bus Control:  00 (0x0) Bus Open
                               :
       Calibration Information:  Negative                 Offset                   Positive
       ======================:  =====================  =====================
======================
                     Channel  0:  0.99923393130302429199  0.00061035156250000000
0.99926293526589870 45
                     Channel  1:  0.99947733711451292038  0.00000000000000000000
0.99953469820320606232
                     Channel  2:  0.99920863285660743713  0.00000000000000000000
0.99925949377939105034
                     Channel  3:  0.99905715836212038994  0.00000000000000000000
0.99905238440260291100
                     Channel  4:  0.99838956817984580994 -0.00061035156250000000
0.99847721355035901070
                     Channel  5:  0.99927859148010611534  0.00000000000000000000
0.99935070564970374107
                     Channel  6:  0.99905843008309602737 -0.00122070312500000000
0.99922330724075436592
                     Channel  7:  0.99851754121482372284  0.00000000000000000000
0.99861228326335549355
                     Channel  8:  1.00050913682207465172  0.00061035156250000000
1.00050572864711284637
                     Channel  9:  1.00065903877839446068  0.00000000000000000000
1.00072941137477755547
                     Channel 10:  0.99857431231066584587  0.00000000000000000000
0.99864789657294750214
                     Channel 11:  0.99878009874373674393 -0.00061035156250000000
0.99878242751583456993
                     Channel 12:  0.99823445873335003853  0.00000000000000000000
0.99833095585927367210
                     Channel 13:  0.99832039466127753258  0.00061035156250000000
0.99833637848496437073
                     Channel 14:  1.00009206961840391159 -0.00122070312500000000
1.00013942923396825790
                     Channel 15:  0.99985691532492637634  0.00000000000000000000
0.99989738129079341888
                                :
                    I/O Control:  Value      Description
                    ===========:  =========  ===========
                     Channel  0:  00 (0x00)  Open
                     Channel  1:  00 (0x00)  Open
                     Channel  2:  00 (0x00)  Open
                     Channel  3:  00 (0x00)  Open
                     Channel  4:  00 (0x00)  Open
                     Channel  5:  00 (0x00)  Open
                     Channel  6:  00 (0x00)  Open
                     Channel  7:  00 (0x00)  Open
                     Channel  8:  00 (0x00)  Open
                     Channel  9:  00 (0x00)  Open
                     Channel 10:  00 (0x00)  Open
                     Channel 11:  00 (0x00)  Open
                     Channel 12:  00 (0x00)  Open
```

```
                Channel 13:  00 (0x00)  Open
                Channel 14:  00 (0x00)  Open
                Channel 15:  00 (0x00)  Open
                           :
             Potentiometer:  Value (Raw)  Ohms
             =============:  ===========  ====
                Channel  0:  204800       1000000
                Channel  1:  204800       1000000
                Channel  2:  204800       1000000
                Channel  3:  204800       1000000
                Channel  4:  204800       1000000
                Channel  5:  204800       1000000
                Channel  6:  204800       1000000
                Channel  7:  204800       1000000
                Channel  8:  204800       1000000
                Channel  9:  204800       1000000
                Channel 10:  204800       1000000
                Channel 11:  204800       1000000
                Channel 12:  204800       1000000
                Channel 13:  204800       1000000
                Channel 14:  204800       1000000
                Channel 15:  204800       1000000
                           :
          Potentiometer Test:  Value (Mask)  Description
          ==================:  ============  ===========
                  Channel  0:  0x00000100    20K Potentiometer Mode
                  Channel  1:  0x00000100    20K Potentiometer Mode
                  Channel  2:  0x00000100    20K Potentiometer Mode
                  Channel  3:  0x00000100    20K Potentiometer Mode
                  Channel  4:  0x00000100    20K Potentiometer Mode
                  Channel  5:  0x00000100    20K Potentiometer Mode
                  Channel  6:  0x00000100    20K Potentiometer Mode
                  Channel  7:  0x00000100    20K Potentiometer Mode
                  Channel  8:  0x00000100    20K Potentiometer Mode
                  Channel  9:  0x00000100    20K Potentiometer Mode
                  Channel 10:  0x00000100    20K Potentiometer Mode
                  Channel 11:  0x00000100    20K Potentiometer Mode
                  Channel 12:  0x00000100    20K Potentiometer Mode
                  Channel 13:  0x00000100    20K Potentiometer Mode
                  Channel 14:  0x00000100    20K Potentiometer Mode
                  Channel 15:  0x00000100    20K Potentiometer Mode
Analog to Digital Converter: Value (Hex)  Volts
===========================: ===========  =====
                  Channel  0: 0x0000      +0.000000
                  Channel  1: 0xffff      -0.000610
                  Channel  2: 0xffff      -0.000610
                  Channel  3: 0x0000      +0.000000
                  Channel  4: 0xffff      -0.000610
                  Channel  5: 0xffff      -0.000610
                  Channel  6: 0xffff      -0.000610
                  Channel  7: 0xffff      -0.000610
                  Channel  8: 0xffff      -0.000610
                  Channel  9: 0xffff      -0.000610
                  Channel 10: 0xffff      -0.000610
                  Channel 11: 0xffff      -0.000610
                  Channel 12: 0xffff      -0.000610
                  Channel 13: 0xffff      -0.000610
                  Channel 14: 0xffff      -0.000610
                  Channel 15: 0x0000      +0.000000
            Any Fuse Tripped:  === No ===
                ADC 0 Failed:  === No ===
                ADC 1 Failed:  === No ===
              Electrical Short:  0x00b4 (0.109863 volts)
                         Delay:  0x0002 (2)
                         Count:  0x0001 (1)
                   I/O Delay:  0x0800 (31.000000 volts)
           Maximum Resistance:  0x00fd (1266.000000 ohms)
              Maximum voltage:  0x5ae0 (14.199219 volts)
          Voltage Fault Delay:  0x0002 (8 microseconds)
                           :
```

```
                                :         Fuse_Base        Fuse_Multiplier  Potentiometer
Threshold         IOS
     Channel Fault Information:  Trip?  hex   volts      hex   volts      hex    ohms
hex   volts       hex
     ========================:  =====  ===============  ===============  ===============
================  ===
                      Channel  0:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  1:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  2:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  3:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  4:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  5:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  6:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  7:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  8:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel  9:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 10:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 11:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 12:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 13:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 14:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                      Channel 15:   ---   0241 (0.352173)  005a (0.054932)  32000 (1000000)
5ae0 (14.199219)  00
                                :
     Channel Fault Information:  AnyTrip? CalNotValid? PotFail? ADCFail? VoltTrip? ADCComp?
ADCValue? (hex/volts)
     ========================:  ======== ============ ======== ======== ========= ========
=====================
                      Channel  0:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  1:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  2:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  3:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  4:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  5:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  6:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  7:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  8:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel  9:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel 10:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel 11:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel 12:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                      Channel 13:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
```

```
                    Channel 14:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
                    Channel 15:   ---        ---        ---      ---      ---       ---
0000 (0.000000)
```

3.2.7 lib/ccurdprc_tst_lib

This is an interactive test that accesses the various supported API calls.

```
Usage: ./ccurdprc_tst_lib [-b board]
 -b board: board number -- default board is 0
```

Example display:

./ccurdprc_tst_lib

```
Device Name: /dev/ccurdprc0
  01 = Abort DMA                          02 = Clear Driver Error
  03 = Clear Library Error                04 = Display BOARD Registers
  05 = Display CONFIG Registers           06 = Get Board CSR
  07 = Get Board Information              08 = Get Driver Error
  09 = Get Driver Information             10 = Get Driver Read Mode
  11 = Get Driver Write Mode              12 = Get Library Error
  13 = Get Mapped Config Pointer          14 = Get Mapped Driver/Library Pointer
  15 = Get Mapped Local Pointer           16 = Get Physical Memory
  17 = Get Value                          18 = Initialize Board
  19 = MMap Physical Memory               20 = Munmap Physical Memory
  21 = Read Operation                     22 = Read Channels
  23 = Reset Board                        24 = Select Driver Read Mode
  25 = Select Driver Write Mode           26 = Set Board CSR
  27 = Set Value                          28 = ### ADC CONTROL MENU ###
  29 = ### CALIBRATION MENU ###           30 = ### DIGITAL POT AND I/O CONTROL MENU
  31 = ### ELECTRONIC FUSE CONTROL MENU ### 32 = ### INTERRUPT MENU ###
  33 = ### SERIAL PROM MENU ###

Main Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 28
   Command: ADC_control_menu()
  01 = ADC Activate                       02 = ADC Disable
  03 = ADC Read Channels

ADC Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 29
   Command: calibration_menu()
  01 = Get Calibrated Values              02 = Get Calibration Bus Control
  03 = Perform Auto Calibration           04 = Perform External Negative Calib.
  05 = Perform External Offset Calib.     06 = Perform External Positive Calib.
  07 = Perform Negative Calibration       08 = Perform Offset Calibration
  09 = Perform Positive Calibration       10 = Read calibration channels
  11 = Reset Calibration                  12 = Write Channels Calibration
  13 = Set Calibration Bus Control

Calibration Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 30
   Command: DigitalPotAndIo_control_menu()
  01 = Digital Potentiometer & I/O Activate 02 = Digital Potentiometer & I/O Disable
  03 = Digital Potentiometer Get Resistance 04 = Digital Potentiometer Get Test
  05 = Digital Potentiometer Set Resistance 06 = Digital Potentiometer Set Test
  07 = I/O Control Get                    08 = I/O Control Set

Digital Pot and I/O Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 31
   Command: ElectronicFuse_control_menu()
  01 = Clear Electronic Fuse Trip         02 = Dump Electronic Fuse Registers
  03 = Get Electronic Fuse Base           04 = Get Electronic Fuse Internals
```

```
  05 = Get Electronic Fuse Multiplier       06 = Get Electronic Fuse Status
  07 = Get Electronic Fuse Threshold        08 = Get Electronic Fuse Trip

Electronic Fuse Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 32
   Command: interrupt_menu()
  01 = Add Irq                              02 = Disable Pci Interrupts
  03 = Enable Pci Interrupts                04 = Get Interrupt Control
  05 = Get Interrupt Status                 06 = Get Interrupt Timeout
  07 = Remove Irq                           08 = Set Interrupt Control
  09 = Set Interrupt Status                 10 = Set Interrupt Timeout

Interrupt Selection ('h'=display menu, 'q'=quit)->
```

```
Main Selection ('h'=display menu, 'q'=quit)-> 33
   Command: serial_prom_menu()
  01 = Clear Serial Prom                    02 = Read Serial PROM
  03 = Serial PROM Write Override           04 = Write Serial PROM

Serial PROM Selection ('h'=display menu, 'q'=quit)->
```

3.2.8 lib/Sprom/ccurdprc_sprom

This is a simple program to demonstrate sprom access.

```
Usage: ./ccurdprc_sprom [-b board] [-C] [-D] [-S serialNo]
 -b <board>            (Board #, default = 0)
 -C                    (Clear ENTIRE serial PROM first)
 -D                    (Dump entire serial prom)
 -S <serialNo>         (Program board serial number)

   e.g.  ./ccurdprc_sprom -C                 -> Clear Entire Serial Prom First
   e.g.  ./ccurdprc_sprom -D                 -> Dump Entire Serial Prom
   e.g.  ./ccurdprc_sprom -S 12345678        -> Write Serial Number
```

Example display:

./Sprom/ccurdprc_sprom

```
Device Name:          /dev/ccurdprc0
Board Serial Number:  680593 (0x000a6291)
Serial PROM Revision: 0 (0x0000)
```

This page intentionally left blank