

# Digilent Adept Asynchronous Communications Interface (DACI) Programmer's Reference Manual

Revision: August 20, 2010



1300 NE Henley Court, Suite 3  
Pullman, WA 99163  
(509) 334 6306 Voice | (509) 334 6300 Fax

## Introduction

This document describes the Digilent Adept Asynchronous Communications Interface (DACI) subsystem for version 2 of the Digilent Adept software system. This document describes the capabilities of the DACI subsystem and the API functions used to access its features.

The DACI subsystem provides access to asynchronous serial communications (UART) ports.

## DACI Port Properties

The port property bits are used to indicate which parts of the DACI interface specification are supported by a given port.

The following port properties bits are defined for DACI ports: These values are defined in the header file *daci.h*.

dprpAciDte	This bit indicates that the port implements an RS232 DTE device.
dprpAciDce	This bit indicates that the port implements an RS232 DCE device.
dprpAciRtsCts	This bit indicates that the port supports RTS/CTS handshaking for flow control.
dprpAciXonXoff	This bit indicates that the port supports Xon/Xoff handshaking for flow control.
dprpAciBaudRate	This bit indicates that the port supports setting the baud rate.
dprpAciStopBits	This bit indicates that the port supports setting the number of stop bits used.
dprpAciDataBits	This bit indicates that the port supports setting the number of data bits per character sent or received.
dprpAciParityNone	This bit indicates that the port supports setting parity to none.
dprpAciParityEven	This bit indicates that the port supports setting parity to even.
dprpAciParityOdd	This bit indicates that the port supports setting parity to odd.
dprpAciParityMark	This bit indicates that the port supports setting parity to mark.
dprpAciParitySpace	This bit indicates that the port supports setting parity to space.

## DACI API Functions

The following API functions make up the DACI interface.

### **DaciGetVersion(char \* szVersion)**

*Parameters:*

szVersion                    - pointer to buffer to receive version string

This function returns a version number string identifying the version number of the DACI DLL. The symbol cchVersionMax declared in dpcdecl.h defines the longest string that can be returned in *szVersion*.

### **DaciGetPortCount(HIF hif, INT32 \* pcprt)**

*Parameters:*

hif                            - open interface handle on the device  
pcprt                         - pointer to variable to receive count of ports

This function returns the number of DACI ports supported by the device specified by *hif*.

### **DaciGetPortProperties(HIF hif, INT32 prtReq, DWORD \* pdprp)**

*Parameters:*

hif                            - open interface handle on the device  
prtReq                        - port number to query  
pdprp                         - pointer to variable to return port property bits

This function returns the port properties bits for the specified DACI port. The port properties bits indicate the specific set of optional features implemented by the port.

### **DaciEnable(HIF hif)**

*Parameters:*

hif                            - open interface handle on the device

This function is used to enable the default DACI port (port 0) on the specified device. This function must be called before any functions that operate on the port may be called for the specified device.

The default communications properties for a DACI port when enabled are: 9600 baud, 8 data bits, 1 stop bit, no parity.

**DaciEnableEx(HIF hif, INT32 prtReq)***Parameters:*

hif	- open interface handle on the device
prtReq	- DACI port number

This function is used to enable a specific port on devices that support multiple DACI ports. This function must be called before any functions that operate on the ACI port may be called. The *prtReq* parameter specifies the port number of the DACI port to enable.

The default communications properties for an ACI port when enabled are: 9600 baud, 8 data bits, 1 stop bit, no parity.

**DaciDisable(HIF hif)***Parameters:*

hif	- open interface handle on the device
-----	---------------------------------------

This function is used to disable and end access to the DACI port currently enabled on the specified interface handle.

**DaciGetMode(HIF hif, INT32 \* pcbtData, INT32 \* pidStop, INT32 \* pidParity)***Parameters:*

hif	- open interface handle on the device
pcbtData	- variable to receive number of data bits
pidStop	- pointer to variable to receive number of stop bits
pidParity	- pointer to variable to receive parity setting

This function is used to query the current mode settings for the communications port associated with the specified interface handle. The allowed return values are the same as the values described below for the DaciSetMode function, and are defined in the header file *daci.h*.

**DaciSetMode(HIF hif, INT32 cbtData, INT32 idStop, INT32 idParity)***Parameters:*

hif	- open interface handle on the device
cbtData	- requested number of data bits
idStop	- requested number of stop bits
idParity	- requested parity setting

This function is used to set the operating mode of the communications port associated with the specified interface handle.

The *cbtData* parameter specifies the number of data bits per transmission element. The allowed range of values is: *cbtAciDataMin* to *cbtAciDataMax* (5 to 8), however not all ports support the full range of values.

The *idStop* parameter specifies the number of stop bits per transmission element. The allowed values are: *idAciOneStopBit*, *idAciOne5StopBit*, *idAciTwoStopBit*, however not all ports support all of these values. The values *idAciOneStopBit* and *idAciTwoStopBit* request one stop bit and two stop bits respectively. The value *idAciOne5StopBit* requests 1.5 stop bits. If this value is requested and the port doesn't support 1.5 stop bits, 2 stop bits will be selected.

The *idParity* parameter specifies the parity mode to use. The allowed values are: *idAciParityNone*, *idAciParityOdd*, *idAciParityEven*, *idAciParityMark*, *idAciParitySpace*, however not all ports support all of these values.

**DaciGetBaud(HIF hif, ULONG \* pbdrCur)***Parameters:*

hif	- open interface handle on the device
pbdrCur	- pointer to variable to receive current baud rate

This function returns the baud rate currently set in the port associated with *hif*.

**DaciSetBaud(HIF hif, ULONG bdrReq, ULONG \* pbdrSet)***Parameters:*

hif	- open interface handle on the device
bdrReq	- requested baud rate
pbdrSet	- pointer to variable to receive baud rate set

This function is used to set the baud rate in the port associated with *hif* to the requested value. The baud rate to be set is specified by *bdrReq*. The actual value set is returned in the variable specified by *pbdrSet*. The baud rate will be set to the highest supported baud rate that doesn't exceed the requested baud rate. If there is no supported baud rate that doesn't exceed the requested baud rate, then the lowest supported baud rate is set.

**DaciGetBufferSize(HIF hif, ULONG \* pcbTxb, ULONG \* pcbRxb)***Parameters:*

hif	- open interface handle on the device
pcbTx	- variable to receive size of transmit buffer
pcbRx	- variable to receive size of receive buffer

The function returns the sizes of the transmit and receive fifo buffers.

### **DaciSetRtsCtsEnable(HIF hif, BOOL fEnable)**

*Parameters:*

hif	- open interface handle on the device
fEnable	- enable (TRUE) or disable (FALSE) handshake

This function will enable/disable RTS/CTS handshaking on the port. This function is ignored for ports that don't support RTS/CTS handshaking. This is indicated by the state of the dprpAciRtsCts bit in the port property value for the port.

### **DaciSetXonXoffEnable(HIF hif, BOOL fEnable)**

*Parameters:*

hif	- open interface handle on the device
fEnable	- enable (TRUE) or disable (FALSE) handshake

This function will enable/disable XON/XOFF handshaking on the port. This function is ignored for ports that don't support XON/OFF handshaking. This is indicated by the state of the dprpAciXonXoff bit in the port property value for the port.

### **DaciQueryStatus(HIF hif, ULONG \* pcbTx, ULONG \* pcbRx, DWORD \* pdwStatus)**

*Parameters:*

hif	- open interface handle on the device
pcbTx	- number of characters in the transmit buffer
pcbRx	- number of characters in the receive buffer
pdwStatus	- bit flags to indicate channel status

This function is used to get information about the state of the communications port associated with the specified interface handle.

The value returned in pdwStatus is the bitwise-or of the following status bits:

mskAciStsTxHalt	transmit buffer is halted
mskAciStsRxBlock	receive buffer is blocked
mskAciStsTxStall	transmit buffer is stalled due to flow control
mskAciStsRxStall	receive buffer is stalled due to flow control
mskAciStsTxFcEnable	transmit flow control is enabled
mskAciStsRxFcEnable	receive flow control is enabled

### **DaciHaltTx(HIF hif, BOOL fHalt)**

*Parameters:*

hif	- open interface handle on the device
fHalt	- halt (TRUE)/resume (FALSE) the transmit buffer

This function is used to halt/resume the transmit buffer of the communications port associated with the specified interface handle. When the transmit buffer is halted, characters in the transmit buffer will not be transmitted. Characters can still be written to the buffer. When transmission is resumed, any

characters in the transmit buffer will be sent. The transmit buffer is halted by passing TRUE and resumed by passing FALSE.

### **DaciSetRxBlock(HIF hif, BOOL fBlock)**

*Parameters:*

- |        |   |
|--------|---|
| hif    | - open interface handle on the device                   |
| fBlock | - block (TRUE)/un-block (FALSE) on receive buffer empty |

This function is used to set the behavior when reading from the receive buffer and the buffer becomes empty before the requested number of characters have been read. If fBlock is TRUE, the device will wait for the requested number of characters to be received. If fBlock is FALSE, the device will return immediately with fewer characters than requested. This should not be used to block for long periods of time. To prevent the system from being locked up by a device that is failing to respond, there is a master timeout in the Adept runtime system that will abort any function call that takes longer than about 15 seconds to complete.

### **DaciPurgeBuffer(HIF hif, BOOL fTx, BOOL fRx)**

*Parameters:*

- |     |                                       |
|-----|---------------------------------------|
| hif | - open interface handle on the device |
| fTx | - purge transmit buffer               |
| fRx | - purge receive buffer                |

This function will remove all characters from the transmit and/or receive buffers on the port associated with the specified interface handle.

### **DaciPutChar(HIF hif, BYTE bSnd, BOOL fOverlap)**

*Parameters:*

- |          |  |
|----------|--|
| hif      | - open interface handle on the device    |
| bSnd     | - character to transmit                  |
| fOverlap | - TRUE if operation should be overlapped |

This function writes the specified character to the transmit fifo buffer. If the transmit fifo buffer is currently full, it will wait until there is room in the buffer for the character. If the transmit fifo is full, and the transmission is stalled, then an error is returned.

### **DaciGetChar(HIF hif, BYTE \* pbRcv, BOOL fOverlap)**

*Parameters:*

- |          |  |
|----------|--|
| hif      | - open interface handle on the device    |
| pbRcv    | - variable to receive the character      |
| fOverlap | - TRUE if operation should be overlapped |

This function returns the next character from the receive fifo buffer. If there is no character waiting in the receive fifo, the function returns an error.

**DaciPutBuf(HIF hif, BYTE \* rgchSnd, ULONG cchReq, BOOL fOverlap)***Parameters:*

hif	- open interface handle on the device
rgchSnd	- buffer containing characters to send
cchSnd	- number of characters to send
fOverlap	- TRUE if operation should be overlapped

This function writes the specified number of characters to the transmit fifo buffer. This function will not return until all characters have been placed into the buffer, which may require waiting for characters to be sent. If transmission is blocked and there isn't room in the fifo to contain the characters being written an error is returned.

**DaciGetBuf(HIF hif, BYTE \* rgchRcv, ULONG cchReq, ULONG \* pcchRcv, BOOL fOverlap)***Parameters:*

hif	- open interface handle on the device
rgchRcv	- buffer to receive characters read
cchReq	- maximum number of characters to read
pcchRcv	- number of characters read
fOverlap	- TRUE if operation should be overlapped

This function returns characters from the receive fifo buffer. It will return the number of characters requested or the number of characters currently in the buffer, whichever is less. If there are fewer characters in the buffer than requested, it returns an error. The value returned in *pcchRcv* will be the number of characters actually returned. Note: If the call to this function is made as an overlapped I/O request (fOverlap specified as TRUE), then the value of *pcchRcv* won't be set until the overlapped I/O completes.