

# Digilent Adept Asynchronous Parallel Port Interface (DEPP) Programmer's Reference Manual

Revision: August 16, 2010



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

## Introduction

This document describes the programming interface to the Digilent Adept Asynchronous Parallel Port Interface (DEPP) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DEPP subsystem and the API functions used to access its features.

The DEPP subsystem provides the ability to transfer data between PC applications software and control registers or memory in certain Digilent Adept compatible devices. These data transfer API functions allow for writing or reading a single register, writing or reading sets of registers, or writing or reading a stream of data into or out of a single register.

Some Digilent Adept compatible devices, such as programmable logic boards, require that a DEPP compatible logic configuration or device firmware be running in the target device for the DEPP interface to work. This device logic must conform to the interface described in the document: Application Note 10, Digilent Asynchronous Parallel Interface. This document is included in the Adept SDK documentation and is also available on the Digilent web site. ([www.digilentinc.com](http://www.digilentinc.com))

All DEPP API calls return a Boolean value: TRUE if the call is successful, FALSE if not successful.

## DEPP API Functions

The following API functions make up the DEPP interface.

### DeppGetVersion(char \* szVersion)

#### Parameters

szVersion - pointer to buffer to receive version string

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

### DeppGetPortCount(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 DEPP ports supported by the device specified by hif.

### DeppGetPortProperties(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 DEPP port. The port properties bits indicate the specific features of the DEPP specification implemented by the specified port.

### DeppEnable(HIF hif)

#### Parameters

hif - open interface handle on the device

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

### DeppEnableEx(HIF hif, INT32 prtReq)

#### Parameters

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

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

**DeppDisable(HIF hif)***Parameters*

hif - open interface handle on the device

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

**DeppSetTimeout(HIF hif, DWORD tnsTimeoutTry, DWORD\* ptnsTimeout)***Parameters*

hif - open interface handle on the device  
tnsTimeoutTry - desired value (in nanoseconds) to be used to determine a timeout  
ptnsTimeout - pointer to return actual actual value set

This function is used to set the amount of delay to allow before an EPP transaction will timeout. This is to prevent a malfunctioning device from blocking access to the port. The timeout value will be set to the nearest supported value that doesn't exceed the value requested in *tnsTimeoutTry*, or the largest supported value. The actual value set is returned in *ptnsTimeout*.

**DeppPutReg (HIF hif, BYTE bAddr, BYTE bData, BOOL fOverlap)***Parameters*

hif - open interface handle on the device  
bAddr - address of register to send data byte  
bData - data byte to send to address  
fOverlap - TRUE if operation should be overlapped

This function writes a single data byte to the register with the specified address.

**DeppGetReg (HIF hif, BYTE bAddr, BYTE\* pbData, BOOL fOverlap)***Parameters*

hif - open interface handle on the device  
bAddr - address of register to read data byte  
pbData - pointer to store data byte read  
fOverlap - TRUE if operation should be overlapped

This function reads a single data byte from the register with the specified address.

**DeppPutRegSet (HIF hif, BYTE\* pbAddrData, DWORD nAddrDataPairs, BOOL fOverlap)***Parameters*

hif	- open interface handle on the device
pbAddrData	- buffer of register address and data pairs
nAddrDataPairs	- number of register address/data pairs in pbAddrData buffer
fOverlap	- TRUE if operation should be overlapped

This function sends multiple data bytes to multiple register addresses. A buffer containing addresses/data pairs is provided in the pbAddrData parameter. An address/data pair consists of 1 register address byte followed by 1 data byte in the pbAddrData buffer. For example, the address in pbAddrData [0] is paired with the data byte in pbAddrData [1]. Each data byte will be written to the register at the associated address during the transaction. The number of address/data pairs in the pbAddrData buffer is specified in the nAddrDataPairs parameter. It is permissible to repeat register addresses in the pbAddrData buffer to cause multiple bytes to be sent to the same register address.

**DeppGetRegSet (HIF hif, BYTE\* pbAddr, BYTE\* pbData, DWORD cbData, BOOL fOverlap)***Parameters*

hif	- open interface handle on the device
pbAddr	- buffer of register addresses
pbData	- pointer to store data bytes read back from specified register addresses
cbData	- number of data bytes to read back
fOverlap	- TRUE if operation should be overlapped

This function gets multiple data bytes from multiple register addresses. A buffer containing specified register addresses is provided along with a buffer to receive bytes read back from the addresses. Each element in the pbData buffer is read from the corresponding address in the pbAddr buffer. For example, the data byte in the register specified by the address in pbAddr [0] is written to pbData [0]. The number of bytes to be read out is specified in the cbData parameter. It is permissible to repeat register addresses in the pbAddr buffer to cause multiple values to be read from the same register address.

**DeppPutRegRepeat (HIF hif, BYTE bAddr, BYTE\* pbData, DWORD cbData, BOOL fOverlap)***Parameters*

hif	- open interface handle on the device
bAddr	- register address to stream data bytes to
pbData	- buffer of data bytes to stream to specified address
cbData	- number of data bytes to stream to specified address
fOverlap	- TRUE if operation should be overlapped

This function sends a stream of data bytes to a single, specified register address. Data bytes from the buffer in pbData are sent to the address specified by bAddr. The data is sent as quickly as the hardware will allow. The number of bytes to be sent to bAddr is specified in cbData.

**DeppGetRegRepeat (HIF hif, BYTE bAddr, BYTE\* pbData, DWORD cbData, BOOL fOverlap)***Parameters*

hif	- open interface handle on the device
bAddr	- register address to stream data bytes from
pbData	- pointer to store data bytes streamed from specified address
cbData	- number of data bytes to stream from specified address
fOverlap	- TRUE if operation should be overlapped

This function gets a stream of data bytes from a single, specified register address. Data bytes are read out of the address specified by bAddr into the buffer in pbData. The data is read as quickly as the hardware will allow. The number of bytes to be read out of bAddr is specified in cbData.