

# Digilent Adept Device Access Manager (DMGR) Programmer's Reference Manual

Revision: September 2, 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 Device Access Manager (DMGR) subsystem for version 2 of the Digilent Adept software system. It describes the features supported by the DMGR subsystem and the API functions used to access these features.

The DMGR subsystem is used to open and close handles for access to Digilent Adept compatible devices. It is also used to enumerate Digilent Adept compatible devices connected to or accessible from the user's PC and to query and set attributes within the devices.

## General API Functions

The following API functions make up the basic DMGR interface.

### DmgrGetVersion(char \* szVersion)

*Parameters:*

szVersion - pointer to buffer to receive version string

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

### DmgrGetLastError()

*Parameters:*

none

This function returns the error code of the last error to occur in the context of the calling process and thread. If no error has occurred since the last time this function was called, the value `ercNoError` is returned. After the error code has been returned, the error status for the calling thread is reset to `ercNoError`.

### DmgrSzFromErc(ERC erc, char \* szErc, char \* szErcMessage)

*Parameters:*

erc - error code  
szErc - buffer to receive symbolic name for the error code  
szErcMessage - buffer to receive descriptive string for the error code

This function returns strings that can be used for generating error messages. It will return a string giving a symbolic name for the error code in `szErc` and another string describing the meaning of the error code in `szErcMessage`. The symbols `cchErcMax` and `cchErcMsgMax`, defined in `dpcdecl.h` give the maximum lengths of the strings that will be returned.

**DmgrSetInfo(DVC \* pdvc, DINFO dinfo, void \* pvInfoSet)***Parameters:*

pdvc	- pointer to DVC identifying the device
dinfo	- DINFO value specifying the device attribute to set
pvInfoSet	- buffer containing the attribute value to set

This function is used to set the values of user settable attributes in Digilent Adept compatible devices. The *pdvc* parameter points to an initialized DVC structure used to specify the device. See the Device Enumeration functions or the function *DmgrGetDvcFromHif* for information on initializing the DVC. The *dinfo* parameter specifies the kind of information to be set. See the section on Device Attribute Information in the Digilent Adept System Programmer's Reference Manual for information on the DINFO values. The *pvInfoSet* parameter points to a buffer containing the value to be set in the device. The type of data to be placed in the buffer depends on the specific DINFO value being set.

**DmgrGetInfo(DVC \* pdvc, DINFO dinfo, void \* pvInfoGet)***Parameters:*

pdvc	- pointer to DVC identifying the device
dinfo	- DINFO value specifying the device attribute to get
pvInfoGet	- pointer to the buffer to receive the attribute value

This function is used to query the values of device attributes in Digilent Adept compatible devices. The *pdvc* parameter points to an initialized DVC structure used to specify the device. See the Device Enumeration functions or the function *DmgrGetDvcFromHif* for information on initializing the DVC. The *dinfo* parameter specifies the kind of information to be returned. See the section on Device Attribute Information in the Digilent Adept System Programmer's Reference Manual for information on the DINFO values. The *pvInfoGet* parameter points to the buffer to receive the value returned by the device. The type of data to be returned in the buffer depends on the specific DINFO value being queried.

**DmgrGetDvcFromHif(HIF hif, DVC \* pdvc)***Parameters:*

hif	- interface handle to the device
pdvc	- buffer to receive the returned DVC

This function will initialize the DVC specified by *pdvc* to correctly access the device currently being accessed via the interface handle *hif*.

## Device Open and Close Functions

The following functions are used to open and close device interface handles. An interface handle is required for most access to devices.

### **DmgrOpen(HIF \* phif, char \* szSel)**

*Parameters:*

phif	- pointer to variable to receive interface handle
szSel	- connection string to use to open the device

This function will attempt to open the device specified by the given connection string. If the connection string is one that requires an enumeration to be performed to discover the requested device, only device transport types that allow 'quick discovery' will be enumerated. See the description of Enumeration in the Digilent Adept System Programmer's Reference Manual. If the function call succeeds, an interface handle is returned in the variable specified by *phif*.

### **DmgrOpenEx(HIF \* phif, char \* szSel, DTP dtpTable, DTP dtpDisc)**

*Parameters:*

phif	- pointer to variable to receive the interface handle
szSel	- connection string to use to open the device
dtpTable	- transport type filter for device table filtering
dtpDisc	- transport type filter for device discovery filtering

This function will attempt to open the device specified by the given connection string. For connection strings that require enumeration to be performed to determine the matching device, the parameters *dtpTable* and *dtpDisc* can be used to filter the devices enumerated by transport type.

The *dtpTable* parameter is used to filter the transport types examined from the device table to search for a matching device. Examination of the device table can be suppressed by passing 0 for this parameter.

The *dtpDisc* parameter is used to filter the transport types enumerated using device discovery. This is the only way to perform device discovery on device transport types that are 'slow discovery' types (e.g. Ethernet). See the description of Enumeration in the Digilent Adept System Programmer's Reference Manual.

If the function call succeeds, an interface handle is returned in the variable specified by *phif*.

### **DmgrClose(HIF hif)**

*Parameters:*

hif	- interface handle to be closed
-----	---------------------------------

This function is used to close an interface handle when access to the device is no longer required. Interface handles are a scarce resource in the system and should be closed as soon as they are no longer needed. Any enabled port should be disabled before the handle is closed.

Once this function has returned, the specified interface handle can no longer be used to access the device.

## Device Enumeration Functions

The following functions are used to identify devices accessible to the user's computer. For an overview of device enumeration, refer to the relevant section of the Digilent Adept System Programmer's Reference Manual.

### **DmgrEnumDevices(int \* pcdvc)**

*Parameters:*

pcdvc                      - variable to receive count of enumerated devices

Perform an enumeration of devices in the system. This function will generate an enumeration list of devices from the device table and perform device discovery to find all 'quick discovery' devices connected to the user's computer and performs no device attribute filtering. The number of entries in the resulting enumeration list is returned in *pcdvc*.

### **DmgrEnumDevicesEx(int \* pcdvc, DTP dtpTable, DTP dtpDisc, DINFO dinfoSel, void \* pvInfoSel)**

*Parameters:*

pcdvc                      - variable to receive count of enumerated devices  
dtpTable                  - filter value for device table filtering  
dtpDisc                   - filter value for device discovery filtering  
dinfoSel                  - specifies type of device attribute filtering  
pvInfoSel                 - filter value for device attribute filtering

Perform an enumeration of devices in the system. This function will generate an enumeration list of devices that meet the constraints specified by the filtering parameters. The *dtpTable* parameter is used to identify the set of device transport types used to filter entries in the device table. The *dtpDisc* parameter is used to identify the set of device transport types to use for performing device discovery. The *dinfoSel* and *pvInfoSel* parameters are used to specify the type of device attribute filtering to perform on devices to be entered into the resulting enumeration list. The number of entries in the resulting enumeration list is returned in *pcdvc*.

### **DmgrStartEnum(DTP dtpTable, DTP dtpDisc, DINFO dinfoSel, void \* pvInfoSel)**

*Parameters:*

dtpTable                  - filter value for device table filtering  
dtpDisc                   - filter value for device discovery filtering  
dinfoSel                  - specifies type of device attribute filtering  
pvInfoSel                 - filter value for device attribute filtering

Begin a non-blocking enumeration of devices in the system. This function begins the process of generating an enumeration list of devices that meet the constraints specified by the filtering parameters. This function returns immediately, with the enumeration being performed by a private thread internal to the Adept system.

The *dtpTable* parameter is used to identify the set of device transport types used to filter entries in the device table. The *dtpDisc* parameter is used to identify the set of device transport types to use for

performing device discovery. The *dinfoSel* and *pvinfosel* parameters are used to specify the type of device attribute filtering to perform on devices to be entered into the resulting enumeration list.

### **DmgrIsEnumFinished()**

*Parameters:*

none

The function is used to query if a previously begun non-blocking enumeration has completed. It will return TRUE if the enumeration has completed, or FALSE if not.

### **DmgrStopEnum()**

*Parameters:*

none

This function is used to terminate a non-blocking enumeration that has not completed.

### **DmgrGetEnumCount(int \* pcdvc)**

*Parameters:*

pcdvc                      - variable to receive count of enumerated devices

This function is used to determine the number of elements in the current enumeration list. It can be called while a non-blocking enumeration is in progress, or after it has completed. If it is called while the enumeration is in progress, the number of elements added to the list so far is returned. The number of entries in the enumeration list is returned in *pcdvc*.

### **DmgrGetDvc(int idvc, DVC \* pdvc)**

*Parameters:*

idvc                      - index to the DVC to return  
pdvc                      - buffer to receive the returned DVC

This function will return information about a device in the current device enumeration list. The index of the device is specified by *idvc*. The index of the first element in the enumeration list is 0. Information about the requested device will be placed in the DVC structure specified by *pdvc*. The DVC structure is declared in the header file *dpcdecl.h*.

### **DmgrFreeDvcEnum()**

*Parameters:*

none

This function will release the current enumeration list. This function must be called after the result of an enumeration is no longer needed and before another enumeration can be performed.

## Transfer Control and Status Functions

### **DmgrGetTransResult(HIF hif, DWORD \* pdwDataOut, DWORD \* pdwDataIn, DWORD tmsWait)**

*Parameters:*

hif	- interface handle
pdwDataOut	- variable to receive count of bytes sent
pdwDataIn	- variable to receive count of bytes received
tmsWait	- wait time

This function is used to query the status/result of the last data transfer transaction on the interface handle specified by *hif*. If the last transaction on *hif* specified an overlapped I/O operation, this function can be used to determine if the transaction has completed, or if it is still in progress.

This function will return TRUE if the last transaction completed without error. It will return FALSE if the last transaction completed with an error, or if the last transaction has not yet completed. The error code associated with the last transaction can be obtained by calling *DmgrGetLastError*. If the last transaction was an overlapped i/o operation that has not yet completed, the error code will be *ercTransferPending*.

If the last transaction on the specified handle was a non-overlapped I/O operation or an overlapped operation that has completed, *DmgrGetTransResult* will return immediately. If the last transaction was an overlapped I/O operation that hasn't yet completed, the *tmsWait* parameter is used to control how long the function waits before it returns. The value '0', will cause *DmgrGetTransResult* to return immediately. The value *tmsWaitInfinite* (defined in *dmgr.h*) will cause it to wait until the I/O operation has completed. Otherwise, the function will wait the specified number of milliseconds or until the I/O operation to completed before returning.

On return, the variables specified by *pdwDataOut* and *pdwDataIn* will be filled in with the number of bytes actually transferred in each direction (or 0, if no data was transferred in that direction).

### **DmgrCancelTrans(HIF hif)**

*Parameters:*

hif	- interface handle
-----	--------------------

This function can be used to cancel a pending overlapped I/O transfer operation in progress on the specified handle. This function schedules the transfer to be canceled, but the operation may not have been canceled yet when *DmgrCancelTrans* returns. It is necessary to continue to call *DmgrGetTransResult* and *DmgrGetLastError* waiting for the error code *ercTransferCanceled* to be returned.

**DmgrSetTransTimeout(HIF hif, DWORD tmsTimeout)***Parameters:*

hif	- interface handle
tmsTimeout	- transfer time limit value to set

This function is used to set the transfer timeout period used to automatically abort transactions that are taking too long to complete. The transfer timeout value is specified in milliseconds.

The Adept system automatically aborts any data transfer operation that takes longer than the current timeout period to complete. The purpose of this timeout is to prevent the system from being locked up waiting on a malfunctioning device. The default timeout period is 10 seconds (10,000 milliseconds).

In some circumstances, the default value may be too short or too long. For example, 10 seconds may be too short of a period for an I/O operation occurring over an Ethernet connection being routed over the internet. In this case, it may be necessary to set the timeout to a longer period than the default.

**DmgrGetTransTimeout(HIF hif, DWORD \* ptmsTimeout)***Parameters:*

hif	- interface handle
ptmsTimeout	- variable to receive current transfer timeout limit

This function queries the current setting of the transfer timeout period.

## Device Table Management Functions

The following functions are used to modify the device table. The enumeration facility can be used to get a list of the current entries in the device table. To do this, use the function `DmgrEnumDevicesEx` with the `otpDisc` parameter set to `otpNone` and the .

### **DmgrOpenDvcMg(HWND hwnd)**

*Parameters:*

hwnd                      - window handle for dialog box parent window

This function will display a modal dialog box that provides a user interface for managing the device table. This function is only supported in the Microsoft Windows version of the Adept system. This function will not return until the user has dismissed the modal dialog box.

### **DmgrDvcTblAdd(DVC \* pdvc)**

*Parameters:*

pdvc                      - pointer to DVC containing table entry to add

This function will add an entry into the device table. The DVC structure specified by *pdvc* must be initialized with the desired alias string in the `szName` member, the corresponding connection string in the `szConn` member, and the corresponding device type in the `otp` member. For USB devices the connection string is a serial number. See the Device Connection String section of the Digilent Adept System Programmer's Reference Manual for more information.

### **DmgrDvcTblRem(char \* szAlias)**

*Parameters:*

szAlias                   - alias string of table entry to remove

This function will remove the specified entry from the device table. The device table entry matching the specified alias string will be removed.

### **DmgrDvcTblSave()**

*Parameters:*

none

This function will save the resulting device table after modification have been made using the `DmgrDvcTblAdd` or `DmgrDvcTblRem` functions.



## Device Transport Type Related Functions

The following functions can be used to determine the set of transport types supported by the current implementation of the system. These functions can be used dynamically build user interfaces that allow the user to filter devices based on the device transport type.

### **DmgrGetDtpCount(none)**

*Parameters:*

none

This function returns the number of device transport types defined by the current implementation of the Adept system.

### **DmgrGetDtpFromIndex(int idtp, DTP \* pdtp)**

*Parameters:*

idtp	- index of the transport type to return
pdtp	- variable to receive the DTP value for the requested transport type

This function will return the DTP value corresponding to the the index value specified in *idtp*.

### **DmgrGetDtpString(DTP dtp, char \* szDtpString)**

*Parameters:*

dtp	- DTP value for the requested transport type
szDtpString	- variable to receive the display string for the requested transport type

This function will return a string that describes the transport type specified by *dtp*.