Digilent Adept Pin Input/Output Interface (DPIO) 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 of the Digilent Adept Pin Input/Output (DPIO) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DPIO subsystem and the API functions used to access its features.

The DPIO system provides a low speed, bi-directional, pin oriented, input/output mechanism as part of the overall Adept 2 system.

The DPIO interface provides for up to 32 individual I/O pins on each port. Pins can be input only, output only, or bi-directional. Bi-directional pins can be individually configured to be either inputs or outputs. Pins that are configured as outputs can be individually set to the logic 1 or the logic 0 state. The state of all pins can be read at any time. For pins set as outputs, the current state is the output level previously set. For pins that are configured as inputs the state is the level present at the pin.

An application can determine the number of I/O pins supported by a given DPIO port and whether they are input pins, output pins, or bi-directional pins. If a port supports fewer than 32 I/O pins, the supported pins are in the low order bits of the DWORD value. A port that supports four I/O pins, for example will use bits 0-3.

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

Streaming Pin I/O is an optional part of the specification. DPIO ports that support this allow writing/reading a sequence of values to/from the pins. The rate at with the values will be updated or sampled can be specified.

Port Properties

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

The following port property bits are defined for DPIO ports. These values are defined in the header file *dpio.h*

dprpPioDelay	– indicates that the port supports delay on DPIO stream functions
dprpPioStream	– indicates that the port supports DPIO stream functions

DPIO API Functions

The following API functions make up the DPIO interface.

DpioGetVersion (char * szVersion)

Parameters

szVersion	- pointer to buffer to receive version string

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

DpioGetPortCount (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 DPIO ports supported by the device specified by *hif*.

DpioGetPortProperties (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 DPIO port. The port properties bits indicate the specific features of the DPIO specification implemented by the specified port.

DpioEnable (HIF hif)

Parameters

hif	- open interface handle on the device

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

DpioEnableEx (HIF hif, INT32 prtReq)

Parameters

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

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

DpioDisable (HIF hif)

Parameters

hif	-open interface handle on the device

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

DpioGetPinMask (HIF hif, DWORD * pfsMskOut, DWORD * pfsMskIn)

Parameters

hif	- open interface handle on the device
pfsMskOut	- variable to receive output mask
pfsMskIn	- variable to receive input mask

This function is used to determine which I/O pins are available on the port currently enabled on the specified interface handle. It returns two 32 bit masks indicating which bits in the port are supported for input, output, or both. The mask returned in *pfsMskOut* will have the bit set for each bit position that can be used for output. The mask returned in *pfsMskIn* will have the bit set for each bit position that can be used for input. Bi-directional pins will have the corresponding bit set in each mask.

DpioGetPinDir (HIF hif, DWORD * pfsDir)

Parameters

hif	- open interface handle on the device
pfsDir	- variable specifying pin directions

This function will return the current setting of the pin directions for the I/O pins on the DPIO port currently enabled on the specified interface handle. The value returned in *pfsDir* will have the bit set to 1 for each pin currently configured as an output. The bit will be 0 for pins that are either configured as inputs or not supported by the port.

DpioSetPinDir (HIF hif, DWORD fsDirReq DWORD * pfsDirSet)

Parameters

hif	- open interface handle on the device
fsDirReq	- requested pin direction mak
pfsDirSet	- variable to receive pin directions actually set

This function is used to set the pin direction (input/output) for the dPIO port currently enabled on the specified interface handle. The bit mask in fsDirReq has the bit set to 1 for each pin to be configured as an output, and 0 for pins to be configured as inputs. Bits corresponding to unsupported pins should be set to 0. The value returned in pfsDirSet indicates the resulting state of the pin directions.

DpioGetPinState (HIF hif, DWORD * pfsState)

Parameters

hif	- open interface handle on the device
pfsState	- variable to receive current input pin state

This function returns the current state of the pins associated with DPIO port currently enabled on the specified interface handle. The current pin state is returned in *pfsState*. Pins that are configured as inputs will have the corresponding bit set to the state of the input condition at the pin. Pins that are configured as outputs will have the corresponding bit set to the value last written to the pin.. Any bits corresponding to unsupported pins will be set to 0.

DpioSetPinState (HIF hif, DWORD fsState)

Parameters

hif	- open interface handle on the device
fsState	- variable specifying output pin states to set

This function is sets the state of pins configured as outputs on the DPIO port currently enabled on the specified interface handle. Each pin configured as an output will be set to 1 or 0 depending on the value of the corresponding bit in *fsState*. Bits corresponding to input pins or unsupported pins will be ignored and should be set to 0.

DPIO Stream API Functions

DpioGetStreamTiming (HIF hif, DWORD * ptnsGetToSet, DWORD * ptnsSetToGet)

Parameters

hif	- open interface handle on the device
ptnsGetToSet	- variable to receive current 'GetToSet' delay time
ptnsSetToGet	- variable to receive current 'SetToGet' delay time

This function returns the current delay values associated with the DPIO port currently enabled on the specified interface handle. These values determine the timing associated with streaming i/o on the port. The value returned in *ptnsGetToSet* is the delay time in nano-seconds from when the pin state is sampled until the output pins are updated with the next output sample. The value returned in *ptnsSetToGet* is the delay time in nano-seconds from when the last output sample was place on the output pins until the next time the input pins will be sampled.

Note: This API call is only available to ports with the dprpPioStream and dprpPioDelay properties.

DpioSetStreamTiming (HIF hif, DWORD tnsGetToSet, DWORD tnsSetToGet, DWORD * ptnsGetToSet, DWORD * ptnsSetToGet)

Parameters

hif	- open interface handle on the device
tnsGetToSet	- requested 'GetToSet' delay time
tnsSetToGet	- requested 'SetToGet' delay time
ptnsGetToSet	- actual 'GetToSet' delay time obtained
ptnsSetToGet	- actual 'SetToGet' delay time obtained

This function sets the read/write delays used by the PIO port currently enabled on the specified interface handle. These values determine the timing used with streaming i/o on the port. The value in *tnsGetToSet* is the requested delay time in nano-seconds between reading a sample from the input pins and when the output pins will be updated with the next sample value. The value returned in *ptnsGetToSet* will be the actual delay value obtained. The value in *tnsSetToGet* is the requested delay time in nano-seconds between writing a sample to the output pins and when the next time the input pins will be sampled. The value returned in *ptnsSetToGet* is the actual delay value obtained.

Note: This API call is only available to ports with the dprpPioStream and dprpPioDelay properties.

DpioStreamState (HIF hif, BYTE * rgfsSet, BYTE * rgfsGet, DWORD cfs, BOOL * fHang, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
rgfsSet	- buffer of values to set on output pins
rgfsGet	- buffer to receive values read from input pins
cfs	- number of values to transfer
fHang	- TRUE when pin IO transfer was not performed at the specified rate.
fOverlap	- TRUE if operation should be overlapped

This function is used to perform streaming pin i/o on the DPIO port current enabled on the specified interface handle. The array specified by *rgfsSet* contains values to be written to the output pins. This pointer can be NULL if only input is being performed. The array specified by *prgfsGet* points to a buffer to receive the values read from the input pins. This value can be NULL if only output is being done. The value in *cfs* specifies the number of values to be written/read. If the fHang parameter is returned as TRUE, a hang occurred in the data transfer (meaning it was not performed at the specified rate).

Note: This API call is only available to ports with the dprpPioStream property.