WinDriver USB
KernelDriver USB

Control Transfers and Setup Packets

White Paper

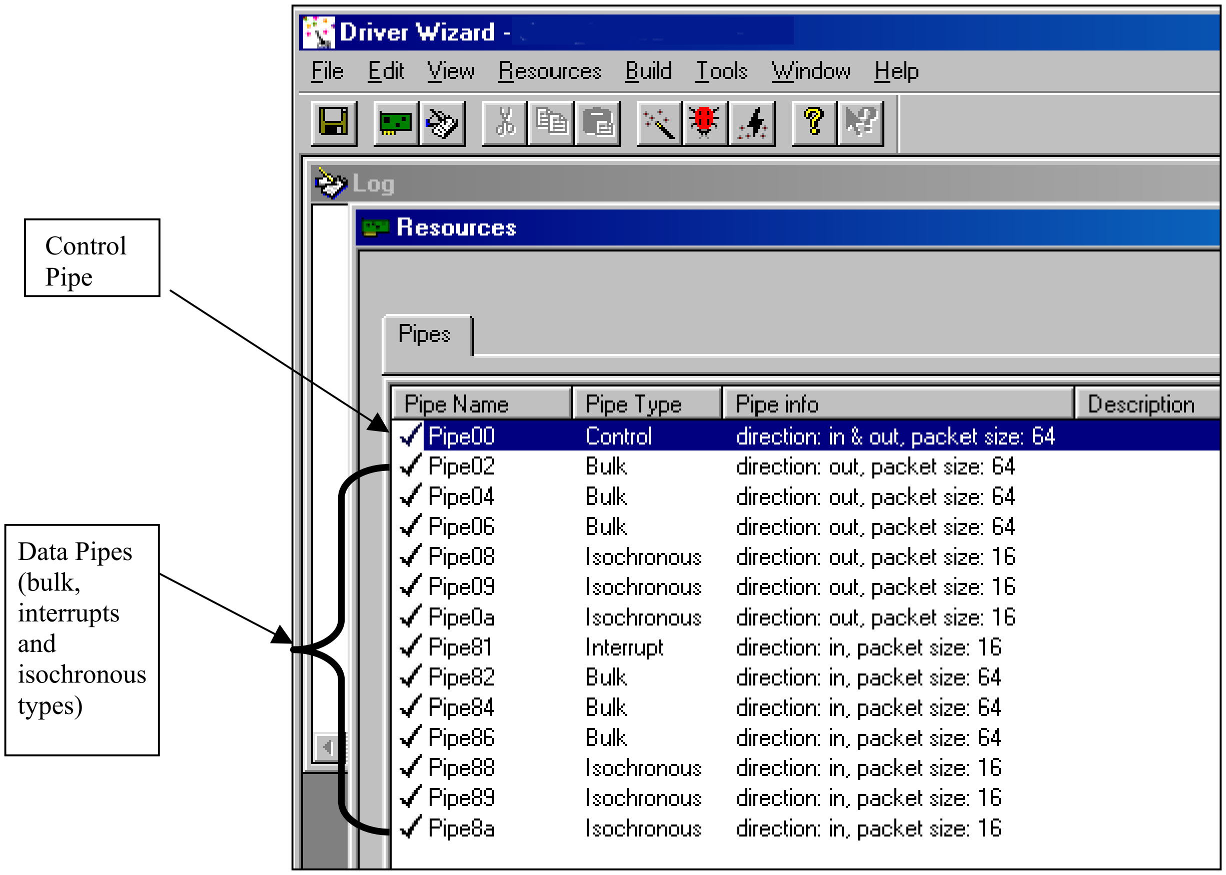
USB Data Exchange

The USB standard supports two kinds of data exchange between the host and the device: functional data exchange and control exchange.

- Functional data exchange is used to move data to and from the device. There are three types of data transfers: Bulk transfers, Interrupt transfers and Isochronous transfers.

- Control exchange is used to configure a device when it is first attached, getting common configuration data, and can be also used for other device-specific purposes, including control of other pipes on the device. The control exchange is transferred via the control pipe (Pipe 00).

The screenshot below shows a device with one bi-directional control pipe and 13 functional data transfer pipes/endpoints:


Control Pipe
Data Pipes (bulk, interrupts and isochronous types)
Driver Wizard -
File Edit View Resources Build Tools Window Help
Log
Resources
Pipes

Pipe Name	Pipe Type	Pipe info	Description
✓ Pipe00	Control	direction: in & out, packet size: 64	
✓ Pipe02	Bulk	direction: out, packet size: 64	
✓ Pipe04	Bulk	direction: out, packet size: 64	
✓ Pipe06	Bulk	direction: out, packet size: 64	
✓ Pipe08	Isochronous	direction: out, packet size: 16	
✓ Pipe09	Isochronous	direction: out, packet size: 16	
✓ Pipe0a	Isochronous	direction: out, packet size: 16	
✓ Pipe81	Interrupt	direction: in, packet size: 16	
✓ Pipe82	Bulk	direction: in, packet size: 64	
✓ Pipe84	Bulk	direction: in, packet size: 64	
✓ Pipe86	Bulk	direction: in, packet size: 64	
✓ Pipe88	Isochronous	direction: in, packet size: 16	
✓ Pipe89	Isochronous	direction: in, packet size: 16	
✓ Pipe8a	Isochronous	direction: in, packet size: 16	

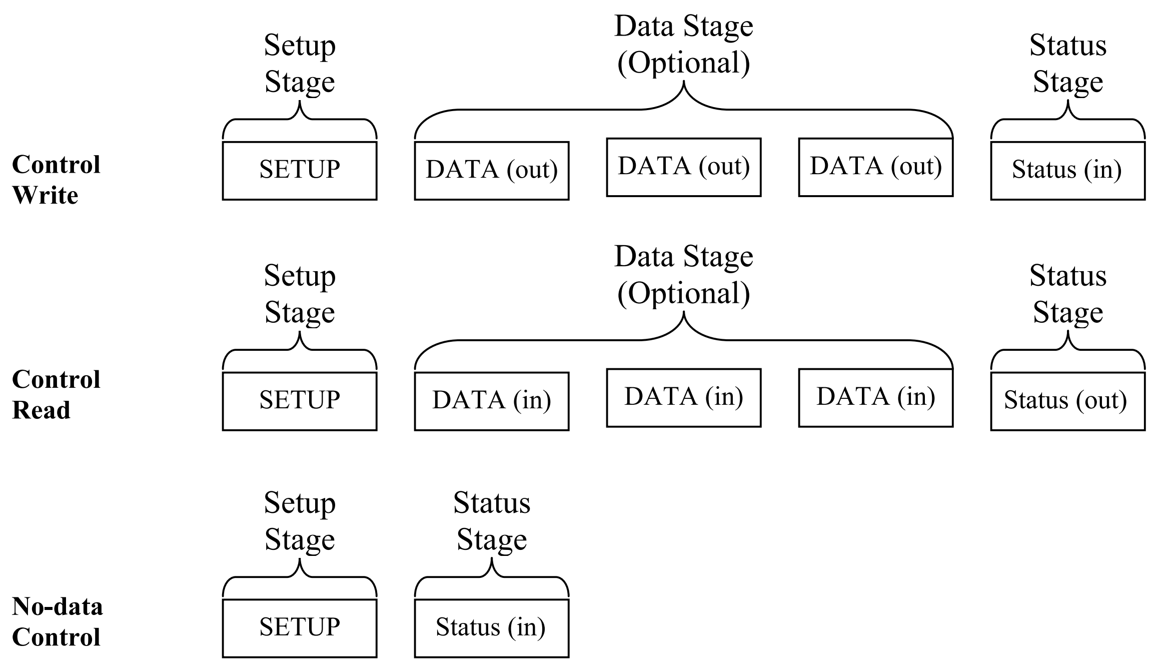
More about the Control Transfer

The control transaction always begins with a setup stage. Then it is followed by zero or more control data transactions (data stage) that carry the specific information for the requested operation, and finally, a Status transaction completes the control transfer by returning the status to the host.

During the setup stage, a SETUP Packet is used to transmit information to the control endpoint of the device. The Setup Packet consists of eight bytes, and its format is specified in the USB specification.

A control transfer can be a read transaction or a write transaction. In a read transaction, the Setup Packet indicates the characteristics and amount of data to be read from the device. In a write transaction, the Setup Packet contains the command sent (written) to the device and the number of control Data bytes, associated with this transaction, that are sent to the device in the data stage.

The following figure shows the sequence of read and write transactions (the figure is taken from the USB specification). 'In' means the data flows from the device to the host. 'Out' means the data flows from the host to the device.



<u>The Setup Packet</u>

The Setup packets (combined with the control data stage and the status stage) are used to configure and send commands to the device. Chapter 9 of the USB specification defines standard device requests. Such USB request is sent from the host to the device, using Setup Packet. The USB device is required to response properly to these requests. In addition, each vendor may define device-specific Setup Packets, to perform device specific operations. The standard Setup Packets (standard USB device requests) are detailed below. The vendor's device-specific Setup Packets are detailed in the vendor's specific data book for each USB device.

USB Setup Packet format

(For further information, please refer to the USB specification at http://www.usb.org)

Byte No.	Field	Description
0	bmRequestType	Bit 7: Request direction (0= Host to device, 1=Device to host) Bits 5..6: Request type (0=standard, 1=class, 2=vendor, 3=reserved) Bits 0..4: Recipient (0=device, 1=Interface, 2=Endpoint, 3=other)
1	bRequest	The actual request (see next table)
2	wValueL	A word-size value that varies according to the request (for example – in the CLEAR_FEATURE request, the value is used to select the feature, in the GET_DESCRIPTOR request, the value indicates the descriptor type, in the SET_ADDRESS request the value contains the device address).
3	wValueH	The upper byte of the Value word
4	wIndexL	A word size value that varies according to the request. The index is generally used to specify an endpoint or an interface.
5	wIndexH	The upper byte of the Index word.
6	wLengthL	Word size value, indicates the number of bytes to be transferred if there is a data stage.
7	wLengthH	The upper byte of the Length word.

Standard device requests codes

bRequest	**Value**
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

Setup Packet Example

WinDriver and KernelDriver allow you to easily send and receive control transfers on Pipe00, while using the Driver Wizard (both with WinDriver or KernelDriver) to test your device and within WinDriver / KernelDriver API.

This is an example of a standard USB device request, to illustrate the Setup Packet format and its different fields. The Setup packet is in Hex format.

The following Setup Packet is a 'Control Read' transaction that retrieves the 'Device descriptor' from the USB device. The 'Device descriptor' includes information such as USB standard revision, the vendor ID and the device product ID.

GET_DESCRIPTOR (device) Setup Packet

80	06	00	01	00	00	12	00

Setup Packet meaning:

Byte No.	Field	Value	Description
0	bmRequestType	80	8h = 1000b bit 7=1 $\rightarrow$ direction of data is from device to host. 0h = 0000b bits 0..1 = 00 $\rightarrow$ the recipient is the 'device'.
1	bRequest	06	The Request is 'GET_DESCRIPTOR'
2	wValueL	00	
3	wValueH	01	The descriptor type is device (the values are defined in the USB spec.).
4	wIndexL	00	(The index is not relevant in this setup packet since there is only one device descriptor)
5	wIndexH	00	
6	wLengthL	12	Length of the data to be retrieved: 18 (12h) bytes (this is the length of the 'device descriptor').
7	wLengthH	00	

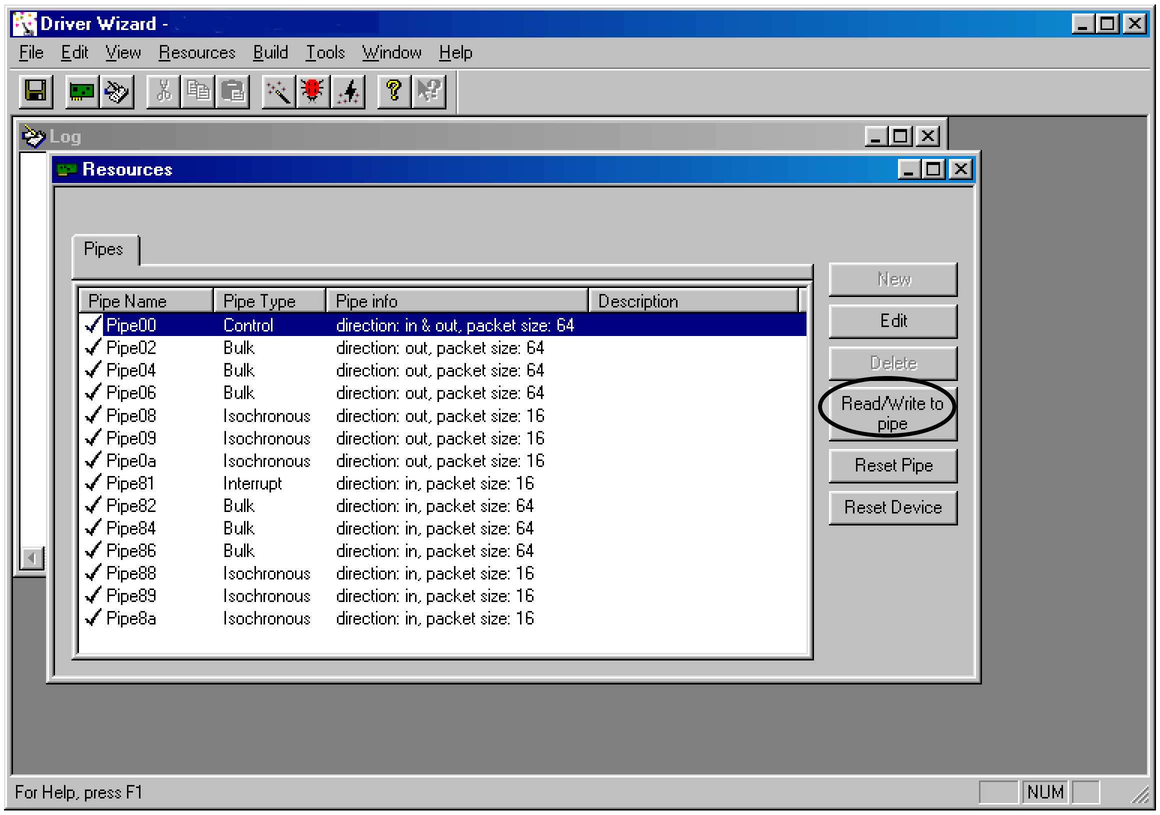
In response, the device sends the 'Device Descriptor' data. For example, this is a 'Device Descriptor' of 'Cypress EZ-USB Integrated Circuit':

Byte No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Content	12	01	00	01	ff	ff	ff	40	47	05	80	00	01	00	00	00	00	01

As defined in the USB specification, byte 0 indicates the length of the descriptor, bytes 2..3 contains the USB specification release number, bytes 7 is the maximum packet size for endpoint 00, bytes 8..9 are the Vendor ID, bytes 10..11 are the Product ID, etc.

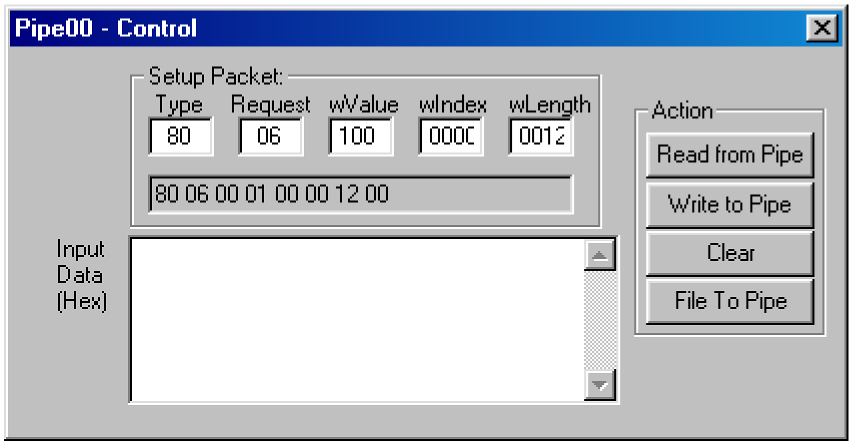
Control Transfers with the Driver Wizard

1. Choose Pipe00 and click on 'Read/Write to pipe'



Pipe Name	Pipe Type	Pipe info	Description
✓ Pipe00	Control	direction: in & out, packet size: 64	
✓ Pipe02	Bulk	direction: out, packet size: 64	
✓ Pipe04	Bulk	direction: out, packet size: 64	
✓ Pipe06	Bulk	direction: out, packet size: 64	
✓ Pipe08	Isochronous	direction: out, packet size: 16	
✓ Pipe09	Isochronous	direction: out, packet size: 16	
✓ Pipe0a	Isochronous	direction: out, packet size: 16	
✓ Pipe81	Interrupt	direction: in, packet size: 16	
✓ Pipe82	Bulk	direction: in, packet size: 64	
✓ Pipe84	Bulk	direction: in, packet size: 64	
✓ Pipe86	Bulk	direction: in, packet size: 64	
✓ Pipe88	Isochronous	direction: in, packet size: 16	
✓ Pipe89	Isochronous	direction: in, packet size: 16	
✓ Pipe8a	Isochronous	direction: in, packet size: 16	

2. Enter the required Setup Packet. For a 'Write' transaction that includes also a data stage, enter the data in the 'Input Data' field. Press 'Read from Pipe' or 'Write to Pipe' according to the required transaction.



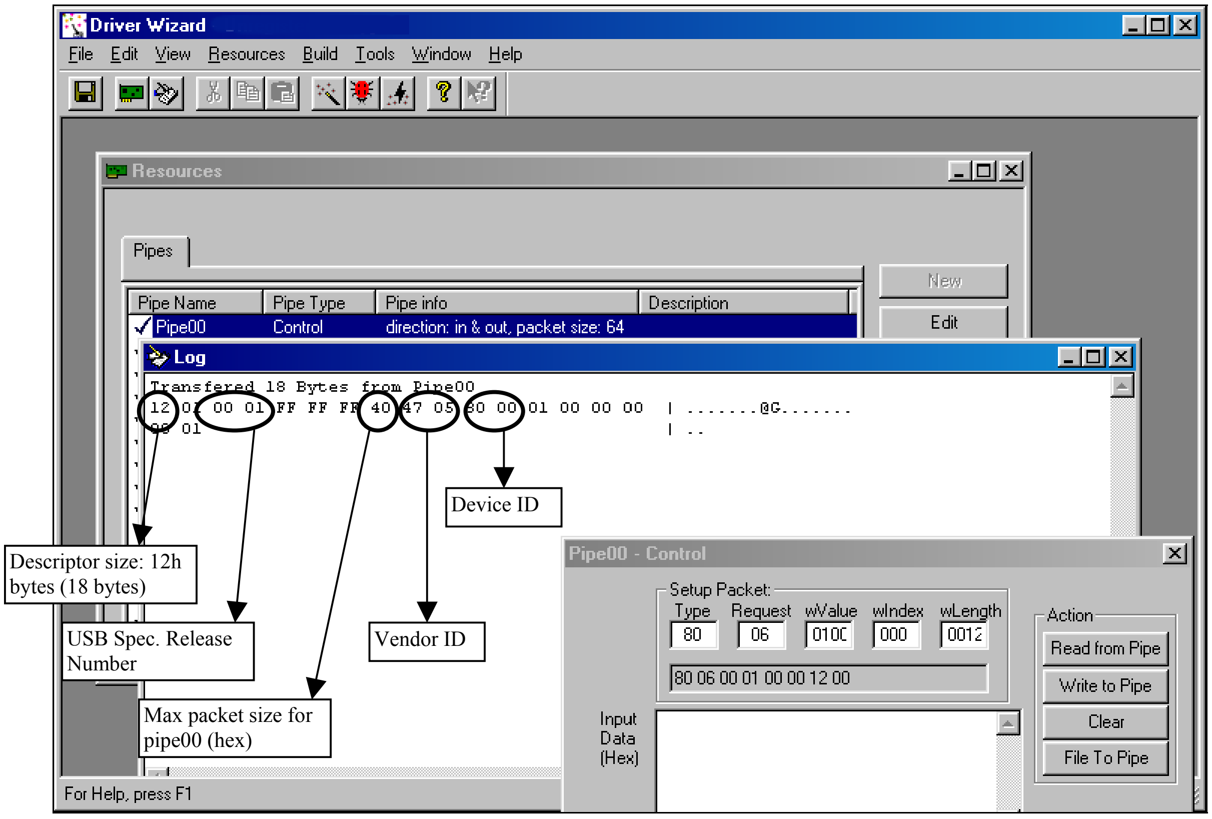
Setup Packet:

Type	Request	wValue	wIndex	wLength
80	06	100	000C	0012

80 06 00 01 00 00 12 00

Input Data (Hex)

3. The ‘Device Descriptor’ data retrieved from the device can be seen in the Wizard log screen:



Pipe Name	Pipe Type	Pipe info	Description
✓ Pipe00	Control	direction: in & out, packet size: 64	

```
Transfered 18 Bytes from Pipe00
12 01 00 01 FF FF FF 40 47 05 80 00 01 00 00 00  | ........@G.......
00 01                                            | ..
```

Type	Request	wValue	wIndex	wLength
80	06	010C	000	0012

80 06 00 01 00 00 12 00

Control Transfers with WinDriver / KernelDriver API

To perform a read or write transaction on the control pipe, you can either use the API generated by DriverWizard for your hardware, or directly call the WinDriver WDU_Transfer function from within your application.

Fill the setup packet in the BYTE SetupPacket[8] array and call these functions to send setup packets on Pipe00 and to retrieve control and status data from the device.

• The following sample demonstrates how to fill the SetupPacket[8] variable with a GET_DESCRIPTOR setup packet:

```
setupPacket[0] = 0x80; //BmRequstType
setupPacket[1] = 0x6; //bRequest [0x6 == GET_DESCRIPTOR]
setupPacket[2] = 0; //wValue
setupPacket[3] = 0x1; //wValue [Descriptor Type: 0x1 == DEVICE ]
setupPacket[4] = 0; //wIndex
setupPacket[5] = 0; //wIndex
setupPacket[6] = 0x12; //wLength [Size for the returned buffer]
setupPacket[7] = 0; //wLength
```

• The following sample demonstrates how to send a setup packet to the control pipe (a GET instruction; the device will return the information requested in the pBuffer variable):

```
WDU_TransferDefaultPipe(hDev, TRUE, 0, pBuffer, dwSize,
bytes_transferred, &setupPacket[0], 10000);
```

• The following sample demonstrates how to send a setup packet to the control pipe (a SET instruction):

```
WDU_TransferDefaultPipe(hDev, FALSE, 0, NULL, 0, bytes_transferred,
&setupPacket[0], 10000);
```

For further information regarding the WDU_Transfer function, please refer to the 'WinDriver Function Reference' Chapter in the WinDriver Manual.

Contacting Jungo

Phone: (USA) 1-877-514-0537 (WorldWide) +972-9-8870878
Fax: (USA) 1-877-514-0538 (WorldWide) +972-9-8870877
Support: support@jungo.com
Sales: sales@jungo.com
Web: http://www.jungo.com