USB INTERFACE SPECIFICATION
IOLab

Document Number 1814F03
Revision 11

Prepared for W.H. Freeman
Date: 24-Jul-2013, 11:10 AM

Indesign, LLC
8225 E 56th Street
Indianapolis, IN 46216

Phone: 317.377.5450
FAX: 317.377.5455
http://www.indesign-llc.com

Document Change Notes

Rev	Description	Date	Changed by
P1	Document created	3/15/2012	Rick Neufelder
1	Initial Release	7/3/2012	Rick Neufelder John Sawyer
2	Corrections after review. Add GET_PACKET_CONFIG, pairing state	8/8/2012	Rick Neufelder John Sawyer
3	Add SET/GET_FIXED_CONFIG, redefine SET/GET_PAIRING argument length, redefine DATA_FROM_REMOTE payload length	8/15/2012	Rick Neufelder John Sawyer
4	Update state diagram, change response length of GET_REMOTE_STATUS, update command GET_CALIBRATION, add ENTER_BOOTLOADER command	9/3/2012	Rick Neufelder John Sawyer
5	Change GET_REMOTE_STATUS response length, add FIND_REMOTE command, allow GET_PAIRING in Idle mode	9/17/2012	Rick Neufelder John Sawyer
6	Added POWER_DOWN command, added STOP_PAIRING command, re-ordered command values	9/27/2012	Rick Neufelder John Sawyer
7	Added START_ASYMETRIC_DATA Command, redefined NAK response codes to match versions 00.25+ of Dongle code, fixed Table 2 (valid modes for STOP_DATA), added OTA Firmware Upgrade commands	4/10/2013	Rick Neufelder John Sawyer
8	Added REMOTE Firmware Upgrade commands. Added SLIP_RF_TIMING command	6/19/2013	Rick Neufelder John Sawyer
9	Add Expanded PAIR command, add Frequency Control command	7/8/2013	Rick Neufelder John Sawyer
10	Corrected some copy/paste errors	7/8/2013	Rick Neufelder John Sawyer
11	Add Expanded Result to GET_PAIRING command	7/24/2013	Rick Neufelder John Sawyer

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1.0 INTRODUCTION

The IOLab system is a "low-cost, easy-to-use, all-purpose handheld device that performs a myriad of functions for both introductory and advanced physics courses."[1] The system consists of a USB dongle (referred to as "dongle") that wirelessly connects to one or two handheld devices (referred to as "remote") to transmit sensor information to a PC for processing and display of the data.

The dongle consists of a single microcontroller that will interface between the RF radio and the USB connection to the PC. The remote consists of two microcontrollers: a sensor microcontroller to interface to the sensors and a radio microcontroller to manage the radio communication.

1.1 PURPOSE

This document describes the USB command/response packet interface between the dongle and the PC application. This document also describes the dongle's USB descriptor configuration.

1.2 DEFINITIONS

- **ACK – acknowledgement**. A packet sent to acknowledge the receipt of a packet.
- **ASCII – American Standard Code for Information Interchange**. A character encoding scheme based on the English alphabet using codes to represent text.
- **Enumeration** – an exchange of data between a PC and a USB device during initial connection to a PC USB port.
- **EOP – end of packet byte**. A specific byte or character used to identify the end of a packet in serial communication.
- **GPIO – general purpose input/output**. Term given to microcontroller connections that can be used for digital input or output.
- **Host** – a USB port on a PC
- **NAK – negative acknowledgement**. A packet sent as a negative response to the receipt of a packet, or when the receiver is not ready or when the packet data was corrupted.
- **PC – personal computer.**
- **RF – radio frequency**. A rate of oscillation in the range of 3kHz to 300GHz which corresponds to the frequency of radio waves and the alternating currents which carry radio signals.

[1] Quoted from an article by Liz Ahlberg, Physical Sciences Editor on October 20, 2011 found at http://news.illinois.edu/ii/11/1020/physics_device.html

- **SOP – start of packet byte**. A specific byte or character used to identify the start of a packet in serial communications.
- **UART -** universal asynchronous receiver/transmitter

1.3 REFERENCES

1. Data Protocol, IOLab, 1814F08, Revision 8.
2. Product Requirements, IOLab System, 1814S01, Revision 6.
3. Serial Debug Protocol, IOLab, 1814F06, Revision 3.

2.0 DONGLE OPERATION

Once the dongle is connected to a PC's USB port, the PC and the dongle exchange data (USB enumeration) and the dongle enters either idle or paired mode. The dongle then waits for the IOLab PC application to send configuration information. Configuration items are used to set up the RF link and to configure remote sensors. Figure 1 shows the dongle's internal states.

The set pairing command instructs the dongle to set up an RF data link between the dongle and a remote. RF communications with the paired remote determine if the dongle and remote are connected. Once the sensors are configured the IOLab application instructs the dongle to enter data mode where the remote sensors are monitored and data is sent over the RF link to the dongle which is then sent to the PC application.

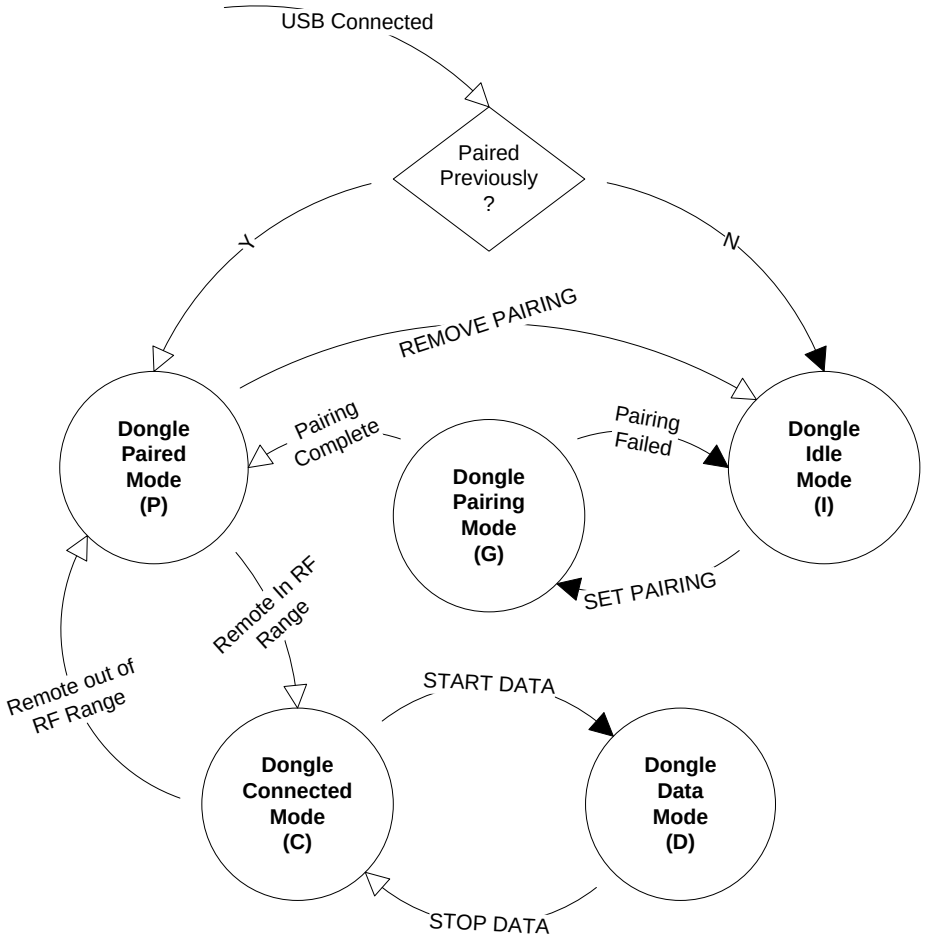


Figure 1 - Dongle State Diagram

3.0 IOLAB PACKET FORMAT

Two packet types are used with the IOLab dongle, as shown in Figure 2 and Figure 3 below. Figure 2 shows a packet that contains a data payload and Figure 3 shows a packet that does not contain a data payload. Packets that do not contain payload data have a length field with a value of zero.

This packet structure is used for all messages sent between the dongle and the PC.

Hexadecimal values (0xXX) are used for all byte values and multi-byte payload values are sent most significant byte first. The payload data field contains from one to fifty bytes.

Packet field definitions are listed below.

- **SOP** (1 byte) – Start Of Packet, delineates the beginning of a packet, equal to: **0x02**
- **CMD** (1 byte) – Command
- **Length** (1 byte) – Length of the payload data in bytes
- **Payload** (1 to 50 bytes) – Data associated with the current command
- **EOP** (1 byte) – End Of Packet, delineates the end of a packet, equal to: **0x0A**

SOP	CMD	Length	Payload	EOP
1 byte	1 byte	1 byte	1 to 50 bytes	1 byte

Figure 2 - IOLab Packet Format With Data Payload

SOP	CMD	Length	EOP
1 byte	1 byte	1 byte	1 byte

Figure 3 - IOLab Packet Format Without Payload

3.1 ACK RESPONSE

The PC initiates most communication sequences by sending a command packet to the dongle. The dongle responds positively to commands by either returning the associated command packet, if one is applicable, or by returning an ACK response.

The ACK response is described below. The PC application should not send successive command packets to the dongle until either the Response Data/ACK/NAK is received or a timeout period of 0.5 seconds has expired.

3.1.1 ACK Response Format

The ACK response shown in Figure 4 is sent from the dongle to the PC.

SOP	CMD	Length	Payload	EOP
1 byte	1 byte	1 byte	1 byte	1 byte

Figure 4 - IOLab ACK Response Packet Format

CMD: 0xAA

Payload Length: 1 byte

Response Payload:

Command
1 byte

- Command = Command being Acknowledged

3.2 NAK RESPONSE

Commands that are not recognized by the dongle, that are sent out of sequence, or are improperly formatted will result in a NAK message being returned to the PC.

3.2.1 NAK Response Format

The NAK response shown in Figure 5 is sent from the dongle to the PC.

SOP	CMD	Length	Payload	EOP
1 byte	1 byte	1 byte	2 bytes	1 byte

Figure 5 - IOLab NAK Response Packet Format

CMD: 0xBB

Payload Length: 2 bytes

Response Payload:

Command	Reason
1 byte	1 byte

- Command = Command being NAK'd
- Reason = Reason for NAK, as listed in Table 1 below

3.2.2 NAK Response Codes

Reason	NAK Response Description
0x01	Invalid command
0x02	Invalid payload data length
0x03	Invalid payload data value
0x04	Invalid state; dongle is not in the proper state to accept the command
0x05	Remote interface is busy
0x06	No response from remote
0x07	Invalid packet SOP
0x08	Invalid packet data length
0x09	Invalid packet EOP
0x0A	Invalid transmit packet data length
0x0B	Invalid transmit packet state
0x0C	Invalid remote firmware version for current command
0x0D	Error performing command

Table 1 - NAK Response Codes

4.0 COMMAND LIST

Table 2 shows a complete list of USB commands. The Command Name and Command Value fields are self explanatory. The next two fields show the direction of the command/response data flow. The Command Payload field indicates if the command requires parameters or not and shows their data lengths. The Command Response shows if the dongle responds with an ACK or a Data packet. The Response Payload indicates the response payload size, if present. The dongle Mode field shows which Dongle Modes the command is valid for (I=Idle Mode, G=Pairing Mode, P=Paired Mode, C=Connected Mode and D=Data Mode.) Note that some commands are valid in multiple modes.

Command Name	Cmd Value	USB→ Dongle	Dongle →USB	Cmd Payload	Cmd Response	Response Payload	Dongle Mode
START_PAIRING	0x10	x		5 or 10 data bytes	ACK	NA	I,P,C
STOP_PAIRING	0x11	x		NA	ACK	NA	G
GET_PAIRING	0x12	x	x	NA	Data	8 or 13 data bytes	I,P,C
FIND_REMOTE	0x13	x	x	NA	Data	3 data bytes	I,P,C
GET_DONGLE_STATUS	0x14	x	x	NA	Data	6 data bytes	I,G,P,C
START_DATA	0x20	x		NA	ACK	NA	C
STOP_DATA	0x21	x		NA	ACK	NA	D
SET_SENSOR_CONFIG	0x22	x		4 to 50 data bytes	ACK	NA	C
GET_SENSOR_CONFIG	0x23	x	x	1 data byte	Data	4 to 50 data bytes	C
SET_OUTPUT_CONFIG	0x24	x		4 to 50 data bytes	ACK	NA	C,D
GET_OUTPUT_CONFIG	0x25	x	x	1 data byte	Data	4 to 50 data bytes	C
SET_FIXED_CONFIG	0x26	x		2 data bytes	ACK	NA	C
GET_FIXED_CONFIG	0x27	x	x	1 data byte	Data	2 data bytes	C
GET_PACKET_CONFIG	0x28	x	x	1 data byte	Data	4 to 50 data bytes	C

Command Name	Cmd Value	USB→ Dongle	Dongle →USB	Cmd Payload	Cmd Response	Response Payload	Dongle Mode
GET_CALIBRATION	0x29	x	x	2 data bytes	Data	4 to 50 data bytes	C
GET_REMOTE_STATUS	0x2A	x	x	1 data byte	Data	7 data bytes	C
POWER_DOWN	0x2B	x		1 data byte	ACK	NA	C
START_ASYMETRIC_DATA	0x2C	x		1 data byte	ACK	NA	C
SLIP_RF_TIMING	0x2D	x		NA	ACK	NA	C,D
FREQUENCY_CONTROL	0x2E	x		2 data bytes	ACK	NA	C,D
RF_CONNECTION	0x40		x	NA	ASYNC	2 data bytes	P,C,D
DATA_FROM_REMOTE	0x41		x	NA	ASYNC	5 to 104 data bytes	D
ENTER_BOOTLOADER	0x50	x		NA	ACK	NA	I,G,P,C
OTA_UPGRADE_START	0x60	x	x	5 bytes	ACK	NA	C
OTA_UPGRADE_DATA	0x61	x	x	34 bytes	ACK	NA	C
OTA_UPGRADE_CRC	0x62	x	x	2 bytes	ACK	NA	C
REMOTE_UPGRADE_START	0x70	x	x	5 bytes	ACK	NA	C
REMOTE_UPGRADE_DATA	0x71	x	x	35 bytes	ACK	NA	C
REMOTE_UPGRADE_CRC	0x72	x	x	2 bytes	ACK	NA	C

Table 2 - Command List

5.0 COMMAND/RESPONSE DESCRIPTIONS

This section provides a high level description of each command and its associated payload data. Further details about each command can be found in Reference 1.

5.1 DONGLE COMMANDS

5.1.1 START_PAIRING (0x10)

The start pairing command instructs the dongle to begin the remote RF pairing process. The command data payload contains: which remote to pair with, whether a pair or un-pair is being requested, and the remote's RF unique identifier. To pair two remotes this command would be issued twice.

The standard START_PAIRING command that is 5 bytes in length is expected to be used in all normal operations. The expanded START_PAIRING command that is 10 bytes in length can be used to over-ride the default frequency allocation. This expanded START_PAIRING command should be used only in test or fact-finding configurations.

CMD: 0x10

Payload Length: 5 or 10 bytes

Command Payload:

Remote Number	**Status**	**Remote ID**	***Expanded Indicator***	***Assigned Frequencies***
1 byte	1 byte	3 bytes	*1 byte*	*4 bytes*

- Remote Number = which remote to pair with. 1=Remote1, 2=Remote2
- Status = select pair or unpair with the remote. 1=Pair, 0=Unpair
- Remote ID = remote ID to pair with, MSB-LSB
- *Expanded Indicator* (optional) = always 0x36
- *Assigned Frequencies* (optional) = 4 frequency indexes to over-ride default. Range for each is 0 to 15 (0xF)

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Modes: Idle, Paired and Connected

5.1.2 STOP_PAIRING (0x11)

The stop data command instructs the dongle to exit Pairing mode. The START_PAIRING command (with Status=1 for Pair) will continuously attempt to pair with the designated remote. This command will halt the pairing attempt.

CMD: 0x11

Payload Length: 0 bytes

Command Payload: NA

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Pairing

5.1.3 GET_PAIRING (0x12)

The get pairing command returns the current pairing information between the dongle and the RF remotes.

The standard GET_PAIRING response is 8 bytes in length and is expected to be used in all normal operations. If the expanded START_PAIRING command has previously been used to over-ride the default frequency allocation, then that information will be included and the GET_PAIRING response will be 13 bytes in length. The expanded START_PAIRING command, and resulting GET_PAIRING response, should be used only in test or fact-finding configurations.

CMD: 0x12

Payload Length: 0 bytes

Command Payload: NA

Command Response: Data

Response Payload Length: 8 or 13 bytes

Response Payload:

Remote 1 Status	**Remote 1 ID**	**Remote 2 Status**	**Remote 2 ID**	***Expanded Indicator***	***Assigned Frequencies***
1 byte	3 bytes	1 byte	3 bytes	*1 byte*	*4 bytes*

- Remote 1 Status = status of Remote 1. 1=Paired, 0=Not paired
- Remote 1 ID = remote 1 ID, MSB-LSB
- Remote 2 Status = status of Remote 2. 1=Paired, 0=Not paired
- Remote 2 ID = remote 2 ID, MSB-LSB
- *Expanded Indicator* (optional) = always 0x36
- *Assigned Frequencies* (optional) = 4 frequency indexes over-riding default. Range for each is 0 to 15 (0xF)

Valid Dongle Modes: Idle, Paired and Connected

5.1.4 FIND_REMOTE (0x13)

The find remote command instructs the dongle to search for a remote that is in the RF pairing state. Note that if more than one remote is in range and in the RF pairing state, the result of this command is undefined.

CMD: 0x13

Payload Length: 0 bytes

Command Payload: NA

Command Response: Data

Response Payload Length: 3 bytes

Response Payload:

Remote ID
3 bytes

- Remote ID = remote ID of nearby remote in the RF pairing state, MSB-LSB

Valid Dongle Modes: Idle, Paired and Connected

5.1.5 GET_DONGLE_STATUS (0x14)

The get dongle status command returns the current dongle status and other ID/version information.

CMD: 0x14

Payload Length: 0 bytes

Command Payload: NA

Command Response: Data

Response Payload Length: 6 bytes

Response Payload:

Dongle FW Version	Dongle Mode	Dongle ID
2 bytes	1 byte	3 bytes

- Dongle FW Version = current dongle firmware version, MSB-LSB
- Dongle Mode = current dongle mode. 1=Idle, 2=Pairing, 3=Paired, 4=Connected, 5=Data
- Dongle ID = dongle ID, MSB-LSB

Valid Dongle Modes: Idle, Pairing, Paired and Connected

5.1.6 ENTER_BOOTLOADER (0x50)

The enter bootloader command is used instruct the dongle to enter the bootloader on the next re-enumeration. Because this is an atypical command that reconfigures the dongle, four Command Validation bytes are required for this command to be processed by the dongle.

CMD: 0x50

Payload Length: 4 bytes

Command Payload:

Command Validation
4 bytes

- Command Validation = 0xAA 0x55 0x39 0x93

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Modes: Idle, Pairing, Paired and Connected

5.2 REMOTE COMMANDS

5.2.1 START_DATA (0x20)

The start data command instructs the dongle to initiate data acquisition by the remote(s). All connected remotes will be put into data mode. To request data from only one remote when two remotes are connected to a single dongle, refer to the START_ASYMETRIC_DATA (0x2C) command.

CMD: 0x20

Payload Length: 1 byte

Command Payload:

Command Payload: NA

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.2 STOP_DATA (0x21)

The stop data command instructs the dongle to halt data acquisition by the remote(s).

CMD: 0x21

Payload Length: 0 bytes

Command Payload: NA

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Data

5.2.3 SET_SENSOR_CONFIG (0x22)

The set sensor configuration command is used to configure sensor parameters. The command payload contains a variable length packet depending on how many sensors are being configured. The Sensor ID and the Key Value are treated as a pair of data items, where 24 is the maximum allowed.

CMD: 0x22

Payload Length: 4 to 50 bytes

Command Payload:

Remote Number	**Number of Sensor Key Values**	**Sensor ID 1**	**Key Value 1**	**Sensor ID 2**	**Key Value 2**	**….**	**Sensor ID 24**	**Key Value 24**
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	….	1 byte	1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2
- Number of Sensor Key Values = number of Sensor ID/Key Value pairs in this packet. Range: 1-24
- Sensor ID X = sensor selection
- Key Value X = parameter setting

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.4 GET_SENSOR_CONFIG (0x23)

The get sensor configuration command is used to retrieve sensor parameters. The response payload contains a variable length packet depending on how many sensors have previously been configured. The Sensor ID and the Key Value are treated as a pair of data items, where 24 is the maximum allowed.

CMD: 0x23

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: Data

Response Payload Length: 4 to 50 bytes

Response Payload:

Remote Number	Number of Sensor Key Values	Sensor ID 1	Key Value 1	Sensor ID 2	Key Value 2	….	Sensor ID 24	Key Value 24
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	….	1 byte	1 byte

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Number of Sensor Key Values = number of Sensor ID/Key Value pairs in this packet. Range: 1-24
- Sensor ID X = sensor selection
- Key Value X = parameter setting

Valid Dongle Mode: Connected

5.2.5 SET_OUTPUT_CONFIG (0x24)

The set output configuration command is used to configure output ports. The command payload contains a variable length packet depending on how many outputs ports are being configured. The Output ID and the Key Value are treated as a pair of data items, where 24 is the maximum allowed.

CMD: 0x24

Payload Length: 4 to 50 bytes

Command Payload:

Remote Number	**Number of Output Key Values**	**Output ID 1**	**Key Value 1**	**Output ID 2**	**Key Value 2**	**….**	**Output ID 24**	**Key Value 24**
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	….	1 byte	1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2
- Number of Output Key Values = number of Output ID/Key Value pairs in this packet. Range: 1-24
- Output ID X = output selector
- Key Value X = parameter setting

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Modes: Connected and Data

5.2.6 GET_OUTPUT_CONFIG (0x25)

The get sensor output configuration command is used to retrieve sensor output status. The response payload contains a variable length packet depending on how many sensor outputs have previously been configured. The Output ID and the Key Values are treated as a pair of data items, where 24 is the maximum allowed.

CMD: 0x25

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: Data

Response Payload Length: 4 to 50 bytes

Response Payload:

Remote Number	Number of Output Key Values	Output ID 1	Key Value 1	Output ID 2	Key Value 2	….	Output ID 24	Key Value 24
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	….	1 byte	1 byte

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Number of Output Key Values = number of Output ID/Key Value pairs in this packet. Range: 1-24
- Output ID X = output selection
- Key Value X = parameter setting

Valid Dongle Mode: Connected

5.2.7 SET_FIXED_CONFIG (0x26)

The set fixed configuration command is used to configure the remote sensors in a pre-set fixed configuration. The command payload contains the index of the selected pre-set configuration.

CMD: 0x26

Payload Length: 2 bytes

Command Payload:

Remote Number	Fixed Config
1 byte	1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2
- Fixed Config = index of selected pre-set configuration to use in this remote

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.8 GET_FIXED_CONFIG (0x27)

The get fixed configuration command is used to retrieve the pre-set fixed configuration currently in use by the selected remote

CMD: 0x27

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: Data

Response Payload Length: 2 bytes

Response Payload:

Remote Number	Fixed Config
1 byte	1 byte

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Fixed Config = index of selected pre-set configuration in use by this remote

Valid Dongle Mode: Connected

5.2.9 GET_PACKET_CONFIG (0x28)

The get packet configuration command is used to retrieve the format of data packets received from the selected remote. The response payload contains a variable length packet depending on how many sensors have previously been configured. The Sensor ID and the Data Length are treated as a pair of data items, where 24 is the maximum allowed.

CMD: 0x28

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: Data

Response Payload Length: 4 to 50 bytes

Response Payload:

Remote Number	Number of Sensor Data Length Values	Sensor ID 1	Data Length 1	Sensor ID 2	Data Length 2	….	Sensor ID 24	Data Length 24
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	….	1 byte	1 byte

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Number of Sensor Data Length Values = number of Sensor ID/Data Length pairs in this packet. Range: 1-24. Only active sensors in the remote are included in this list
- Sensor ID X = sensor selection
- Data Length X = data length field allocated for this sensor in incoming packets

Valid Dongle Mode: Connected

5.2.10 GET_CALIBRATION (0x29)

The get calibration command returns the calibration values for sensors that contain this information. The actual data in the response will depend on the sensor.

CMD: 0x29

Payload Length: 2 bytes

Command Payload:

Remote Number	Sensor ID
1 byte	1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2
- Sensor ID = sensor selection

Command Response: Data

Response Payload Length: 4 to 50 bytes

Response Payload:

Remote Number	Sensor ID	Number of Calibration Bytes	Data
1 byte	1 byte	1 byte	1-47 bytes

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Sensor ID = selected sensor ID
- Number of Calibration Bytes = the number of bytes of calibration data in the response
- Data = calibration data for the selected sensor. See Data Protocol Document for more information about this data

Valid Dongle Mode: Connected

5.2.11 GET_REMOTE_STATUS (0x2A)

The get remote status command returns the current remote status and other ID/version information.

CMD: 0x2A

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: Data

Response Payload Length: 7 bytes

Response Payload:

Remote Number	Remote Sensor FW Version	Remote RF FW Version	Remote Battery Level
1 byte	2 bytes	2 bytes	2 bytes

- Remote Number = selected remote. 1=Remote1, 2=Remote2
- Remote Sensor FW Version = current remote sensor micro firmware version, MSB-LSB
- Remote RF FW Version = current remote RF micro firmware version, MSB-LSB
- Remote Battery Level = current battery voltage level at remote

Valid Dongle Mode: Connected

5.2.12 POWER_DOWN (0x2B)

The power down command is used to turn off the remote.

CMD: 0x2B

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to select. 1=Remote1, 2=Remote2

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.13 START_ASYMETRIC_DATA (0x2C)

The start asymmetric data command instructs the dongle to initiate data acquisition by the remotes. This command is intended to be used only when two remotes are currently connected to the dongle, and data is going to be requested from only one of the remotes. To request data from both remotes, use the START_DATA (0x20) command.

CMD: 0x2C

Payload Length: 1 byte

Command Payload:

Remote Number
1 byte

- Remote Number = which remote to receive data from. 1=Remote1, 2=Remote2

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.14 SLIP_RF_TIMING (0x2D)

The slip RF timing command instructs the dongle to “slip” its RF timing. After receiving this command, the dongle will jump two frequencies ahead in the four frequency sequence that is used by the RF protocol. This operation is intended to improve operation where two units with similar frequency sets are interfering with each other.

CMD: 0x2D

Payload Length: 0 bytes

Command Payload: NA

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Modes: Connected and Data

5.2.15 FREQUENCY_CONTROL (0x2E)

The frequency control command instructs the dongle to stop or re-start using a selected radio frequency while communicating with its remote(s). After receiving this command, the dongle either stop or re-start all communications on the selected frequency channel.

This operation is intended to improve operation where two units with similar frequency sets are interfering with each other.

CMD: 0x2E

Payload Length: 2 bytes

Command Payload:

Control Operation	Control Frequency
1 byte	1 byte

- Control Operation = select stop or re-start using frequency. 0=stop using frequency, 1=re-start using frequency
- Control Frequency = select which of the four frequency channels to control. Range is 1 through 4

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Modes: Connected and Data

5.2.16 OTA_UPGRADE_START (0x60)

The OTA Upgrade start command is used initiate over-the-air (OTA) firmware upgrade of the remote radio micro device. Following this OTA Upgrade command, multiple OTA Data commands will follow, containing the new firmware version for that remote radio micro. After the OTA Data commands are complete, an OTA CRC command will complete the firmware upgrade to the remote radio micro.

Because this is an atypical command that may reconfigure a remote, four Command Validation bytes are required for this command to be processed by the dongle.

CMD: 0x60

Payload Length: 5 bytes

Command Payload:

Remote Number	Command Validation
1 byte	4 bytes

- Remote Number = which remote to initiate OTA with. 1=Remote1, 2=Remote2
- Command Validation = 0xBB 0xCC 0x27 0x72

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.17 OTA_UPGRADE_DATA (0x61)

Multiple OTA Data commands are sent to upgrade the firmware version of a remote radio micro. After the OTA Data commands are complete, an OTA CRC command will complete the firmware upgrade to the remote radio micro.

CMD: 0x61

Payload Length: 34 bytes

Command Payload:

Code Location	Code Data
2 bytes	32 bytes

- Code Location = location in remote radio micro Flash where Code Data is to be stored
- Code Data = firmware upgrade data for new code version

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.18 OTA_UPGRADE_CRC (0x62)

The OTA CRC command completes the firmware upgrade to the remote radio micro. The CRC sent with this command is used by the remote radio micro to validate the received copy of updated code.

CMD: 0x62

Payload Length: 2 bytes

Command Payload:

CRC
2 bytes

- CRC = CRC of firmware image sent to remote radio micro in previous OTA Data commands

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.19 REMOTE_UPGRADE_START (0x70)

The Remote Upgrade start command is used initiate over-the-air (OTA) firmware upgrade of the remote sensor micro device. Following this Remote Upgrade command, multiple Remote Data commands will follow, containing the new firmware version for that remote sensor micro. After the Remote Data commands are complete, a Remote CRC command will complete the firmware upgrade to the remote sensor micro.

Because this is an atypical command that may reconfigure a remote, four Command Validation bytes are required for this command to be processed by the dongle.

CMD: 0x70

Payload Length: 5 bytes

Command Payload:

Remote Number	Command Validation
1 byte	4 bytes

- Remote Number = which remote to initiate OTA with. 1=Remote1, 2=Remote2
- Command Validation = 0x44 0x77 0x6C 0xC6

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.20 REMOTE_UPGRADE_DATA (0x71)

Multiple Remote Data commands are sent to upgrade the firmware version of a remote sensor micro. After the Remote Data commands are complete, a Remote CRC command will complete the firmware upgrade to the remote sensor micro.

CMD: 0x71

Payload Length: 35 bytes

Command Payload:

Code Location	Code Data
3 bytes	32 bytes

- Code Location = location in remote sensor micro Flash where Code Data is to be stored
- Code Data = firmware upgrade data for new code version

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.2.21 REMOTE_UPGRADE_CRC (0x72)

The Remote CRC command completes the firmware upgrade to the remote sensor micro. The CRC sent with this command is used by the remote sensor micro to validate the received copy of updated code.

CMD: 0x72

Payload Length: 2 bytes

Command Payload:

CRC
2 bytes

- CRC = CRC of firmware image sent to remote radio sensor in previous Remote Data commands

Command Response: ACK

Response Payload Length: NA

Response Payload: NA

Valid Dongle Mode: Connected

5.3 DATA RESPONSES (ASYNC)

As the dongle manages the RF link or receives sensor data from the remotes, the dongle will send asynchronous packets to the PC not directly associated with a command from the PC application.

5.3.1 RF_CONNECTION (0x40)

Changes in RF connection status to either of the remotes are provided by this data packet. As remotes finish pairing, the RF paired event will be sent. As paired remotes come into RF range (for example, turning on), the RF connected event will be sent. As remotes go out of RF range or get turned off, the RF disconnected event will be sent.

CMD: 0x40

Response Payload Length: 2 bytes

Response Payload:

Remote Number	RF Status
1 byte	1 byte

- Remote Number = active remote. 1=Remote1, 2=Remote2
- RF Status = new RF connection status. 2=Paired, 1=Connected, 0=Disconnected

Valid Dongle Modes: Paired, Connected and Data

5.3.2 DATA_FROM_REMOTE (0x41)

The data from remote packet is sent as data is received by the dongle from its paired remotes. This is expected to occur at 10 msec intervals from each remote.

CMD: 0x41

Response Payload Length: 5 to 104 bytes

Response Payload:

Remote Number	Frame Number	RF Statistics	Remote Data	RSSI
1 byte	1 byte	1 byte	1 to 100 bytes	1 byte

- Remote Number = remote that sent data. 1=Remote1, 2=Remote2
- Frame Number = current frame counter, incremented every 10 msec. If an RF message is not received in any given 10 msec frame, then no data packet with that frame number will be sent to the PC. This can be used to monitor "lost" RF packets
- RF Statistics = RF packet statistics. This byte indicates which of the four active frequencies was used in the reception of this packet. 0=Frequency 1, 1=Frequency 2, 2=Frequency 3, 3=Frequency 4
- Remote Data = data as received from remote
- RSSI = receive signal strength value of this received packet (from remote)

Valid Dongle Mode: Data

6.0 IOLAB USB INTERFACE

The dongle implements the USB Communication Device Class (CDC). This specification includes USB virtual communications port devices that allow a USB device to appear to the PC as an ordinary UART communications port which provides straightforward PC application development to and from the dongle.

This configuration uses a native Windows driver for this operation but requires an information file (iolab.inf) to be installed along with the IOLab PC application.

Figure 6 shows the descriptor organization for the dongle.

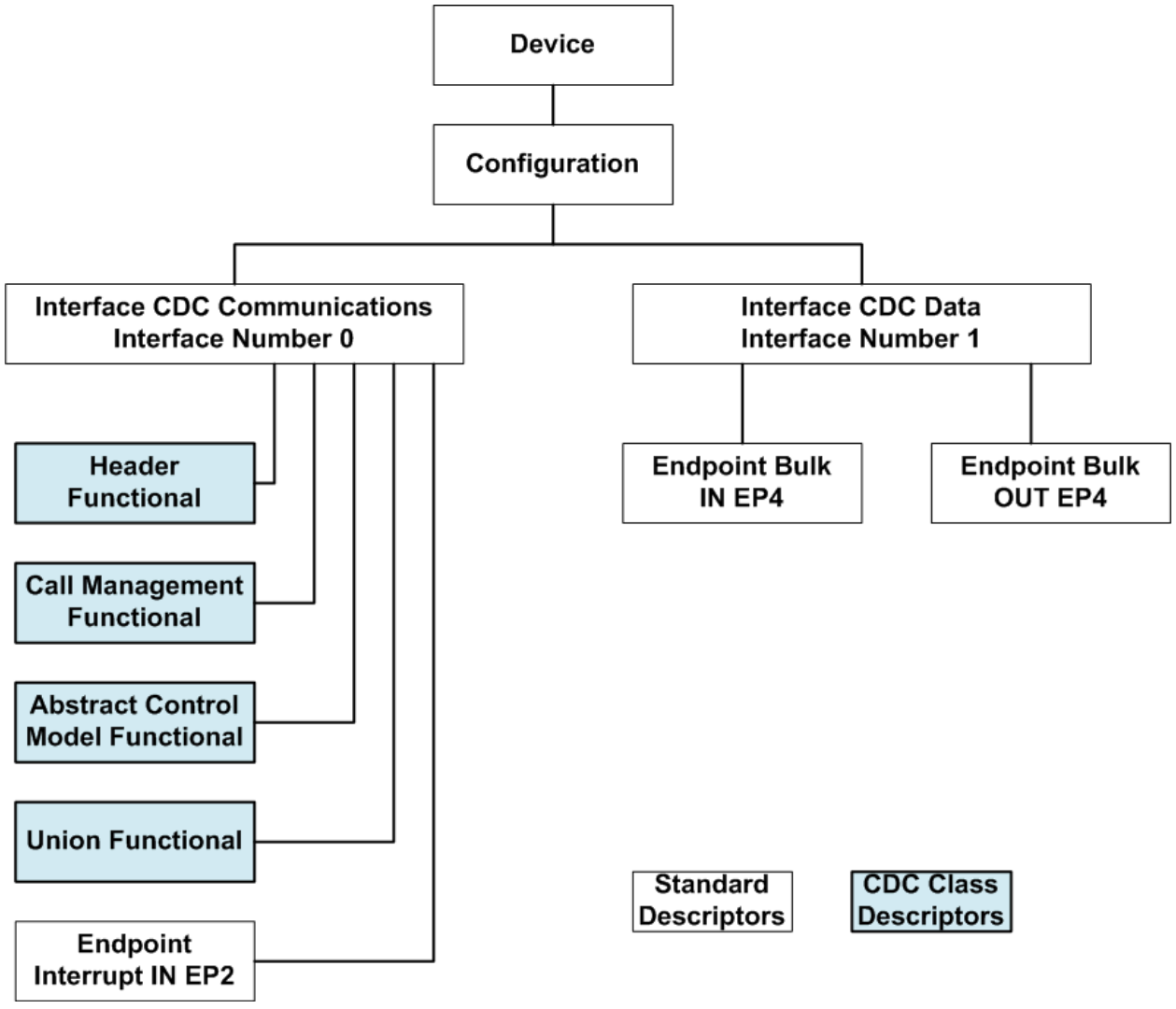


Figure 6 - USB Descriptor Hierarchy for a CDC Device

6.1 INFORMATION FILE (IOLAB.INF)

When connecting the IOLab dongle to a PC for the first time the iolab.inf file will be needed to properly install the driver for the dongle. The IOLab PC application installation should load this file on the target PC.

Appendix A shows the steps for a Windows XP installation of the dongle. Unfortunately Windows treats the .inf file as part of the PC's driver installation process, so a hardware warning message of "This driver is not digitally signed!" is displayed even though a native Windows driver is being installed for the dongle.

6.2 IOLAB DEVICE DESCRIPTORS

Table 3 below shows the descriptor format for the IOLab virtual communications device. Descriptors shown in blue are associated with the CDC class and other descriptors are standard non-class related descriptors.

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor in bytes
1	bDescriptorType	1	0x01	**DEVICE** Descriptor Type
2	bcdUSB	2	0x0200	Device compliant to USB spec version number 2.00
4	bDeviceClass	1	0x02	Device Class = CDC
5	bDeviceSubClass	1	0x00	Subclass is described in the interface
6	bDeviceProtocol	1	0x00	Device protocol is defined in the interface
7	bMaxPacketSize	1	0x20	EP0 Max Packet Size is 32 bytes
8	idVendor	2	0x1881	Vendor ID VID = 0x1881
10	idProduct	2	0x0400	Product ID PID = 0x0400
12	bcdDevice	2	0x0001	Device Release Number FW Version =00.01
14	iManufacturer	1	0x01	Index for manufacturer string
15	iProduct	1	0x02	Index for Manufacturer String
16	iSerialNumber	1	0x03	Index for Serial number string
17	bNumConfigurations	1	0x01	Device has a Single USB configuration

Table 3 - Device Descriptor

6.2.1 Configuration Descriptor

The Configuration Descriptor is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

The Configuration descriptor informs the host of how many bytes are contained in the configuration descriptor, the number of interfaces used by the device and how much power the device draws from the host.

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor in bytes
1	bDescriptorType	1	0x02	**CONFIGURATION** Descriptor Type
2	wTotalLength	2	0x0043	Device configuration contains 67 decimal bytes
4	bNumInterfaces	1	0x02	Device has two interfaces
5	bConfigurationValue	1	0x01	The number the Host uses to set the device configuration
6	iConfiguration	1	0x00	Index for configuration string index. Not used
7	bmAttributes	1	0x80	Bus powered and remote wake up
8	MaxPower	1	0x00	Max Power from the buss is = 0 mA

Table 4 - Configuration Descriptor (9-byte)

6.2.2 Interface Descriptor - Communications (IF0)

This Descriptor is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor in bytes
1	bDescriptorType	1	0x04	**INTERFACE** Descriptor Type
2	bInterfaceNumber	1	0x00	The number of this interface= 0
3	bAlternateSetting	1	0x00	Value use to select the alternate setting of this interface=0 (none)
4	bNumEndpoints	1	0x01	Number of endpoints used by this

				interface=1(excluding EP0)
5	bInterfaceClass	1	0x02	Class=CDC
6	bInterfaceSubClass	1	0x02	SubClass=Abstract Control Class
7	bInterfaceProtocol	1	0x01	Protocol=AT Commands
8	iInterface	1	0x00	Interface string index

Table 5 - Interface Zero Descriptor

6.2.3 Functional Descriptor – Header

The Functional Descriptor – Header is a Class Specific Descriptor for CDC Devices and can be found in Universal Serial Bus Class Definitions for Communications Devices.

This descriptor is one of four CDC functional descriptors. The Functional descriptors are in sequential order and can be found in Table 13 of Universal Serial Bus Class Definitions for Communications Devices.

Offset	Field	Size	Value	Description
0	bFunctionalLength	1	0x05	Size of this descriptor in bytes
1	bDescriptorType	1	0x24	**HEADER** Functional Descriptor Type
2	bDescriptorSubType	1	0x00	
3	bAlternateSetting	1	0x10	
4	bNumEndpoints	1	0x01	

Table 6 - Functional Header Descriptor

6.2.4 Functional Descriptor - Call Management

The Functional Descriptor – Call Management is a Class Specific Descriptor for CDC Devices and can be found in Universal Serial Bus Class Definitions for Communications Devices.

Offset	Field	Size	Value	Description
0	bFunctionalLength	1	0x05	Size of this descriptor in bytes
1	bDescriptorType	1	0x24	**HEADER** Functional Descriptor Type
2	bDescriptorSubType	1	0x01	
3	bAlternateSetting	1	0x00	

4	bNumEndpoints	1	0x01	

Table 7 - Functional Descriptor - Call Management

6.2.5 Functional Descriptor - Abstract Control Model

The Functional Descriptor – Abstract Control Model is a Class Specific Descriptor for CDC Devices and can be found in Universal Serial Bus Class Definitions for Communications Devices.

Offset	Field	Size	Value	Description
0	bFunctionalLength	1	0x04	Size of this descriptor in bytes
1	bDescriptorType	1	0x24	**HEADER** Functional Descriptor Type
2	bDescriptorSubType	1	0x02	
3	bAlternateSetting	1	0x00	

Table 8 - Functional Descriptor - Abstract Control Model

6.2.6 Functional Descriptor - Union

The Functional Descriptor - Union is a Class Specific Descriptor for CDC Devices and can be found in Universal Serial Bus Class Definitions for Communications Devices.

Offset	Field	Size	Value	Description
0	bFunctionalLength	1	0x05	Size of this descriptor in bytes
1	bDescriptorType	1	0x24	**HEADER** Functional Descriptor Type
2	bDescriptorSubType	1	0x06	
3	bAlternateSetting	1	0x00	
4	bNumEndpoints	1	0x01	

Table 9 - Functional Descriptor - Union

6.2.7 Endpoint Descriptor - Interrupt (EP1-IN)

The Endpoint Descriptor - Interrupt is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor in bytes
1	bDescriptorType	1	0x05	**ENDPOINT** Descriptor Type
2	bEndpointAddress	1	0x81	Endpoint Address= EP1-IN
3	bmAttributes	1	0x03	Transfer Type=Interrupt
4	wMaxPacketSize	2	0x0020	Maximum packet size for this endpoint=32 bytes
6	bInterval	1	0x02	Polling Interval of this endpoint in milliseconds. Range 1 to 255. Ignored for Bulk and Control Endpoints.

Table 10 - Endpoint Descriptor - Interrupt IN

6.2.8 Interface Descriptor - Data (IF1)

The Interface Descriptor - Data is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor in bytes
1	bDescriptorType	1	0x04	**INTERFACE** Descriptor Type
2	bInterfaceNumber	1	0x01	
3	bAlternateSetting	1	0x00	
4	bNumEndpoints	1	0x02	
5	bInterfaceClass	1	0x0A	Class Code= CDC Device
6	bInterfaceSubClass	1	0x00	
7	bInterfaceProtocol	1	0x00	
8	iInterface	1	0x00	

Table 11 - Interface Descriptor - Data

6.2.9 Endpoint Descriptor - Bulk Data IN (EP2-IN)

The Endpoint Descriptor – Bulk Data IN is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

Offset	Field	Size	Value	Description
0	bLength	1	0x00	Size of this descriptor in bytes
1	bDescriptorType	1	0x00	**ENDPOINT** Descriptor Type
2	bEndpointAddress	1	0x00	Endpoint Address
3	bmAttributes	1	0x00	Endpoint Type **BULK**
4	wMaxPacketSize	2	0x0040	Maximum packet size for this endpoint=64 bytes
6	bInterval	1	0x00	Polling Interval= Ignored for Bulk Endpoints.

Table 12 - Endpoint Descriptor - Bulk Data IN

6.2.10 Endpoint Descriptor - Bulk Data OUT (EP3-OUT)

The Endpoint Descriptor – Bulk Data OUT is a Standard USB descriptor and can be found in Chapter 9 of the Universal Serial Bus Specification.

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor in bytes
1	bDescriptorType	1	0x05	**ENDPOINT** Descriptor Type
2	bEndpointAddress	1	0x03	Endpoint Address=OUT EP3
3	bmAttributes	1	0x00	Endpoint Type **BULK**
4	wMaxPacketSize	2	0x0040	Maximum packet size for this endpoint=64 decimal bytes
6	bInterval	1	0x00	Polling Interval= Ignored for Bulk Endpoints.

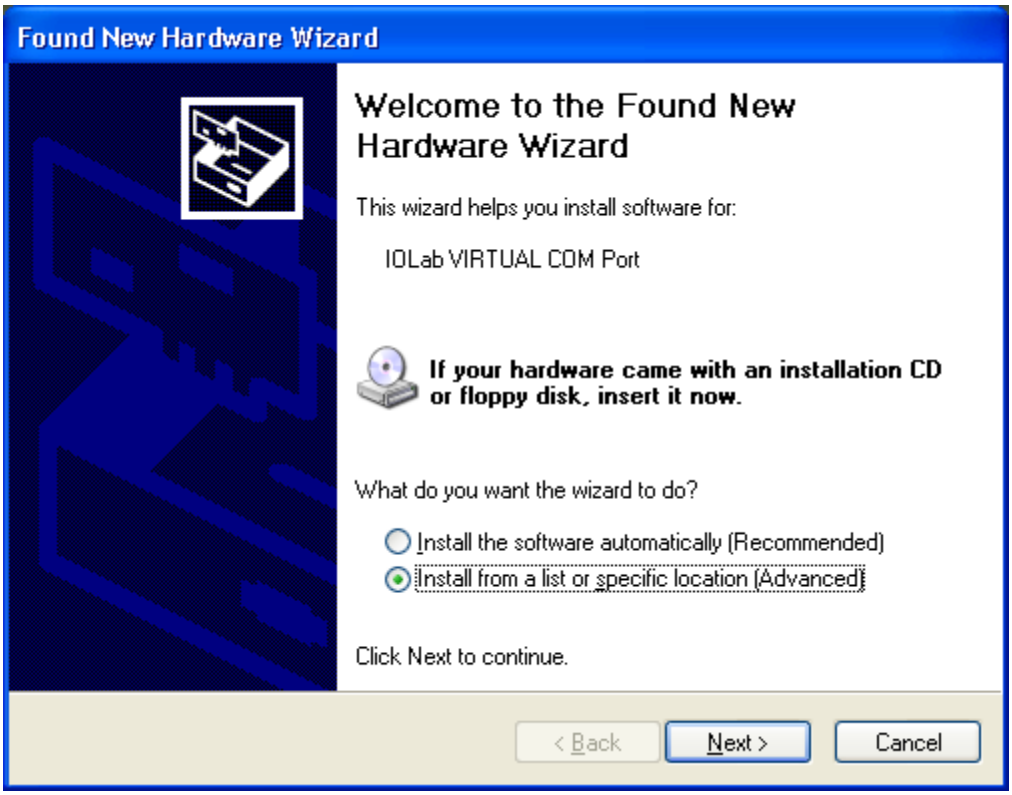
Table 13 - Endpoint Descriptor - Bulk Data OUT

7.0 IOLAB WIN XP INSTALLATION STEPS

When the IOLab dongle is inserted into any PC for the first time, the user must direct Windows to find the proper .inf file to allow the dongle to work properly. Upon insertion, the "Found New Hardware Wizard" should automatically pop up.

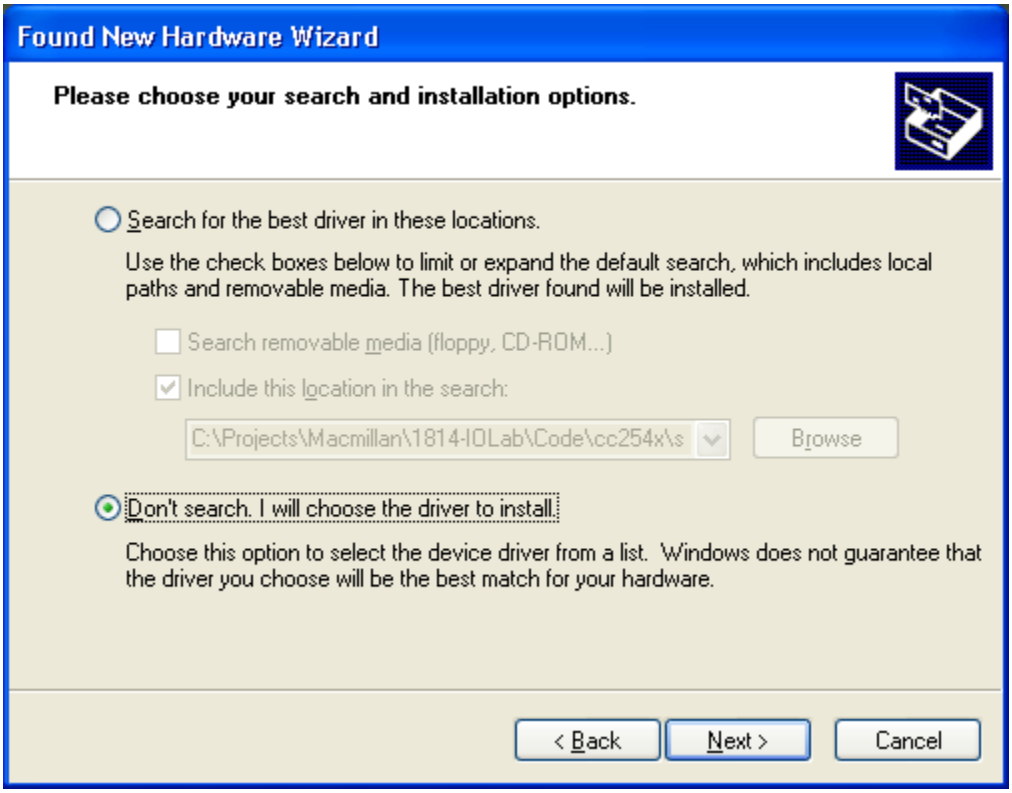
7.1 STEP 1

Select "Install from a list or specific location (Advanced)" and press the "Next" button. Note that your screen will probably not show the text "IOLab VIRTUAL COM Port" shown in the screen capture below.



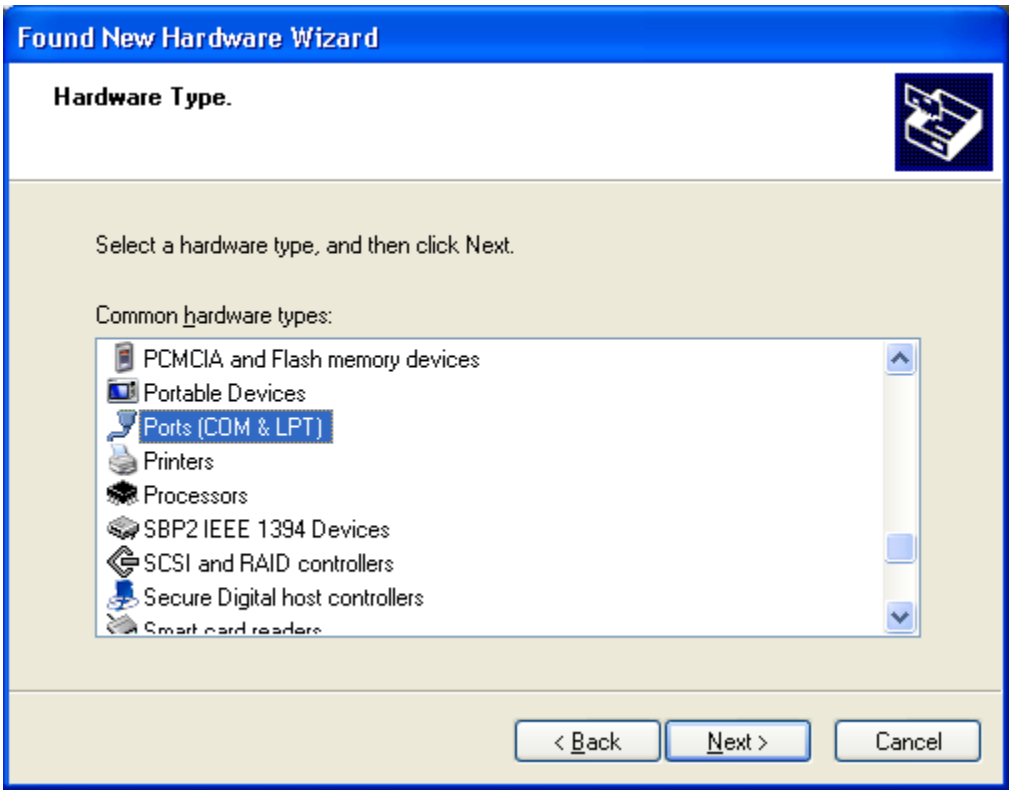
7.2 STEP 2

Select “Don’t search. I will choose the driver to install” and press the “Next” button.



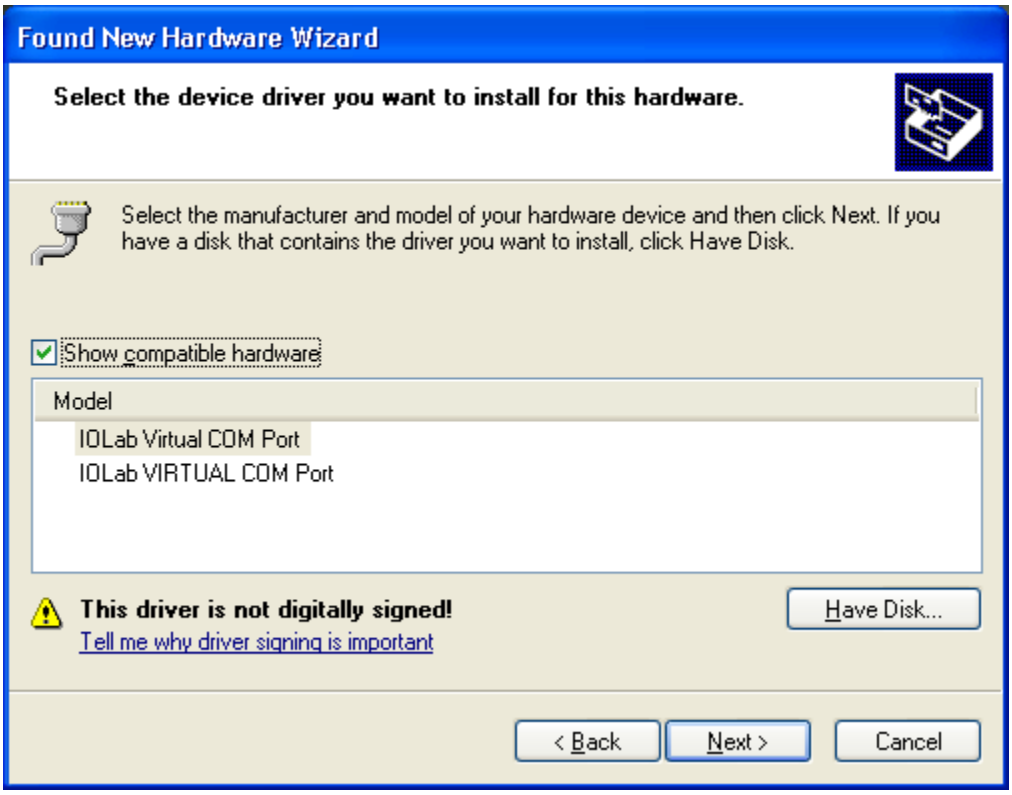
7.3 STEP 3

The found new hardware process will ask what type of device is being installed. Select "Ports (COM & LPT)" and press the "Next" button.



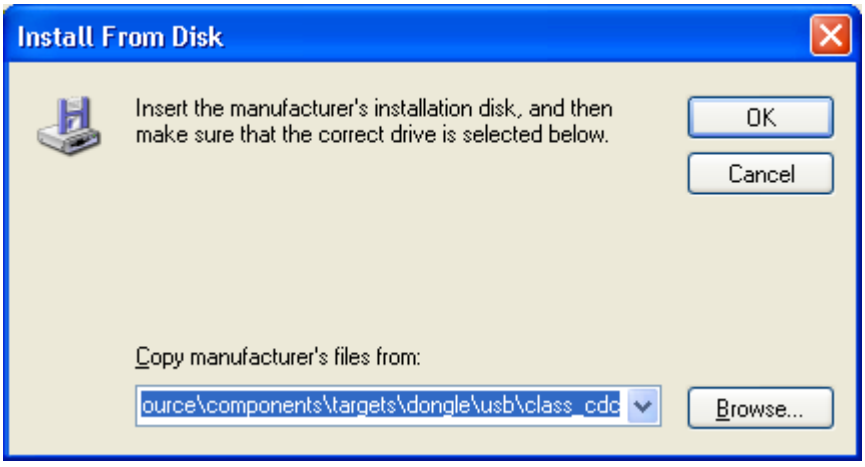
7.4 STEP 4

Press the “Have Disk…” button. Note that your screen will probably not show text “IOLab VIRTUAL COM Port” show in the screen capture below.



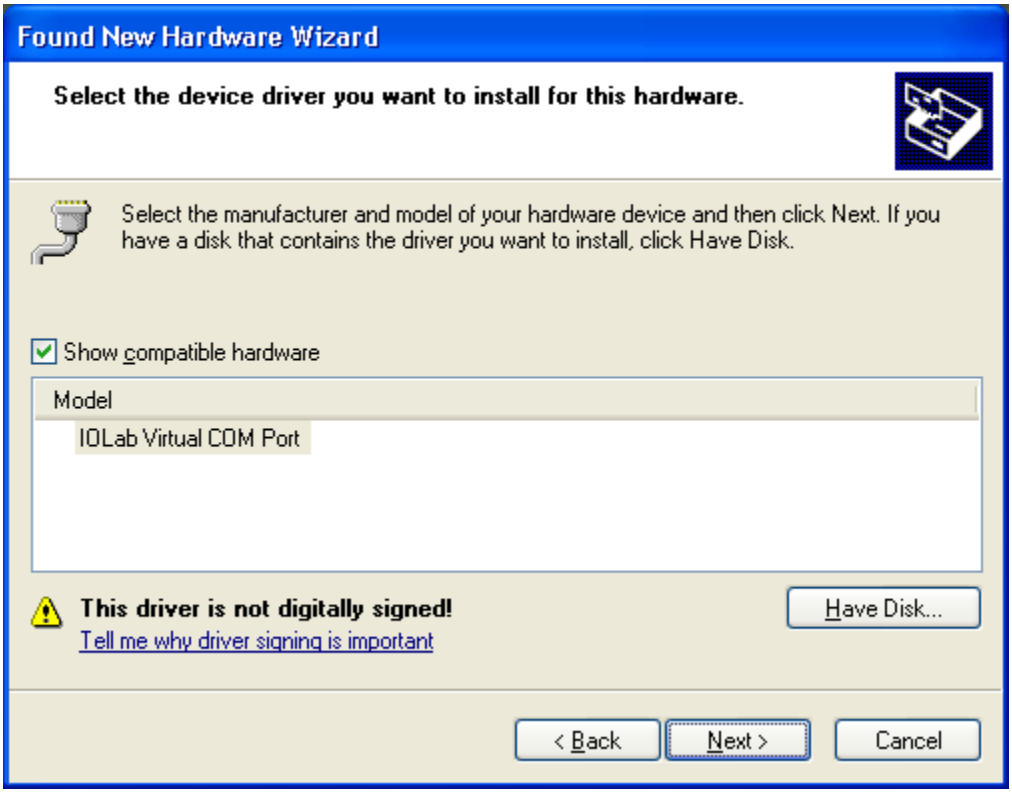
7.5 STEP 5

Press the “Browse…” button and then navigate to the iolab.inf file located on your PC and press the “OK” button.



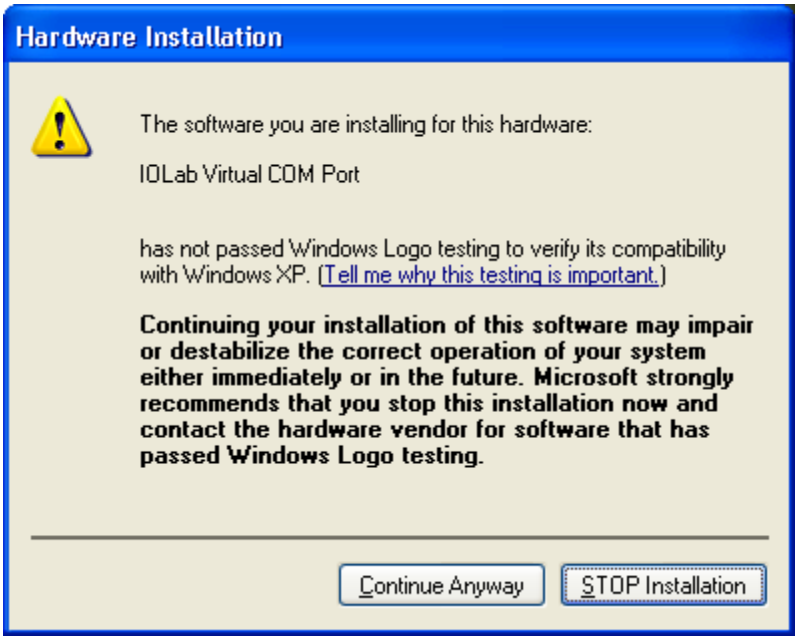
7.6 STEP 6

Press the “Next” button



7.7 STEP 7

Press the “Continue Anyway” button.



7.8 STEP 8

Press the “Finish” button.

7.9 STEP 9

You should see a message of "Your new hardware is installed and ready to use". Open the Windows device manager and make sure that you screen looks like the figure below. Note that your COM port number may be different.