Python POD API

Release v1.5.0

Thresa Kelly

CONTENTS:

1	1 Python POD API Project						
	1.1	PodAp	oi package	·	. 1		
		1.1.1	Subpacl	kages	. 1		
			1.1.1.1	PodApi.Commands package	. 1		
			1.1.1.2	PodApi.Devices package			
			1.1.1.3	PodApi.Packets package	. 28		
			1.1.1.4	PodApi.Parameters package	. 44		
			1.1.1.5	PodApi.Stream package			
		1.1.2	Module	e contents			
	1.2	Setup					
		1.2.1		kages			
			1.2.1.1	Setup.Inputs package			
			1.2.1.2	Setup.SetupAllDevices package			
			1.2.1.3	Setup.SetupOneDevice package			
		1.2.2	Module	e contents			
2 Indices and tables							
Рy	Python Module Index						
In	Index						

PYTHON POD API PROJECT

1.1 PodApi package

1.1.1 Subpackages

1.1.1.1 PodApi.Commands package

Submodules

PodApi.Commands.PodCommands module

class PodApi.Commands.PodCommands.CommandSet

Bases: object

POD_Commands manages a dictionary containing available commands for a POD device.

commands

Dictionary containing the available commands for a POD device. Each entry is formatted as { key(command number) : value([command name, number of argument ASCII bytes, number of return bytes, binary flag) }.

Type

dict[int,list[str|tuple[int]|bool]]

AddCommand(commandNumber: int, commandName: str, argumentBytes: tuple[int], returnBytes: tuple[int], isBinary: bool, description: str) \rightarrow bool

Adds a command entry to the current commands dictionary (__commands) if the command does not exist.

Parameters

- **commandNumber** (*int*) Integer of the command number.
- **commandName** (*str*) String of the command's name.
- **argumentBytes** (tuple[int]) Integer of the number of bytes in the argument.
- returnBytes (tuple[int]) Integer of the number of bytes in the return.
- **isBinary** (*bool*) Boolean flag to mark if the command is binary (True) or standard (False).
- **description** (*str*) String description of the command.

True if the command was successfully added, False if the command could not be added because it already exists.

Return type

bool

ArgumentHexChar($cmd: int \mid str$) \rightarrow tuple[int] | None

Gets the tuple for the number of hex characters in the argument for a given command.

Parameters

cmd (int | str) – Integer command number or string command name.

Returns

Tuple representing the number of bytes in the argument for cmd. If the command could not be found, return None.

Return type

tuple[int]|None

CommandNumberFromName(name: str) \rightarrow int | None

Gets the command number from the command dictionary using the command's name.

Parameters

name (str) – String of the command's name.

Returns

Integer representing the command number. If the command could not be found, return None.

Return type

int|None

Description($cmd: int \mid str$) $\rightarrow str \mid None$

Gets the description for a given command.

Parameters

cmd (int | str) – Integer command number or string command name.

Returns

String description for the command. If the command could not be found, return None.

Return type

str|None

DoesCommandExist($cmd: int \mid str$) \rightarrow bool

Checks if a command exists in the __commands dictionary.

Parameters

cmd (int | str) – Integer command number or string command name.

Returns

True if the command exists, false otherwise.

Return type

bool

static GetBasicCommands() → dict[int, list[str | tuple[int] | bool]]

Creates a dictionary containing the basic POD command set (0,1,2,3,4,5,6,7,8,9,10,11,12).

Returns

Dictionary containing the available commands for this POD device. Each entry is formatted as

{ key(command number) : value([command name, number of argument ASCII bytes, number of return bytes, binary flag, description) }.

Return type

dict[int,list[str|tuple[int]|bool|str]]

$\textbf{GetCommands()} \rightarrow dict[int, list[str \mid tuple[int] \mid bool]]$

Gets the contents of the current command dictionary (__commands).

Returns

Dictionary containing the available commands for a POD device. Each entry is formatted as { key(command number) : value([command name, number of argument ASCII bytes, number of return bytes, binary flag, description) }.

Return type

dict[int, list[str|tuple[int]|bool|str]]

IsCommandBinary($cmd: int \mid str$) \rightarrow bool | None

Gets the binary flag for a given command.

Parameters

cmd (*int* / *str*) – Integer command number or string command name.

Returns

Boolean flag that is True if the command is binary and False if standard. If the command could not be found, return None.

Return type

bool|None

static NoValue() \rightarrow int

Gets value of __NOVALUE.

Returns

Value of __NOVALUE.

Return type

int

RemoveCommand($cmd: int \mid str$) \rightarrow bool

Removes the entry for a given command in __commands dictionary.

Parameters

cmd (int | str) – Integer command number or string command name.

Returns

True if the command was successfully removed, False if the command does not exist.

Return type

bool

RestoreBasicCommands() \rightarrow None

Sets the current commands (__commands) to the basic POD command set.

ReturnHexChar($cmd: int \mid str$) \rightarrow tuple[int] | None

Gets the tuple for the number of hex characters in the return for a given command.

Parameters

cmd (int | str) – Integer command number or string command name.

Tuple representing the number of hex characters in the return for cmd. If the command could not be found, return None.

Return type

tuple[int]|None

Search(*cmd*: *int* | *str*, *idx*: *int* | *None* = *None*) \rightarrow str | tuple[int] | bool | None

Searches the __commands dictionary for the command.

Parameters

- cmd (int | str) Integer command number or string command name.
- idx (int, optional) Index for the desired value in the command information list. Defaults to None.

Returns

If an idx was given, this returns the idx value of the command information list if the command was found (None otherwise). If no idx is given, this returns true if the command is found (False otherwise).

Return type

str|tuple[int]|bool|str|None

```
static U16() \rightarrow int
```

Gets value of U16.

Returns

Value of __U16.

Return type

int

static $U32() \rightarrow int$

Gets value of __U32.

Returns

Value of __U32.

Return type

int

static U8() \rightarrow int

Gets value of U8.

Returns

Value of __U8.

Return type

int

ValidateCommand(cmd: str | int, pld: int | bytes | tuple[int | bytes] | None = None)

Raises an exception if the command and its payload are invalid for this POD device.

Parameters

- **cmd** (str / int) Command name or number.
- pld (int | bytes | tuple[int | bytes] | None, optional) Standard command packet payload. Defaults to None.

Raises

- **Exception** Command '+str(cmd)+' does not exist.
- **Exception** This command does not take a payload.
- **Exception** This command requires a payload.
- **Exception** Command needs more than one argument in the payload. Use a tuple of values.
- **Exception** Payload must have '+str(sum(args))+' bytes.
- **Exception** Payload must have '+str(len(args))+' integer items in the tuple.
- Exception Bytes in the payload are the wrong sizes. The sizes must be '+str(args)+'.'
- **Exception** The payload tuple must only contain int or bytes items.
- Exception Payload is of incorrect type. It must be an int, bytes, or tuple of int/bytes.

__ARGUMENTS: int = 1

Class-level integer representing the index key for the number of bytes in an argument for __commands list values.

$__BINARY: int = 3$

Class-level integer representing the index key for the binary flag for __commands list values.

__DESCRIPTION: int = 4

Class-level integer representing the index key for the description for __commands list values.

$_{\text{NAME}}$: int = 0

Class-level integer representing the index key for the command name for __commands list values.

$__NOVALUE: int = -1$

Class-level integer used to mark when a list item in __commands means 'no value' or is undefined.

__RETURNS: int = 2

Class-level integer representing the index key for the number of bytes in the return for __commands list values.

$_{_{_{_{_{_{}}}}}}$ U16: int = 4

Class-level integer representing the number of hexadecimal characters for an unsigned 16-bit value.

$_{\tt U32:}$ int = 8

Class-level integer representing the number of hexadecimal characters for an unsigned 32-bit value.

$__U8: int = 2$

Class-level integer representing the number of hexadecimal characters for an unsigned 8-bit value.

Module contents

1.1.1.2 PodApi.Devices package

Subpackages

PodApi.Devices.SerialPorts package

Submodules

PodApi.Devices.SerialPorts.PortAccess module

class PodApi.Devices.SerialPorts.PortAccess.FindPorts

Bases: object

Contains methods for the user to view and select a serial port.

```
static ChoosePort(forbidden: list[str] = []) \rightarrow str
```

Systems checks user's Operating System, and chooses ports accordingly.

Parameters

forbidden (list[str], optional) – List of port names that the user should not use. This may be because these ports are already in use or that the port is not a POD device. Defaults to Π .

Returns

String name of the port.

Return type

str

static GetAllPortNames() \rightarrow list[str]

Finds all the available COM ports on the user's computer and appends them to an accessible list.

Returns

List containing the names of available COM ports.

Return type

list[str]

$\textbf{static GetSelectPortNames}(\textit{forbidden: list[str]} = []) \rightarrow list[str]$

Gets the names of all available ports.

Parameters

forbidden (list[str], optional) – List of port names that the user should not use. This may be because these ports are already in use or that the port is not a POD device. Defaults to [].

Returns

List of port names.

Return type

list[str]

$\textbf{static _ChoosePortLinux}(\textit{forbidden: list[str]} = []) \rightarrow \textit{str}$

User picks Serial port in Linux.

Parameters

forbidden (list[str], optional) – List of port names that the user should not use. This may be because these ports are already in use or that the port is not a POD device. Defaults to [].

Returns

String name of the port.

Return type

stı

static _ChoosePortWindows($forbidden: list[str] = []) \rightarrow str$

User picks COM port in Windows.

Parameters

forbidden (list[str], optional) – List of port names that the user should not use. This may be because these ports are already in use or that the port is not a POD device. Defaults to [].

Returns

String name of the port.

Return type

str

PodApi.Devices.SerialPorts.SerialComm module

```
class PodApi.Devices.SerialPorts.SerialComm.PortIO(port: str | int, baudrate: int = 9600)
```

Bases: object

COM_io handles serial communication (read/write) using COM ports.

__serialInst

Instance-level serial COM port.

Type

Serial

$CloseSerialPort() \rightarrow None$

Closes the instance serial port if it is open.

Flush() \rightarrow bool

Reset the input and output serial buffer.

Returns

True of the buffers are flushed, False otherwise.

Return type

bool

GetPortName() \rightarrow str | None

Gets the name of the open port.

Returns

If the serial port is open, it will return a string of the port's name. If the port is closed, it will return None.

Return type

str|None

$\textbf{IsSerialClosed()} \rightarrow bool$

Returns False if the serial instance port is open, True otherwise.

Returns

True if the COM port is closed, False otherwise.

Return type

bool

$\textbf{IsSerialOpen()} \rightarrow bool$

Returns True if the serial instance port is open, false otherwise.

Returns

True if the COM port is open, False otherwise.

Return type

bool

OpenSerialPort(*port*: $str \mid int, baudrate$: int = 9600) \rightarrow None

First, it closes the serial port if it is open. Then, it opens a serial port with a set baud rate.

Parameters

- **port** (*str* / *int*) String of the serial port to be opened.
- baudrate (int, optional) Integer baud rate of the opened serial port. Defaults to 9600.

Raises

Exception – Port does not exist.

Read($numBytes: int, timeout_sec: int | float = 5) \rightarrow bytes | None$

Reads a specified number of bytes from the open serial port.

Parameters

- **numBytes** (*int*) Integer number of bytes to read.
- timeout_sec (int|float, optional) Time in seconds to wait for serial data. Defaults to 5.

Raises

Exception – Timeout for serial read.

Returns

If the serial port is open, it will return a set number of read bytes. If it is closed, it will return None.

Return type

bytes|None

ReadLine() \rightarrow bytes | None

Reads until a new line is read from the open serial port.

Returns

If the serial port is open, it will return a complete read line. If closed, it will return None.

Return type

bytes|None

ReadUntil(eol: bytes) \rightarrow bytes | None

Reads until a set character from the open serial port.

Parameters

eol (*bytes*) – end-of-line character.

Returns

If the serial port is open, it will return a read line ending in eol. If closed, it will return None.

Return type

bytes|None

SetBaudrate(baudrate: int) \rightarrow bool

Sets the baud rate of the serial port

Parameters

baudrate (*int*) – Baud rate, or signals per second.

True if the baudrate was set, False otherwise.

Return type

bool

Write(message: bytes) \rightarrow None

Write a set message to the open serial port.

Parameters

message (*bytes*) – byte string containing the message to write.

```
__BuildPortName(port: str \mid int) \rightarrow str
```

Converts the port parameter into the "COM"+<number> format for Windows or "/dev/tty..."+<number> for Linux.

Parameters

port (str / int) – Name of a COM port. Can be an integer or string.

Returns

Name of the COM port.

Return type

str

Module contents

Submodules

PodApi.Devices.BasicPodProtocol module

class PodApi.Devices.BasicPodProtocol.**Pod**(port: str | int, baudrate: int = 9600)

Bases: object

POD_Basics handles basic communication with a generic POD device, including reading and writing packets and packet interpretation.

_port

Instance-level COM_io object, which handles the COM port

Type

COM_io

_commands

Instance-level POD_Commands object, which stores information about the commands available to this POD device.

Type

POD_Commands

static BuildPODpacket_Standard($commandNumber: int, payload: bytes \mid None = None$) \rightarrow bytes

Builds a standard POD packet as bytes: STX (1 byte) + command number (4 bytes) + optional packet (? bytes) + checksum (2 bytes)+ ETX (1 bytes).

Parameters

• **commandNumber** (*int*) – Integer representing the command number. This will be converted into a 4 byte long ASCII-encoded bytes string.

• payload (bytes | None, optional) — bytes string containing the payload. Defaults to None.

Returns

Bytes string of a complete standard POD packet.

Return type

bytes

static Checksum(bytesIn: bytes) \rightarrow bytes

Calculates the checksum of a given bytes message. This is achieved by summing each byte in the message, inverting, and taking the last byte.

Parameters

bytesIn (bytes) – Bytes message containing POD packet data.

Returns

Two ASCII-encoded bytes containing the checksum for bytesIn.

Return type

bytes

static ChoosePort($forbidden: list[str] = []) \rightarrow str$

Systems checks user's Operating System, and chooses ports accordingly.

Parameters

forbidden (list[str], optional) – List of port names that are already used. Defaults to [].

Returns

String name of the port.

Return type

stı

$FlushPort() \rightarrow bool$

Reset the input and output serial port buffer.

Returns

True of the buffers are flushed, False otherwise.

Return type

bool

GetDeviceCommands() \rightarrow dict[int, list[str | tuple[int] | bool]]

Gets the dictionary containing the class instance's available POD commands.

Returns

Dictionary containing the available commands and their information. Formatted as key(command number): value([command name, number of argument ASCII bytes, number of return bytes, binary flag])

Return type

dict[int, list[str|tuple[int]|bool]]

GetPODpacket(cmd: str | int, payload: int | bytes | tuple[int | bytes] | None = None) \rightarrow bytes

Builds a POD packet and writes it to a POD device via COM port. If an integer payload is give, the method will convert it into a bytes string of the length expected by the command. If a bytes payload is given, it must be the correct length.

Parameters

- cmd (str | int) Command number.
- payload (int | bytes | tuple[int | bytes], optional) None when there is no payload. If there is a payload, set to an integer value, bytes string, or tuple. Defaults to None.

Raises

- Exception POD command does not exist.
- **Exception** POD command requires a payload.

Returns

Bytes string of the POD packet.

Return type

bytes

static GetU(u: int) \rightarrow int

number of hexadecimal characters for an unsigned u-bit value.

Parameters

u (*int*) – 8, 16, or 32 bits. Enter any other number for NOVALUE.

Returns

number of hexadecimal characters for an unsigned u-bit value.

Return type

int

static PayloadToBytes(payload: $int \mid bytes \mid tuple[int \mid bytes]$, argSizes: tuple[int]) \rightarrow bytes Converts a payload into a bytes string (assuming that the payload is for a valid command).

Parameters

- payload (int | bytes | tuple[int | bytes]) Integer, bytes, or tuple containing the payload.
- argSizes (tuple[int]) Tuple of the argument sizes.

Returns

Bytes string of the payload.

Return type

bytes

ReadPODpacket($validateChecksum: bool = True, timeout_sec: int | float = 5) <math>\rightarrow Packet$

Reads a complete POD packet, either in standard or binary format, beginning with STX and ending with ETX. Reads first STX and then starts recursion.

Parameters

- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.
- timeout_sec (int/float, optional) Time in seconds to wait for serial data. Defaults to 5.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

SetBaudrateOfDevice(baudrate: int) \rightarrow bool

If the port is open, it will change the baud rate to the parameter's value.

Parameters

baudrate (*int*) – Baud rate to set for the open serial port.

Returns

True if successful at setting the baud rate, false otherwise.

Return type

bool

TestConnection($pingCmd: str \mid int = 'PING') \rightarrow bool$

Tests if a POD device can be read from or written. Sends a PING command.

Parameters

pingCmd (str | int, optional) - Command name or number to ping. Defaults to 'PING'.

Returns

True for successful connection, false otherwise.

Return type

bool

Raises

Exception – Ping command does not exist for this POD device.

WritePacket($cmd: str \mid int, payload: int \mid bytes \mid tuple[int \mid bytes] \mid None = None) <math>\rightarrow PacketStandard$ Builds a POD packet and writes it to the POD device.

Parameters

- cmd (str / int) Command number.
- payload (int | bytes | tuple[int | bytes], optional) None when there is no payload. If there is a payload, set to an integer value, bytes string, or tuple. Defaults to None.

Returns

Packet that was written to the POD device.

Return type

Packet Standard

WriteRead(cmd: $str \mid int$, payload: $int \mid bytes \mid tuple[int \mid bytes] \mid None = None$, validateChecksum: bool = True) $\rightarrow Packet$

Writes a command with optional payload to POD device, then reads (once) the device response.

Parameters

- cmd (str / int) Command number.
- payload (int | bytes | tuple[int|bytes], optional) None when there is no payload. If there is a payload, set to an integer value or a bytes string. Defaults to None.
- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

_ReadPODpacket_Recursive(validateChecksum: bool = True) $\rightarrow Packet$

Reads the command number. If the command number ends in ETX, the packet is returned. Next, it checks if the command is allowed. Then, it checks if the command is standard or binary and reads accordingly, then returns the packet.

Parameters

validateChecksum (*bool*, *optional*) – Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Raises

Exception – Cannot read an invalid command.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet | Packet _ Standard | Packet _ Binary Standard

_Read_Binary($prePacket: bytes, validateChecksum: bool = True) <math>\rightarrow PacketBinary$

Reads the remaining part of the variable-length binary packet. It first reads the standard packet (prePacket+payload+checksum+ETX). Then it determines how long the binary packet is from the payload of the standard POD packet and reads that many bytes. It then reads to ETX to get the checksum+ETX.

Parameters

- **prePacket** (*bytes*) Bytes string containing the beginning of a POD packet: STX (1 byte) + command number (4 bytes)
- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Raises

Exception – An exception is raised if the checksum is invalid (only if validateChecksum=True).

Returns

Variable-length binary POD packet.

Return type

Packet_BinaryStandard

_Read_GetCommand(validateChecksum: bool = True) \rightarrow bytes

Reads one byte at a time up to 4 bytes to get the ASCII-encoded bytes command number. For each byte read, it can (1) start the recursion over if an STX is found, (2) returns if ETX is found, or (3) continue building the command number.

Parameters

validateChecksum (*bool*, *optional*) – Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Returns

4 byte long string containing the ASCII-encoded command number.

Return type

bytes

_Read_Standard($prePacket: bytes, validateChecksum: bool = True) <math>\rightarrow PacketStandard$

Reads the payload, checksum, and ETX. Then it builds the complete standard POD packet in bytes.

Parameters

- **prePacket** (*bytes*) Bytes string containing the beginning of a POD packet: STX (1 byte) + command number (4 bytes).
- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Raises

Exception – An exception is raised if the checksum is invalid (only if validateChecksum=True).

Returns

Complete standard POD packet.

Return type

Packet_Standard

_Read_ToETX(validateChecksum: bool = True) \rightarrow bytes

Reads one byte at a time until an ETX is found. It will restart the recursive read if an STX is found anywhere.

Parameters

validateChecksum (*bool*, *optional*) – Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Returns

Bytes string ending with ETX.

Return type

bytes

static _ValidateChecksum(msg: bytes) \rightarrow bool

Validates the checksum of a given POD packet. The checksum is valid if the calculated checksum from the data matches the checksum written in the packet.

Parameters

msg (*bytes*) – Bytes message containing a POD packet: STX (1 bytes) + data (? bytes) + checksum (2 bytes) + ETX (1 byte).

Returns

True if the checksum is correct, false otherwise.

Return type

bool

Raises

Exception – msg does not begin with STX or end with ETX.

PodApi.Devices.PodDevice_8206HR module

 $\textbf{class} \ \ \textbf{PodApi.Device}. \ \textbf{PodDevice}_\textbf{8206HR}. \ \textbf{Pod8206HR} (port: \textit{str} \mid \textit{int}, \textit{preampGain: int}, \textit{baudrate: int} = 9600)$

Bases: Pod

POD 8206HR handles communication using an 8206HR POD device.

_preampGain

Instance-level integer (10 or 100) preamplifier gain.

Type

int

ReadPODpacket (validateChecksum: bool = True, timeout_sec: int | float = 5) \rightarrow Packet

Reads a complete POD packet, either in standard or binary format, beginning with STX and ending with ETX. Reads first STX and then starts recursion.

Parameters

- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.
- timeout_sec (int | float, optional) Time in seconds to wait for serial data. Defaults to 5.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

_Read_Binary($prePacket: bytes, validateChecksum: bool = True) <math>\rightarrow PacketBinary4$

After receiving the prePacket, it reads the 8 bytes(TTL+channels) and then reads to ETX (checksum+ETX).

Parameters

- **prePacket** (*bytes*) Bytes string containing the beginning of a POD packet: STX (1 byte) + command number (4 bytes).
- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Raises

Exception – Bad checksum for binary POD packet read.

Returns

Binary4 POD packet.

Return type

Packet_Binary4

$static _TranslateTTLbyte_ASCII(ttlByte: bytes) \rightarrow dict[str, int]$

Separates the bits of each TTL (0-3) from a ASCII encoded byte.

Parameters

ttlByte (*bytes*) – One byte string for the TTL (ASCII encoded).

Returns

Dictionary of the TTLs. Values are 1 when input, 0 when output.

Return type

dict[str,int]

PodApi.Devices.PodDevice_8229 module

class PodApi.Devices.PodDevice_8229.Pod8229(port: str | int, baudrate: int = 19200)

Bases: Pod

POD_8229 handles communication using an 8229 POD device.

static BuildSetDayScheduleArgument($day: str \mid int, hours: list \mid tuple[bool \mid int], speed: int \mid list \mid tuple[int]) <math>\rightarrow$ tuple[int]

Appends the day of the week code to the front of the encoded hourly schedule. this tuple is formatted to be used as the #141 'SET DAY SCHEDULE' argument.

Parameters

- **day** (*str* / *int*) Day of the week. Can be either the name of the day (i.e. Sunday, Monday, etc.) or the 0-6 day code (0 for Sunday increacing to 6 for Saturday).
- hours (list | tuple[bool | int]) Array of 24 items. The value is 1 for motor on and 0 for motor off.
- **speed** (*int* | 1*ist* | tuple[*int*]) Speed of the motor (0-100). This is an integer of all hours are the same speed. If there are multiple speeds, this should be an array of 24 items.

Returns

description

Return type

tuple[int]

static CodeDayOfWeek(day: str) \rightarrow int

Converts the day of the week to an integer code understandable by the POD device. The day is determined by the first 1-2 characters of the string, which supports multiple abbreviations for days of the week.

Parameters

day (str) – Day of the week.

Raises

Exception – Invalid day of the week.

Returns

Code for the day of the week. Values are 0-6, with 0 for Saturday, 1 for Monday, ..., and 6 for Saturday.

Return type

int

 $static\ CodeDaySchedule(hours:\ list\ |\ tuple[bool\ |\ int],\ speed:\ int\ |\ list\ |\ tuple[int]) \to list[int]$

Bitmasks the day schedule to encode the motor on/off status and the motor speed. Use this for getting the command #141 'SET DAY SCHEDULE' U8x24 argument component.

Parameters

- hours (list | tuple[bool | int]) Array of 24 items. The value is 1 for motor on and 0 for motor off.
- **speed** (*int* | 1*ist* | tuple[*int*]) Speed of the motor (0-100). This is an integer of all hours are the same speed. If there are multiple speeds, this should be an array of 24 items.

List of 24 integer items. The msb is the motor on/off flag and the remaining 7 bits are the speed.

Return type

list[int]

static DecodeDayAndSchedule(dayschedule: bytes)

static DecodeDayOfWeek(day: int) $\rightarrow str$

Converts the integer code for a day of the week to a human-readable string.

Parameters

day (int) – Day of the week code must be 0-6.

Returns

Day of the week ('Sunday', 'Monday', etc.).

Return type

str

static DecodeDaySchedule($schedule: bytes) \rightarrow dict[str, int | tuple[int]]$

Interprets the return bytes from the command #142 'GET DAY SCHEDULE'.

Parameters

schedule (*bytes*) – 24 byte long bitstring with one U8 per hour in a day.

Returns

Dictionary with 'Hour' as a tuple of 24 0/1 values (0 is motor off and 1 is motor on) and 'Speed' as the motor speed (0-100). If the motor speed is the same every hour, 'Speed' will be an integer; otherwise, 'Speed' will be a tuple of 24 items.

Return type

dict[str,int|tuple[int]]

$\textbf{static DecodeLCDSchedule}(\textit{schedule}: \textit{bytes}) \rightarrow \text{dict}[\textit{str}, \textit{str} \mid \textit{list}[\textit{int}]]$

Interprets the return bytes from the command #202 'LCD SET DAY SCHEDULE'.

Parameters

schedule (*bytes*) – 4 Byte long bitstring. Byte 3 is weekday, Byte 2 is hours 0-7, Byte 1 is hours 8-15, and byte 0 is hours 16-23.

Returns

Dictionary with Day as the day of the week, and Hours containing a list of 24 0/1 values (one for each hour). Each bit represents the motor state in that hour, 1 for on and 0 for off.

Return type

dict[str,int|list[int]]

static GetCurrentTime() → tuple[int]

Gets a tuple to use as the argument for command #140 SET TIME containing values for the current time.

Returns

Tuple of 7 integer values. The format is (Seconds, Minutes, Hours, Day, Month, Year (without century, so 23 for 2023), Weekday (0 for Sunday))

Return type

tuple[int]

ReadPODpacket(validateChecksum: bool = True, timeout_sec: int | float = 5) \rightarrow PacketStandard

Reads a complete POD packet, either in standard or binary format, beginning with STX and ending with ETX. Reads first STX and then starts recursion.

Parameters

- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.
- timeout_sec (int | float, optional) Time in seconds to wait for serial data. Defaults to 5.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

WritePacket($cmd: str \mid int, payload: int \mid bytes \mid tuple[int \mid bytes] \mid None = None) <math>\rightarrow$ bytes Builds a POD packet and writes it to the POD device.

Parameters

- cmd (str | int) Command number.
- payload (int | bytes | tuple[int | bytes], optional) None when there is no payload. If there is a payload, set to an integer value, bytes string, or tuple. Defaults to None.

Returns

Bytes string that was written to the POD device.

Return type

bytes

static $_$ CodeDecimalAsHex $(val: int) \rightarrow int$

Builds an integer that equals the val argument when converted into hexadecimal. All integers are converted to hexadecimal ASCII encoded bytes. Some commands (i.e. 8229 #140) need decimal ASCII encoded bytes; to do this, give the return value of _CodeDecimalAsHex() as the payload. Example: I want a number that is equal to 16 in hex. $1*16^1 + 6*16^0 = 22$. Calling _CodeDecimalAsHex(16) will return 22.

Parameters

val (*int*) – Unsigned integer number.

Returns

integer that equals the val argument when converted into hexadecimal.

Return type

int

static _Custom140SETTIME(payload: tuple[int]) → tuple[int]

Custom function to translate the payload for command #140 SET TIME.

Parameters

payload (tuple[int]) - Default translated payload.

Returns

Tuple of times.

Return type

tuple[int]

static _DecodeDecimalAsHex(val: int) \rightarrow int

Interprets an integer that was converted to a hexadecimal representation of a decimal value. In other words, this method reverses _CodeDecimalAsHex().

Parameters

val (*int*) – Unsigned integer that was converted to a hexadecimal representation of a decimal value.

Returns

Unsigned integer as a true decimal number.

Return type

int

static _Validate_Day($day: str \mid int$) \rightarrow int

Raises an exception if the day is incorrectly formatted. If the day is given as a string, it will be converted to its integer code.

Parameters

day (str | int) – String day of the week or its repsective integer code.

Raises

- **Exception** The day integer argument must be 0-6.
- **Exception** The day argument must be a str or int.

Returns

Integer code representing a day of the week.

Return type

int

static _Validate_Hours($hours: list \mid tuple[bool \mid int]$) $\rightarrow list[bool \mid int]$

Raises an exception if the hours is incorrectly formatted. Converts the hours into a list before returning it.

Parameters

 $\begin{tabular}{ll} \textbf{hours} (list \mid tuple [bool \mid int]) - Array with 24 items with values of 1/0 representing each hour \end{tabular}$

Raises

- **Exception** The hours argument must be a list or tuple.
- **Exception** The hours argument must have exactly 24 items.
- **Exception** The hours items must be 0 or 1.

Returns

List with 24 items for each hour. The values are 1/0.

Return type

list[bool|int]

static _Validate_Schedule($schedule: bytes, size: int) \rightarrow bytes$

Raises an exception if the schedule is incorrectly formatted

Parameters

- **schedule** (*bytes*) Bytes string containing the day schedule.
- size (int) Number of U8 bytes.

Raises

- **Exception** The schedule must be bytes.
- **Exception** The schedule is the incorrect size

Same as the schedule argument.

Return type

bytes

static _Validate_Speed($speed: int \mid list \mid tuple[int]$) \rightarrow list[int]

Raises an exception if the speed is incorrectly formatted. If an integer speed is given, it will convert it to a list.

Parameters

speed (int | list | tuple[int]) – Motor speed (0-100). This can either be an integer or a tuple/list of 24 speeds.

Raises

- **Exception** The speed argument must be an int, list, or tuple.
- **Exception** The speed must be between 0 and 100.
- **Exception** The speed argument must have exactly 24 items as a list/tuple.
- **Exception** The speed must be between 0 and 100 for every list/tuple item.

Returns

List containing 24 motor speeds.

Return type

list[int]

PodApi.Devices.PodDevice 8401HR module

Bases: Pod

POD_8401HR handles communication using an 8401-HR POD device.

_ssGain

Instance-level dictionary storing the second-stage gain for all four channels.

Type

dict[str,int|None]

_preampGain

Instance-level dictionary storing the pramplifier gain for all four channels.

Type

dict[str,int|None]

$static CalculateBiasDAC_GetDACValue(vout: int | float) \rightarrow int$

Calculates the DAC value given the output voltage. Used for 'GET/SET BIAS' commands.

Parameters

```
vout (int | float) – Output voltage (+/- 2.048 V).
```

Returns

Integer of the DAC value (16 bit 2's complement).

Return type

int

$static CalculateBiasDAC_GetVout(value: int) \rightarrow float$

Calculates the output voltage given the DAC value. Used for 'GET/SET BIAS' commands.

Parameters

value (*int*) – DAC value (16 bit 2's complement).

Returns

Float of the output bias voltage [V].

Return type

float

$\textbf{static DecodeChannelBitmask}(\textit{channels: bytes}) \rightarrow \textit{dict}[\textit{str}, \textit{int}]$

Converts the channel bitmask byte to a dictionary with each channel value. Use for 'GET INPUT GROUND' command payloads.

Parameters

channels (*bytes*) – U8 byte containing the channel configuration.

Returns

Dictionary with the channels as keys and values as the state. 0=Grounded and 1=Connected to Preamp.

Return type

dict[str,int]

static DecodeSSConfigBitmask(config: bytes)

Converts the SS configuration byte to a dictionary with the high-pass and gain. Use for 'GET SS CONFIG' command payloads.

Parameters

config (*bytes*) – U8 byte containing the SS configuration. Bit 0 = 0 for 0.5Hz Highpass, 1 for DC Highpass. Bit 1 = 0 for 5x gain, 1 for 1x gain.

static DecodeTTLByte(ttlByte: bytes) \rightarrow dict[str, int]

Converts the TTL bytes argument into a dictionary of integer TTL values.

Parameters

ttlByte (*bytes*) – U8 byte containing the TTL bitmask.

Returns

Dictinoary with TTL name keys and integer TTL values.

Return type

dict[str,int]

$\textbf{static DecodeTTLPayload}(\textit{payload: bytes}) \rightarrow tuple[\text{dict[str, int]}]$

Decodes a paylaod with the two TTL bytes.

Parameters

payload (*bytes*) – Bytes string of the POD packet payload.

Returns

Tuple with two TTL dictionaries.

Return type

tuple[dict[str, int]]

static GetChannelBitmask(a: bool, b: bool, c: bool, d: bool) \rightarrow int

Gets a bitmask, represented by an unsigned integer, used for 'SET INPUT GROUND' command.

Parameters

- **a** (boo1) State for channel A, 0=Grounded and 1=Connected to Preamp.
- **b** (*boo1*) State for channel B, 0=Grounded and 1=Connected to Preamp.
- **c** (*bool*) State for channel C, 0=Grounded and 1=Connected to Preamp.
- **d** (*bool*) State for channel D, 0=Grounded and 1=Connected to Preamp.

Returns

Integer representing a bitmask.

Return type

int

static GetChannelMapForPreampDevice(preampName: str) \rightarrow dict[str, str] | None

Get the channel mapping (channel labels for A,B,C,D) for a given device.

Parameters

preampName (str) – String for the device/sensor name.

Returns

Dictionary with keys A,B,C,D with values of the channel names. Returns None if the device name does not exist.

Return type

dict[str,str]|None

$static GetSSConfigBitmask(gain: int, highpass: float) \rightarrow int$

Gets a bitmask, represented by an unsigned integer, used for 'SET SS CONFIG' command.

Parameters

- **gain** (*int*) 1 for 1x gain. else for 5x gain.
- **highpass** (*float*) 0 for DC highpass, else for 0.5Hz highpass.

Returns

Integer representing a bitmask.

Return type

int

$\textbf{static GetSupportedPreampDevices()} \rightarrow list[str]$

Gets a list of device/sensor names used for channel mapping.

Returns

List of string names of all supported sensors.

Return type

list[str]

static GetTTLbitmask(ext0: bool = 0, ext1: bool = 0, ttl4: bool = 0, ttl3: bool = 0, ttl2: bool = 0, ttl1: bool = 0) \rightarrow int

Builds an integer, which represents a binary mask, that can be used for TTL command arguments.

Parameters

- ext0 (bool, optional) boolean bit for ext0. Defaults to 0.
- ext1 (bool, optional) boolean bit for ext1. Defaults to 0.

- ttl4 (bool, optional) boolean bit for ttl4. Defaults to 0.
- ttl3 (bool, optional) boolean bit for ttl3. Defaults to 0.
- ttl2 (bool, optional) boolean bit for ttl2. Defaults to 0.
- ttl1 (bool, optional) boolean bit for ttl1. Defaults to 0.

Integer number to be used as a bit mask.

Return type

int

static IsPreampDeviceSupported(name: str) \rightarrow bool

Checks if the argument exists in channel map for all preamp sensors.

Parameters

name (str) – name of the device

Returns

True if the name exists in __CHANNELMAPALL, false otherwise.

Return type

bool

ReadPODpacket (validateChecksum: bool = True, timeout_sec: int | float = 5) \rightarrow Packet

Reads a complete POD packet, either in standard or binary format, beginning with STX and ending with ETX. Reads first STX and then starts recursion.

Parameters

- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.
- **timeout_sec** (*int|float*, *optional*) Time in seconds to wait for serial data. Defaults to 5.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

WritePacket($cmd: str \mid int, payload: int \mid bytes \mid tuple[int \mid bytes] \mid None = None) <math>\rightarrow PacketStandard$ Builds a POD packet and writes it to the POD device.

Parameters

- cmd (str / int) Command number.
- payload (int | bytes | tuple[int | bytes], optional) None when there is no payload. If there is a payload, set to an integer value, bytes string, or tuple. Defaults to None.

Returns

Packet that was written to the POD device.

Return type

Packet_Standard

static _FixABCDtype(*info: tuple* | *list* | *dict, thisIs: str* = ") \rightarrow dict

Converts the info argument into a dictionary with A, B, C, and D as keys.

Parameters

- **info** (tuple | list | dict) Variable to be converted into a dictionary.
- **thisIs** (*str*, *optional*) Description of the info argument, which is used in Exception statements. Defaults to "."

Raises

- **Exception** The dictionary has improper keys; keys must be ['A','B','C','D'].
- **Exception** The argument must have only four values.
- **Exception** The argument must be a tuple, list, or dict.

Returns

The info argument converted to a dictionary with A, B, C, and D as keys.

Return type

dict

_Read_Binary(prePacket: bytes, validateChecksum: bool = True)

After receiving the prePacket, it reads the 23 bytes (binary data) and then reads to ETX.

Parameters

- **prePacket** (*bytes*) Bytes string containing the beginning of a POD packet: STX (1 byte) + command number (4 bytes).
- validateChecksum (bool, optional) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.

Raises

Exception – Bad checksum for binary POD packet read.

static _ValidatePreampGain(preampGain: dict) \rightarrow None

Checks that the preamplifier gain dictionary has proper values (10, 100, or None).

Parameters

preampGain (*dict*) – preamplifier gain dictionary.

Raises

Exception – EEG/EMG preampGain must be 10 or 100. For biosensors, the preampGain is None.

static _ValidateSsGain(ssgain: dict)

Checks that the second stage gain dictionary has proper values (1, 5, or None).

Parameters

ssgain (*dict*) – Second stage gain dictionary.

Raises

Exception – The ssGain must be 1 or 5; set ssGain to None if no-connect.

```
__CHANNELMAPALL: dict[str, dict[str, str]] = {'8406-2BIO': {'A': 'Bio1', 'B': 'Bio2', 'C': 'NC', 'D': 'NC'}, '8406-BIO': {'A': 'Bio', 'B': 'NC', 'C': 'NC', 'D': 'NC'}, '8406-EEG2BIO': {'A': 'Bio1', 'B': 'EEG1', 'C': 'EMG', 'D': 'Bio2'}, '8406-SE': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8406-SE3': {'A': 'Bio', 'B': 'EEG1', 'C': 'EEG3', 'D': 'EEG2'}, '8406-SE3IM': {'A': 'EMG', 'B': 'EEG1', 'C': 'EEG3', 'D': 'EEG2'}, '8406-SE4': {'A': 'EEG4', 'B': 'EEG1', 'C': 'EEG3', 'D': 'EEG2'}, '8406-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SE': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SE-2BIO': {'A': 'Bio1', 'B': 'Bio2', 'C': 'EMG', 'D': 'EEG2'}, '8407-SE3': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG3', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL': {'A': 'Bio', 'B': 'EEG1', 'C': 'EMG', 'D': 'EEG2'}, '8407-SL-2BIO': {'A': 'Bio', 'B': 'Bio2', 'C': 'EMG', 'D': 'EEG2'}}
```

Class-level dictionary containing the channel map for all preamplifier devices.

PodApi.Devices.PodDevice 8480SC module

```
class PodApi.Devices.PodDevice_8480SC.Pod8480SC(port: str | int, baudrate: int = 9600)
```

Bases: Pod

POD_8480SC handles communication using an 8480-SC POD device.

```
\verb+static DecodeStimulusConfigBits+(config: int) \rightarrow \operatorname{dict}
```

Converts an integer into 3 values, representing 3 individual bits of the Stimulus Config Bits.

Parameters

config (int) – an integer is passed in, and it represents the Config Flag byte.

Returns

Keys as the names of the bits, the values representing values at each bit.

Return type

dict

static DecodeSyncConfigBits(config: int) \rightarrow dict

Converts an integer into 3 values, representing 3 individual bits of the Sync Config Bits.

Parameters

config (*int*) – an integer is passed in, and it represents the Sync Config Flag byte.

Returns

Keys as the names of the bits, the values representing values at each bit.

Return type

dict

static DecodeTTlConfigBits(config: int) \rightarrow dict

Converts an interger into 3 values, representing 3 individual bits of the TTL Config Bits.

Parameters

config (int) – an integer is passed in, and it represents the TTL Setup Config Flag Byte.

Returns

Keys as the names of the bits, the values representing values at each bit.

Return type

dict

ReadPODpacket($validateChecksum: bool = True, timeout_sec: int | float = 5) <math>\rightarrow PacketStandard$

Reads a complete POD packet, either in standard or binary format, beginning with STX and ending with ETX. Reads first STX and then starts recursion.

Parameters

- **validateChecksum** (*bool*, *optional*) Set to True to validate the checksum. Set to False to skip validation. Defaults to True.
- timeout_sec (int | float, optional) Time in seconds to wait for serial data. Defaults to 5.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Return type

Packet

static StimulusConfigBits(optoElec: bool, monoBiphasic: bool, Simul: bool) → int

Incoming inputs are bitmasked into an integer value. This value is later given as part of a payload to command #102 'SET STIMULUS'.

Parameters

- **optoElec** (*bool*) Bit is Opto/Electrical.
- monoBiphasic (bool) Bit 1 is Monophasic/Biphasic.
- Simul (bool) Bit 2 is Simultaneous.

Returns

which represents the Config flag byte in the Stimulus Command. The return value is the seventh item in the payload for command 'SET STIMULUS'.

Return type

int

static SyncConfigBits(*sync_level: bool, sync_idle: bool, signal_trigger: bool*) → int

Incoming inputs are bitmasked into an integer value. This value is later given as the payload to command #127 'SET SYNC CONFIG'.

Parameters

- **sync_level** (*bool*) Bit 0 is Sync Level.
- **sync_idle** (*bool*) Bit 1 is Stimulus Triggering.
- **signal_trigger** (*bool*) Bit 2 is Signal/Trigger.

Returns

which represents the Sync Config Bits format value.

Return type

int

static TtlConfigBits($trigger: bool, stimtrig: bool, input_sync: bool) <math>\rightarrow$ int

Incoming inputs are bitmasked into an integer value. This value is later given as part of the payload to command #109 'SET TTL SETUP'. This commands accepts 3 items in the payload, and the return value of this function is given as the second item.

Parameters

• **trigger** (*boo1*) – Bit 0 is 0 for rising edge, 1 for falling edge.

- **stimtrig** (*bool*) Bit 1 is 0 for TTL event notifications, 1 for TTL inputs as triggers.
- **input_sync** (*boo1*) Bit 7 is 0 for normal TTL operation, 1 for TTL pin operates as a sync output.

which represents the TTL Config Bits Format value.

Return type

int

WritePacket(cmd: $str \mid int$, payload: $int \mid bytes \mid tuple[int \mid bytes] \mid None = None$) $\rightarrow PacketStandard$ Builds a POD packet and writes it to the POD device.

Parameters

- cmd (str | int) Command number.
- payload (int | bytes | tuple[int | bytes], optional) None when there is no payload. If there is a payload, set to an integer value, bytes string, or tuple. Defaults to None.

Returns

Packet that was written to the POD device.

Return type

Packet_Standard

static _Custom108GETTTLSETUP(payload: bytes) \rightarrow tuple[int | dict]

Custom function to translate the TTL setup for command #108 GET TTL SETUP.

Parameters

payload (*bytes*) – Bytes string of the POD packet payload.

Returns

Tuple of the TTL setup.

Return type

tuple[int|dict]

static _Custom109SETTTLSETUP(payload: bytes) \rightarrow tuple[int | dict]

Custom function to translate the TTL setup for command #109 SET TTL SETUP.

Parameters

payload (*bytes*) – Bytes string of the POD packet payload.

Returns

Tuple of the TTL setup.

Return type

tuple[int|dict]

 $\begin{array}{l} \textbf{static _CustomSTIMULUS}(\textit{payload: bytes, defaultPayload: tuple}) \rightarrow \textbf{tuple} \\ \textbf{_summary_} \end{array}$

Parameters

- **payload** (*bytes*) Bytes string of the POD packet payload.
- **defaultPayload** (*tuple*) Default translated payload.

Returns

Tuple of the translated stimulus payload.

Return type

tuple

static _CustomSYNCCONFIG(payload: bytes) \rightarrow dict

Custom function to translate the sync config.

Parameters

payload (*bytes*) – Bytes string of the POD packet payload.

Returns

Keys as the names of the bits, the values representing values at each bit.

Return type

dict

Module contents

1.1.1.3 PodApi.Packets package

Submodules

PodApi.Packets.Binary module

class PodApi.Packets.Binary.**PacketBinary**(pkt: bytes, commandS: CommandSet | None = None)

Bases: Packet

Container class that stores a standard binary command packet for a POD device. The format is STX (1 byte) + command number (4 bytes) + length of binary (4 bytes) + checksum (2 bytes) + ETX (1 bytes) + binary (LENGTH bytes) + checksum (2 bytes) + ETX (1 bytes)

binaryLength

Number of bytes of binary data from the packet.

Type

bytes

binaryData

Variable length binary datafrom the packet.

Type

bytes

$BinaryLength() \rightarrow int$

Translate the binary ASCII encoding of the binary data length into a readable integer

Returns

Integer of the binary data length.

Return type

int

static CheckIfPacketIsValid(msg: bytes)

Raises an Exception if the packet is incorrectly formatted.

Parameters

msg (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

- **Exception** Packet is too small to be a standard packet.
- **Exception** A standard binary packet must have an ETX before the binary bytes.

GetBinaryData() \rightarrow bytes

Gets the binary data from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet.

Returns

Bytes string containg binary data.

Return type

bytes

static GetBinaryLength(pkt: bytes) \rightarrow bytes

Gets the length, or number of bytes, of the binary data in a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet.

Returns

Bytes string of the length of the binary data.

Return type

bytes

static GetMinimumLength() \rightarrow int

Gets the number of bytes in the smallest possible packet; STX (1 byte) + something + ETX (1 byte).

Returns

integer representing the minimum length of a binary POD command packet. Format is STX (1 byte) + command number (4 bytes) + length of binary (4 bytes) + checksum (2 bytes) + ETX (1 bytes) + binary (LENGTH bytes) + checksum (2 bytes) + ETX (1 bytes)

Return type

int

$\textbf{TranslateAll()} \rightarrow dict[str, Any]$

Builds a dictionary containing all parts of the POD packet in readable values.

Returns

Dictionary with the command number, binary packet length, and binary data.

Return type

dict[str,Any]

UnpackAll() \rightarrow dict[str, bytes]

Builds a dictionary containing all parts of the POD packet in bytes.

Returns

Dictionary with the command number, binary packet length, and binary data.

Return type

dict[str,bytes]

PodApi.Packets.Binary4 module

```
class PodApi.Packets.Binary4.PacketBinary4(pkt: bytes, preampGain: int, commands: CommandSet |
                                                       None = None)
      Bases: Packet
      Container class that stores a binary4 command packet for a POD device. The format is STX (1 byte) + command
      (4 bytes) + packet number (1 byte) + TTL (1 byte) + CH0 (2 bytes) + CH1 (2 bytes) + CH2 (2 bytes) + checksum
      (2 \text{ bytes}) + \text{ETX } (1 \text{ byte}).
      _preampGain
           Preamplifier gain. This should be 10 or 100 for an 8206-HR device.
                    int
      packetNumber
           Packet number for this POD packet.
                Type
                    bytes
      ttl
           TTL data for this packet.
                Type
                    bytes
      ch0
           channel 0 data for this packet.
                Type
                    bytes
      ch1
           channel 1 data for this packet.
                Type
                    bytes
      ch2
           channel 2 data for this packet.
                Type
                    bytes
      static BinaryBytesToVoltage(value: bytes, preampGain: int) \rightarrow float
           Converts a binary bytes value read from POD device and converts it to the real voltage value at the pream-
           plifier input.
                Parameters
                    value (bytes) – Bytes string containing voltage measurement.
                Returns
                    A number containing the voltage in Volts [V].
                Return type
                    float
```

Ch $(n: int) \rightarrow float$

Translates the binary channel n bytes into a voltage.

Parameters

n (*int*) – Channel number. Should be 0, 1, or 2.

Raises

Exception – Channel does not exist.

Returns

Voltage of channel n in Volts.

Return type

float

static CheckIfPacketIsValid(msg: bytes)

Raises an Exception if the packet is incorrectly formatted.

Parameters

msg (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

Exception – Packet the wrong size to be a binary4 packet.

static GetBinaryLength() \rightarrow int

Gets the number of bytes of binary data in a binary4 packet.

Returns

Integer representing the number of binary encoded bytes in a binary4 packet.

Return type

int

static GetCh(n: int, pkt: bytes) \rightarrow bytes

Gets the channel n bytes from a POD packet.

Parameters

- **n** (*int*) Channel number. Should be 0, 1, or 2.
- **pkt** (*bytes*) Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the channel 0 data.

Return type

bytes

Returns

Channel does not exist.

Return type

bytes

static GetMinimumLength() \rightarrow int

Gets the number of bytes in the smallest possible binary4 packet; STX (1 byte) + command (4 bytes) + packet number (1 byte) + TTL (1 byte) + CH0 (2 bytes) + CH1 (2 bytes) + CH2 (2 bytes) + checksum (2 bytes) + ETX (1 byte).

Returns

Integer representing the minimum length of a binary4 POD packet.

Return type

int

static GetPacketNumber(pkt: bytes) \rightarrow bytes

Gets the packet number in bytes from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the packet number.

Return type

bytes

static GetTTL(pkt: bytes) \rightarrow bytes

Gets the TTL bytes from a POD packet

Parameters

pkt (bytes) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the TTL data.

Return type

bytes

$PacketNumber() \rightarrow int$

Translates the binary packet number into a readable integer.

Returns

Integer of the packet number.

Return type

int

$\textbf{TranslateAll()} \rightarrow dict[str, Any]$

Builds a dictionary containing all parts of the POD packet in readable values.

Returns

Dictionary with the command number, packet number, TTL and channels 0, 1, and 2.

Return type

dict[str, Any]

static TranslateBinaryTTLbyte(ttlByte: bytes) \rightarrow dict[str, int]

Separates the bits of each TTL (0-3) from a binary encoded byte.

Parameters

ttlByte (*bytes*) – One byte string for the TTL (binary encoded).

Returns

Dictionary of the TTLs. Values are 1 when input, 0 when output.

Return type

dict[str,int]

Ttl() \rightarrow dict[str, int]

Translates the binary TTL bytes into a dictionary containing each TTL value.

Returns

Dictionary with TTL name keys and TTL data as values.

Return type

dict[str,int]

UnpackAll() \rightarrow dict[str, bytes]

Builds a dictionary containing all parts of the POD packet in bytes.

Returns

Dictionary with the command number, packet number, TTL and channels 0, 1, and 2.

Return type

dict[str,bytes]

PodApi.Packets.Binary5 module

Bases: Packet

Container class that stores a binary5 command packet for a POD device. The format is STX (1 byte) + command (4 bytes) + packet number (1 byte) + status (1 byte) + channels (9 bytes) + AEXT0 (2 bytes) + AEXT1 (2 bytes) + ATTL1 (2 bytes) + ATTL2 (2 bytes) + ATTL2 (2 bytes) + ATTL4 (2 bytes) + checksum (2 bytes) + EXT (1 byte)

_ssGain

Dictionary with A, B, C, D keys and second stage gain values (1, 5, or None).

Type

dict[str,int|None]

_preampGain

Dictionary with A, B, C, D keys and preamplifier gain values (10, 100, or None).

Type

dict[str,int|None]

packetNumber

Packet number for this POD packet.

Type

bytes

status

Status for this POD packet.

Type

bytes

channels

channel A, B, C, and D data for this POD packet.

Type

bytes

```
aEXT0
     Analog EXT0 data for this POD packet.
         Type
             bytes
aEXT1
     Analog EXT1 data for this POD packet.
         Type
             bytes
aTTL1
     Analog TTL1 data for this POD packet.
         Type
             bytes
aTTL2
     Analog TTL2 data for this POD packet.
         Type
             bytes
aTTL3
     Analog TTL3 data for this POD packet.
         Type
             bytes
aTTL4
     Analog TTL4 data for this POD packet.
         Type
             bytes
AnalogEXT(n: int) \rightarrow float
     Translates the analog EXT value into a voltage.
         Parameters
             n (int) – Analog EXT number. Should be 0 or 1.
         Raises
             Exception – AEXT does not exist.
         Returns
             Analog EXT voltage in volts (V).
         Return type
             float
AnalogTTL(n: int) \rightarrow float
     Translates the analog TTL value into a voltage.
         Parameters
             n (int) – Analog TTL number. Should be 1, 2, 3, or 4.
             Exception – ATTL does not exist.
         Returns
             Analog TTL voltage in volts (v).
```

float

Channel (c: str) \rightarrow float

Translates the channel data into a voltage.

Parameters

c (str) – Channel character. Should be A, B, C, or D.

Raises

Exception – Channel does not exist.

Returns

Voltage of the channel in volts (V).

Return type

float

static CheckIfPacketIsValid(msg: bytes)

Raises an Exception if the packet is incorrectly formatted.

Parameters

msg (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

Exception – Packet the wrong size to be a binary5 packet.

static GetAnalogEXT(n: int, pkt: bytes) \rightarrow bytes

Gets the analog EXT from a POD packet.

Parameters

- **n** (*int*) Analog EXT number. Should be 0 or 1.
- **pkt** (*bytes*) Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

Exception – AEXT does not exist.

Returns

Bytes string of the AEXT.

Return type

bytes

static GetAnalogTTL(n: int, pkt: bytes) \rightarrow bytes

Gets the analog TTL from a POD packet.

Parameters

- n (int) Analog TTL number. Should be 1, 2, 3, or 4.
- pkt (bytes) Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

Exception – ATTL does not exist.

Returns

Bytes string of the ATTL.

bytes

static GetBinaryLength() \rightarrow int

Gets the number of bytes of binary data in a binary 5 packet.

Returns

Integer representing the number of binary encoded bytes in a binary5 packet.

Return type

int

static GetChannels(pkt: bytes) \rightarrow bytes

Gets the channel bytes for channels A, B, C, and D together from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the channels A, B, C, and D together.

Return type

bytes

$static GetMinimumLength() \rightarrow int$

Gets the number of bytes in the smallest possible binary4 packet; STX (1 byte) + command (4 bytes) + packet number (1 byte) + status (1 byte) + channels (9 bytes) + AEXTO (2 bytes) + AEXTI (2 bytes) + ATTL1 (2 bytes) + EXT (1 byte)

Returns

Integer representing the minimum length of a binary5 POD packet.

Return type

int

static GetPacketNumber(pkt: bytes) \rightarrow bytes

Gets the packet number in bytes from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the packet number.

Return type

bytes

static GetStatus(pkt: bytes) \rightarrow bytes

Gets the status value in bytes from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the status.

Return type

bytes

$PacketNumber() \rightarrow int$

Translates the binary packet number into a readable integer.

Returns

Integer of the packet number.

Return type

int

Status() \rightarrow int

Translates the binary status value into a readable integer

Returns

Integer status value.

Return type

int

TranslateAll() \rightarrow dict[str, Any]

Builds a dictionary containing all parts of the POD packet in readable values.

Returns

Dictionary with the command number, packet number, status, channels, analog EXT, and analog TTL.

Return type

dict[str, bytes]

UnpackAll() \rightarrow dict[str, bytes]

Builds a dictionary containing all parts of the POD packet in bytes.

Returns

Dictionary with the command number, packet number, status, channels, analog EXT, and analog TTL.

Return type

dict[str, bytes]

static _Voltage_PrimaryChannels($value: int, ssGain: int \mid None = None, PreampGain: int \mid None = None) \rightarrow float$

Converts a value to a voltage for a primary channel.

Parameters

- **value** (*int*) Value to be converted to voltage.
- **ssGain** (int | None, optional) Second stage gain. Defaults to None.
- **PreampGain** (int | None, optional) Preamplifier gain. Defaults to None.

Returns

Number of the voltage in volts [V]. Returns value if no gain is given (no-connect).

Return type

float

$static _Voltage_PrimaryChannels_Biosensor(value: int, ssGain: int) \rightarrow float$

Converts a value to a voltage for a biosensor primary channel.

Parameters

• **value** (*int*) – Value to be converted to voltage.

• ssGain (int) – Second stage gain.

Returns

Number of the voltage in volts [V].

Return type

float

 $\textbf{static _Voltage_PrimaryChannels_EEGEMG}(\textit{value: int, ssGain: int, PreampGain: int)} \rightarrow \text{float}$

Converts a value to a voltage for an EEG/EMG primary channel.

Parameters

- **value** (*int*) Value to be converted to voltage.
- **ssGain** (*int*) Second stage gain.
- **PreampGain** (int) Preamplifier gain.

Returns

Number of the voltage in volts [V].

Return type

float

$\textbf{static _Voltage_SecondaryChannels}(\textit{value: int}) \rightarrow \text{float}$

Converts a value to a voltage for a secondary channel.

Parameters

value (*int*) – Value to be converted to voltage.

Returns

Number of the voltage in volts [V].

Return type

float

PodApi.Packets.Packet module

```
class PodApi.Packets.Packet(pkt: bytes, commands: CommandSet | None = None)
```

Bases: object

Container class that stores a command packet for a POD device. The format is STX (1 byte) + command number (4 bytes) + data (? bytes) + ETX (1 byte). This class also has a collection of methods for creating and interpreting POD packets.

_commands

Available commands for a POD device.

Type

POD_Commands | None

rawPacket

Bytes string containing a POD packet. Should begin with STX and end with ETX.

Type

bytes

commandNumber

Command number from the Pod packet.

Type

bytes | None

static ASCIIbytesToInt_Split($msg: bytes, keepTopBits: int, cutBottomBits: int) <math>\rightarrow$ int

Converts a specific bit range in an ASCII-encoded bytes object to an integer.

Parameters

- msg (bytes) Bytes message holding binary information to be converted into an integer.
- **keepTopBits** (*int*) Integer position of the msb of desired bit range.
- **cutBottomBits** (*int*) Integer number of lsb to remove.

Returns

Integer result from the ASCII-encoded bytes message in a given bit range.

Return type

int

static AsciiBytesToInt(msg_b : bytes, signed: bool = False) \rightarrow int

Converts a ASCII-encoded bytes message into an integer. It does this using a base-16 conversion. If the message is signed and the msb is '1', the integer will be converted to it's negative 2's complement.

Parameters

- msg_b (bytes) Bytes message to be converted to an integer. The bytes must be base-16 or the conversion will fail.
- **signed** (*bool*, *optional*) True if the message is signed, false if unsigned. Defaults to False.

Returns

Integer result from the ASCII-encoded byte conversion.

Return type

int

 $\textbf{static BinaryBytesToInt}(\textit{msg: bytes, byteorder: str} = \textit{'big', signed: bool} = \textit{False}) \rightarrow \text{int}$

Converts binary-encoded bytes into an integer.

Parameters

- msg (bytes) Bytes message holding binary information to be converted into an integer.
- **byteorder** (*str*, *optional*) Ordering of bytes. 'big' for big endian and 'little' for little endian. Defaults to 'big'.
- **signed** (*bool*, *optional*) Boolean flag to mark if the msg is signed (True) or unsigned (False). Defaults to False.

Returns

Integer result from the binary-encoded bytes message.

Return type

int

static BinaryBytesToInt_Split(msg: bytes, keepTopBits: int, cutBottomBits: int, byteorder: <math>str = 'big', signed: bool = False) \rightarrow int

Converts a specific bit range in a binary-encoded bytes object to an integer.

Parameters

- msg (bytes) Bytes message holding binary information to be converted into an integer.
- **keepTopBits** (*int*) Integer position of the msb of desired bit range.
- **cutBottomBits** (*int*) Integer number of lsb to remove.
- **byteorder** (*str*, *optional*) Ordering of bytes. 'big' for big endian and 'little' for little endian. Defaults to 'big'.
- **signed** (*bool*, *optional*) Boolean flag to mark if the msg is signed (True) or unsigned (False). Defaults to False.

Returns

Integer result from the binary-encoded bytes message in a given bit range.

Return type

int

static CheckIfPacketIsValid(msg: bytes)

Raises an Exception if the packet is incorrectly formatted.

Parameters

msg (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

- **Exception** Packet must begin with STX.
- $\bullet \ \textbf{Exception} Packet \ must \ end \ in \ ETX \\$

CommandNumber() \rightarrow int

Translate the binary ASCII encoding into a readable integer

Returns

Integer of the command number.

Return type

int

static ETX() \rightarrow bytes

Get end-of-transmission (ETX) character in bytes. ETX marks the end byte of a POD Packet.

Returns

Bytes for ETX(0x03).

Return type

bytes

static GetCommandNumber(pkt: bytes) \rightarrow bytes | None

Gets the command number bytes from a POD packet.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Returns

Bytes string of the command number, if available.

Return type

bytes|None

static GetMinimumLength() \rightarrow int

Gets the number of bytes in the smallest possible packet; STX (1 byte) + something + ETX (1 byte).

Returns

integer representing the minimum length of a generic bytes string.

Return type

int

$HasCommandNumber() \rightarrow bool$

Checks if the packet has a command number.

Returns

True if the packet has a command number, False otherwise.

Return type

bool

$HasCommands() \rightarrow bool$

Checks if the Packet instance has commands set.

Returns

True if the commands have been set, false otherwise.

Return type

bool

$\textbf{static IntToAsciiBytes}(\textit{value: int, numChars: int}) \rightarrow \textit{bytes}$

Converts an integer value into ASCII-encoded bytes.

First, it converts the integer value into a usable uppercase hexadecimal string. Then it converts the ASCII code for each character into bytes. Lastly, it ensures that the final message is the desired length.

Example: if value=2 and numBytes=4, the returned ASCII will show b'0002', which is '0x30 0x30 0x30 0x32' in bytes. Uses the 2's complement if the val is negative.

Parameters

- value (int) Integer value to be converted into ASCII-encoded bytes.
- **numChars** (*int*) Number characters to be the length of the ASCII-encoded message.

Returns

Bytes that are ASCII-encoded conversions of the value parameter.

Return type

bytes

static STX() \rightarrow bytes

Get start-of-transmission (STX) character in bytes. STX marks the starting byte of a POD Packet.

Returns

Bytes for STX (0x02).

Return type

bytes

TranslateAll() \rightarrow dict[str, Any]

Builds a dictionary containing all parts of the POD packet in readable values.

Raises

Exception – Nothing to translate.

Returns

Dictionary with the command number.

Return type

dict[str,Any]

$static TwosComplement(val: int, nbits: int) \rightarrow int$

Gets the 2's complement of the argument value.

Parameters

- **val** (*int*) Value to be complemented.
- **nbits** (*int*) Number of bits in the value.

Returns

Integer of the 2's complement for the val.

Return type

int

UnpackAll() \rightarrow dict[str, bytes]

Builds a dictionary containing all parts of the POD packet in bytes.

Raises

Exception – Nothing to unpack.

Returns

Dictionary with the command number.

Return type

dict[str,bytes]

PodApi.Packets.Standard module

class PodApi.Packets.Standard.PacketStandard(pkt: bytes, commands: CommandSet)

Bases: Packet

Container class that stores a standard command packet for a POD device. The format is STX (1 byte) + command number (4 bytes) + optional payload (? bytes) + checksum (2 bytes) + ETX (1 bytes)

_customPayload

Optional function to translate the payload.

Type

Callable[[Any],tuple]|None

_customPayloadArgs

Optional arguments for the _customPayload.

Type

tuple[Any]|None

payload

Optional payload from the packet.

Type

bytes

static CheckIfPacketIsValid(msg: bytes)

Raises an Exception if the packet is incorrectly formatted.

Parameters

msg (*bytes*) – Bytes string containing a POD packet. Should begin with STX and end with ETX.

Raises

Exception – Packet is too small to be a standard packet.

DefaultPayload() \rightarrow tuple[int]

Splits the payload up into its components and translates the binary ASCII encoding into a readable integer.

Returns

Tuple with integer values for each component of the payload.

Return type

tuple[int]

$static GetMinimumLength() \rightarrow int$

Gets the number of bytes in the smallest possible packet.

Returns

integer representing the minimum length of a standard POD command packet. Format is STX (1 byte) + command number (4 bytes) + optional packet (? bytes) + checksum (2 bytes) + ETX (1 bytes)

Return type

int

static GetPayload(pkt: bytes) \rightarrow bytes | None

Gets the payload from a POD packet, if available.

Parameters

pkt (*bytes*) – Bytes string containing a POD packet.

Returns

Bytes string of the payload, if available.

Return type

bytes|None

${\tt HasCustomPayload()} \rightarrow bool$

Checks if a custom payload has been set.

Returns

True if there is a custom payload, False otherwise.

Return type

bool

$\textbf{HasPayload()} \rightarrow bool$

Checks if this Packet_Standard instance has a payload.

Returns

True if there is a payload, false otherwise.

Return type

bool

```
Payload() \rightarrow tuple | None
```

Gets the payload as a readable tuple of values.

Returns

Translated payload, if available.

Return type

tuple|None

SetCustomPayload(*func: Callable*[[Any], tuple], args: tuple[Any] | None = None) \rightarrow None Sets a custom function with optional arguments to translate the payload.

Parameters

- **func** (Callable[[Any], tuple]) Function to translate the payload.
- args (tuple[Any], optional) Arguments . Defaults to None.

TranslateAll() \rightarrow dict[str, Any]

Builds a dictionary containing all parts of the POD packet in readable values.

Returns

Dictionary with the command number and payload.

Return type

dict[str,Any]

UnpackAll() \rightarrow dict[str, bytes]

Builds a dictionary containing all parts of the POD packet in bytes.

Returns

Dictionary with the command number and payload.

Return type

dict[str,bytes]

Module contents

1.1.1.4 PodApi.Parameters package

Submodules

PodApi.Parameters.Params8206HR module

Bases: Params

Container class that stores parameters for an 8206-HR POD device.

port

Name of the COM port.

Type

str

sampleRate

Sample rate in 100-2000 Hz range.

Type

int

preamplifierGain

Preamplifier gain. Should be 10x or 100x.

Type

int

lowPass

Low-pass for EEG/EMG in 11-500 Hz range.

Type

tuple[int]

EEG1() \rightarrow int

Gets the filter value of EEG1 in Hz from the low-pass.

Returns

EEG1 low-pass filter in Hz.

Return type

int

EEG2() \rightarrow int

Gets the filter value of EEG2 in Hz from the low-pass.

Returns

EEG2 low-pass filter in Hz.

Return type

int

EEG3_EMG() \rightarrow int

Gets the filter value of EEG3/EMG in Hz from the low-pass.

Returns

EEG3/EMG low-pass filter in Hz.

Return type

int

$GetInit() \rightarrow str$

Builds a string that represents the Params_8206HR constructor with the arguments set to the values of this class instance.

Returns

String that represents the Params_8206HR constructor.

Return type

str

$_$ CheckParams() \rightarrow None

Throws an exception if Params_8206HR instance variable is an invalid value.

Raises

- **Exception** Sample rate must be between 100-2000 Hz.
- **Exception** Preamplidier gain must be 10x or 100x.

```
• Exception – Low-pass EEG/EMG must be between 11-500 Hz.
```

```
lowPassLabels: tuple[str] = ('EEG1', 'EEG2', 'EEG3/EMG')
```

Tuple describing the items in the lowPass.

PodApi.Parameters.Params8229 module

Bases: Params

Container class that stores parameters for an 8229 POD device.

port

Name of the COM port.

Type

str

systemID

ID of this 8229 POD system. Must be a positive integer.

Type

int

motorDirection

False for clockwise and true for counterclockwise.

Type

bool

motorSpeed

Motor speed as a percentage 0-100%.

Type

int

randomReverse

True to enable random reverse, False otherwise. The random reverse time will be reverseBaseTime + random value in reverseVarTime range.

```
Type
```

bool

reverseBaseTime

Base time for a random reverse in seconds. Must be a positive integer.

Type

int

reverseVarTime

Variable time for a random reverse in seconds. Must be a positive integer.

Type

int

mode

System mode; 0 = Manual, 1 = PC Control, and 2 = Internal Schedule.

Type int

schedule

Schedule for a week. The keys are the weekdays (Sunday-Saturday). The values are a tuple of 24 bools that are either 1 for motor on or 0 for motor off

Type

dict[str, tuple[int]]

static BuildEmptySchedule() → dict[str, tuple[bool]]

Creates a schedule where the motor is off for all hours of every day.

Returns

Dictionary of the empty schedule. The keys are the days of the week. The values are tuples of 24 zeros.

Return type

dict[str, tuple[bool]]

$GetInit() \rightarrow str$

Builds a string that represents the Params_8229 constructor with the arguments set to the values of this class instance.

Returns

String that represents the Params_Interface constructor.

Return type

str

_CheckParams() \rightarrow None

Throws an exception if Params_8229 instance variable is an invalid value.

Raises

- **Exception** The system ID must be a positive integer.
- **Exception** The motor speed must be between 0-100%.
- **Exception** The reverse base time (sec) must be a positive integer.
- **Exception** The reverse variable time (sec) must be a positive integer.
- **Exception** The mode must be 0, 1, or 2.
- **Exception** The schedule must have exactly ('Sunday','Monday','Tuesday', 'Wednesday','Thursday','Friday','Saturday') as keys.
- **Exception** There must be 24 items in the schedule for each day.

hoursPerDay: int = 24

Integer storing the number of hours in a day.

```
week: tuple[str] = ('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday')
```

Tuple containing strings of the 7 days of the week.

PodApi.Parameters.Params8401HR module

```
class PodApi.Parameters.Params8401HR.Params8401HR(port: str, preampDevice: str, sampleRate: int,
                                                              muxMode: bool, preampGain: tuple[int], ssGain:
                                                              tuple[int], highPass: tuple[float], lowPass:
                                                              tuple[int], bias: tuple[float], dcMode: tuple[str],
                                                              checkForValidParams: bool = True)
     Bases: Params
     Container class that stores parameters for an 8401-HR POD device.
     port
           Name of the COM port.
               Type
                   str
     preampDevice
           Name of the mouse/rat preamplifier device.
               Type
                   str
     sampleRate
           Sample rate (2000-20000 Hz).
               Type
                   int
     muxMode
           Using mux mode when True, false otherwise.
               Type
                   bool
     preampGain
           Preamplifier gain (1, 10, or 100) for all channels.
               Type
                   tuple[int]
     ssGain
           Second stage gain (1 or 5) for all channels.
               Type
                   tuple[int]
     highPass
           High-pass filter (0, 0.5, 1, or 10 Hz) for all channels.
               Type
                   tuple[float]
     lowPass
           Low-pass filter (21-15000 Hz) for all channels.
               Type
                   tuple[int]
```

bias

Bias voltage (+/- 2.048 V) for all channels.

Type

tuple[float]

dcMode

DC mode (VBIAS or AGND) for all channels.

Type

tuple[str]

$GetInit() \rightarrow str$

Builds a string that represents the Params_8401HR constructor with the arguments set to the values of this class instance.

Returns

String that represents the Params_8401HR constructor.

Return type

str

_CheckParams() \rightarrow None

Throws an exception if Params_8401HR instance variable is an invalid value.

Raises

- **Exception** Mouse/rat preamplifier does not exist.
- **Exception** Sample rate must be between 2000-20000 Hz.
- **Exception** EEG/EMG preamplifier gain must be 10x or 100x. For biosensors, the preampGain is None.
- **Exception** The second stage gain must be 1x or 5x; set gain to None if no-connect.
- **Exception** The high-pass filter must be 0.5, 1, or 10 Hz. If the channel is DC, input 0.
- Exception The low-pass filter must be between 21-15000 Hz.
- Exception The bias voltage must be +/- 2.048 V.
- Exception The DC mode must be VBIAS or AGND.

```
channelLabels: tuple[str] = ('A', 'B', 'C', 'D')
```

Tuple listing the four channel characters in order.

PodApi.Parameters.Params8480SC module

Bases: Params

Container class that stores parameters for an 8401-HR POD device.

```
port
     Name of the COM port.
          Type
              str
stimulus
     Stimulus configuration on selected channel.
          Type
              tuple[int]
preamp
     Preamp value (0-1023).
          Type
              int
ledCurrent
     Led-Current (0-600 mA) for both channels.
          Type
              tuple[int]
ttlPUllups
     TTL Pullups disabled for value 0, pullups enabled for values that are non-zero.
          Type
              int
estimCurrent
     Estim-Current (0-100 %) for both channels.
          Type
              tuple[int]
syncConfig
     Sets Sync-Config byte.
          Type
              int
ttlSetup
     TTL-Setup for selected channel.
          Type
              tuple[int]
GetInit() \rightarrow str
     Builds a string that represents the Params_8480SC constructor with the arguments set to the values of this
     class instance.
          Returns
              String that represents the Params_8480SC constructor.
          Return type
              str
_CheckParams() \rightarrow None
     Throws an exception if Params_8206HR member variable is an invalid value.
```

Raises

```
• Exception – The preamp must be between 0-1023.
```

- **Exception** Led-Curent must be between 0-600.
- **Exception** Estim-Current must be between 0-100.

$estimCurrent_CHO() \rightarrow int$

Gets the estimCurrent value for Channel 0.

Returns

Channel 0 estimCurrent in percentage.

Return type

int

$\textbf{estimCurrent_CH1()} \rightarrow \text{int}$

Gets the estimCurrent value for Channel 1.

Returns

Channel 1 estimCurrent in percentage.

Return type

int

$\textbf{ledCurrent_CHO()} \rightarrow \text{int}$

Gets the ledCurrent value for Channel 0.

Returns

Channel 0 ledCurrent in mA.

Return type

int

$ledCurrent_CH1() \rightarrow int$

Gets the ledCurrent value for Channel 1.

Returns

Channel 1 ledCurrent in mA.

Return type

int

PodApi.Parameters.ParamsBasic module

```
class PodApi.Parameters.ParamsBasic.Params(port: str, checkForValidParams: bool = True)
```

Bases: object

Interface for a container class that stores parameters for a POD device.

port

Name of the COM port.

Type

str

$GetInit() \rightarrow str$

Builds a string that represents the Params_Interface constructor with the arguments set to the values of this class instance.

Returns

String that represents the Params_Interface constructor.

str

_CheckParams() \rightarrow None

Throws an exception if Params_Interface instance variable is an invalid value.

Raises

Exception – The port name must begin with COM.

static _**FixTypeInTuple**(*arr: tuple, itemType: type*) → tuple['type']

Retypes each item of the arr arguemnt to itemType.

Parameters

- **arr** (*tuple*) Tuple of items to be re-typed.
- **itemType** (*type*) Type to be casted to each tuple item.

Returns

Tuple with values of all itemType types.

Return type

tuple[type]

Module contents

1.1.1.5 PodApi.Stream package

Subpackages

PodApi.Stream.Drain package

Subpackages

PodApi.Stream.Drain.PodHandler package

Submodules

PodApi.Stream.Drain.PodHandler.Handle8206HR module

class PodApi.Stream.Drain.PodHandler.Handle8206HR.Drain8206HR

Bases: DrainDeviceHandler

Class to help handle 8206-HR POD devices for the Drain classes.

DropToDf(timestamps: list[float], data: list[PodApi.Packets.Packet | None]) → DataFrame

Converts the timestamps and data into a Pandas DataFrame. The columns should match GetDeviceCol-Names().

Parameters

- **timestamps** (*list[float]*) List of timestamps in seconds for each data packet.
- data (list[Packet / None]) List of streaming binary data packets.

Returns

DataFrame containing the timestamps and packet data.

pd.DataFrame

DropToListOfArrays(*data: list*[PodApi.Packets.Packet | *float*]) → list[numpy.array]

Unpacks the data Packets into a list of np. arrays formatted to write to an EDF file.

Parameters

data (list[Packet | float]) – List of streaming binary data packets.

Returns

List of np.arrays for each Packet part.

Return type

list[np.array]

$GetDeviceColNames() \rightarrow str$

Gets a string of the column names formatter for a text file.

Returns

String of the filenames separated by commas and ending in a newline.

Return type

str

$GetDeviceColNamesList(includeTime: bool = True) \rightarrow list[str]$

Gets a list of all collumn titles.

Parameters

 $includeTime \ (bool, optional) - Flag to include 'Time' in the columns list. Defaults to True.$

Returns

List of columns.

Return type

list[str]

PodApi.Stream.Drain.PodHandler.Handle8401HR module

class PodApi.Stream.Drain.PodHandler.Handle8401HR.Drain8401HR(preampDevice: str | None = None)

Bases: DrainDeviceHandler

Class to help handle 8206-HR POD devices for the Drain classes.

preampDevice

Optional preamplifier for the 8401-HR.

Type

str|None

 $\textbf{DropToDf}(\textit{timestamps: list[float], data: list[PodApi.Packets.Packet | \textit{None]})} \rightarrow DataFrame$

Converts the timestamps and data into a Pandas DataFrame. The columns should match GetDeviceCol-Names().

Parameters

- **timestamps** (list[float]) List of timestamps in seconds for each data packet.
- data (list[Packet | None]) List of streaming binary data packets.

Returns

DataFrame containing the timestamps and packet data.

Return type

pd.DataFrame

DropToListOfArrays($data: list[PodApi.Packets.Packet.Packet | float]) <math>\rightarrow list[numpy.array]$

Unpacks the data Packets into a list of np.arrays formatted to write to an EDF file.

Parameters

data (list[Packet | float]) – List of streaming binary data packets.

Returns

List of np.arrays for each Packet part.

Return type

list[np.array]

${\tt GetDeviceColNames()} \rightarrow {\rm str}$

Gets a string of the column names formatter for a text file.

Returns

String of the filenames separated by commas and ending in a newline.

Return type

stı

GetDeviceColNamesList(includeTime: bool = True) \rightarrow list[str]

Gets a list of all collumn titles.

Parameters

 $includeTime\ (bool,\ optional)$ — Flag to include 'Time' in the columns list. Defaults to True.

Returns

List of columns.

Return type

list[str]

PodApi.Stream.Drain.PodHandler.HandlerInterface module

class PodApi.Stream.Drain.PodHandler.HandlerInterface.DrainDeviceHandler

Bases: object

Interface class for the POD device handlers used by the Drain classes.

DropToDf(timestamps: list[float], data: list[PodApi.Packets.Packet | None]) → DataFrame

Converts the timestamps and data into a Pandas DataFrame. The columns should match GetDeviceColNames().

Parameters

- **timestamps** (*list[float]*) List of timestamps in seconds for each data packet.
- data (list[Packet | None]) List of streaming binary data packets.

Returns

DataFrame containing the timestamps and packet data.

pd.DataFrame

DropToListOfArrays(*data: list*[PodApi.Packets.Packet | *float*]) → list[numpy.array]

Unpacks the data Packets into a list of np.arrays formatted to write to an EDF file.

Parameters

data (list[Packet | float]) - List of streaming binary data packets.

Returns

List of np.arrays for each Packet part.

Return type

list[np.array]

$\textbf{GetDeviceColNamesList}(includeTime:\ bool = True) \rightarrow list[str]$

Gets a list of all collumn titles.

Parameters

includeTime (bool, optional) - Flag to include 'Time' in the columns list. Defaults to True.

Returns

List of columns.

Return type

list[str]

static GetPodFromBucket(bkt: Bucket) $\rightarrow Pod8206HR \mid Pod8401HR$

Gets the POD device contained in the Bucket.

Parameters

bkt (Bucket) – Bucket to collect streaming data from a POD device.

Returns

POD device connected to the Bucket.

Return type

Pod8206HR | Pod8401HR

static $_uV(voltage: float | int) \rightarrow float$

Converts volts to microVolts, rounded to 6 decimal places.

Parameters

```
voltage (float | int) - number of volts.
```

Returns

voltage in of uV.

Return type

float

Module contents

Submodules

PodApi.Stream.Drain.ToEDF module

```
class PodApi.Stream.Drain.ToEDF.DrainToEDF (dataBucket: Bucket, fileName: str, preampDevice: str |
                                                      None = None
     Bases: DrainToFile
     Class to to drain, or save, the data drops collected by a Bucket into an EDF file.
     file
           Text file where streaming data is saved to.
               Type
                   EdfWriter|None
     CloseFile()
           Closes the file that data is saved to.
     static DigitalMax() → Literal[32767]
     static DigitalMin() → Literal[-32768]
     DrainDropToFile()
           Write one drop of data to the save file.
                   Exception – Can only save data to EDF if the hose is filtered with InsertValue, TakePast, or
                   TakeFuture.
     OpenFile()
           Opens and initializes a file using the fileName to save data to.
     static PhysicalBound() \rightarrow Literal[2046]
```

PodApi.Stream.Drain.ToFile module

```
class PodApi.Stream.Drain.ToFile.DrainToFile(dataBucket: Bucket, fileName: str, preampDevice: str \mid None = None)

Bases: object
```

Interface class to to drain, or save, the data drops collected by a Bucket into a file.

dataBucket

Bucket to collect streaming data.

```
Type
Bucket
```

fileName

Name (with optional file path) of the file to save data to.

```
Type str
```

preampDevice

Optional preamplifier for the 8401-HR.

Type

str | None

deviceHandler

Class to help handle different POD device types.

Type

Drain8206HR | Drain8401HR

CloseFile()

Closes the file that data is saved to.

DrainDropToFile()

Write one drop of data to the save file.

static GetExtension(fileName: str) \rightarrow str

Gets the extension from a file, such as '.txt', '.csv', or '.edf'.

Parameters

fileName (str) – Name of the file with an extension.

Returns

File extension.

Return type

st

static GetHandlerForBucket(bkt: Bucket, preampDevice: $str \mid None = None$) $\rightarrow Drain8206HR \mid Drain8401HR$

Selects the proper POD device handler for a given Bucket.

Parameters

- **bkt** (Bucket) Bucket to collect streaming data.
- **preampDevice** (str | None, optional) Optional preamplifier for the 8401-HR. Defaults to None.

Raises

Exception – POD Device is not supported.

Returns

POD Device Handler.

Return type

Drain8206HR | Drain8401HR

OpenFile()

Opens and initializes a file using the fileName to save data to.

PodApi.Stream.Drain.ToText module

```
class PodApi.Stream.Drain.ToText.DrainToTXT(dataBucket: Bucket, fileName: str, preampDevice: str |
                                                        None = None)
      Bases: DrainToFile
      Class to to drain, or save, the data drops collected by a Bucket into a text file.
      file
           Text file where streaming data is saved to.
                Type
                    TextIOWrapper|None
      CloseFile()
           Closes the file that data is saved to.
      DrainDropToFile()
           Write one drop of data to the save file.
      OpenFile()
           Opens and initializes a file using the fileName to save data to.
      static _GetTimeHeader() \rightarrow str
           Builds a string containing the current date and time to be written to the text file header.
                Returns
                    String containing the date and time. Each line begins with '#' and ends with a newline.
```

Module contents

Submodules

PodApi.Stream.DataBucket module

Return type str

totalDropsCollected

Counts the total number of Drops collected from the Hose.

Type

int

isCollecting

True when collecting drops from the Hose, False otherwise.

Type

bool

$DripDrop() \rightarrow tuple[list[float], list[PodApi.Packets.Packet.Packet \mid None]]$

Dequeues the first point (timestamp, data) in the drops queue.

Raises

Exception – No drops left to dequeue.

Returns

Tuple (x,y) with ~ 1 sec of timestamps (x) and data (y).

Return type

tuple[list[float], list[Packet|None]]

EmptyBucket()

Resets the class.

$\textbf{GetVolumeOfDrops()} \rightarrow int$

Get the number of data drops currently in the queue.

Returns

Size of the drops queue.

Return type

int

$\textbf{StartCollecting}(\textit{duration_sec: float} \mid \textit{None} = \textit{None}) \rightarrow \text{Thread}$

Start collecting stream data into the Bucket.

Parameters

duration_sec (float | None, optional) - How long to stream data in seconds. Defaults to None.

Returns

Started Thread for data collection.

Return type

Thread

StopCollecting()

Tells the POD device to stop streaming data.

_CollectDrop()

Adds a point to the drops queue. Each point is a tuple (x,y) of the timestamps list (x) and the data list (y) for one drop (values from ~1 sec of streaming, or the number of values approximatly equal to the sample rate).

_CollectForDuration(duration_sec: float)

Collect streaming data for a given duration.

Parameters

duration_sec (*float*) – How long to stream data in seconds.

_CollectWhileOpen()

Collect streaming data until the Hose is finished dripping.

$\verb|_IsDropAvailableInHose()| \rightarrow bool$

Checks if the Hose has any uncollected drops.

Returns

True if there is a drop to be collected, False otherwise.

Return type

bool

PodApi.Stream.DataHose module

Bases: object

Collects streaming data from an 8206-HR or 8401-HR POD device. The data and timestamps are updated about every 1 second when streaming.

sampleRate

Sample rate of the POD device.

Type int

deviceValve

To open or close the data stream.

Type

Valve

data

List of streaming binary data packets.

Type

list[*Packet*|None]

timestamps (list[float]

List of timestamps for each data packet.

numDrops

Number of drops, or number of times the data and timestamps have been updated.

Type int

corruptedPointsRemoved

Total number of corrupted data points removed from the data and timestamps lists.

Type int

filterMethod

Method used to filter out corrupted data.

Type str

filterInsert

Value to replace corrupted data with if using the 'InsertValue' filter method. Defaults to np.nan.

Type

float

EmptyHose()

```
static GetSampleRate(podDevice: Pod8206HR | Pod8401HR) \rightarrow int
```

Writes a command to the POD device to get its sample rate in Hz.

Parameters

podDevice (Pod8206HR / Pod8401HR) – POD device to get the sample rate for.

Raises

- **Exception** Cannot get the sample rate for this POD device.
- **Exception** Could not connect to this POD device.

Returns

Sample rate in Hz.

Return type

int

PickFilterMethod(filterMethod: str)

Set the method used to filter corrupted data when streaming. The filter methods include 'RemoveEntry', 'InsertValue', 'TakePast', 'TakeFuture', or 'DoNothing'. The default method is 'DoNothing'.

Parameters

filterMethod (*str*) – Filter method, which can be 'RemoveEntry', 'InsertValue', 'TakePast', 'TakeFuture', or 'DoNothing'/other.

SetFilterInsertValue(insert: float)

Sets the value to insert in place of currupted data. This is only used if the filter method is 'InsertValue'.

Parameters

insert (*float*) – Numerical value to insert.

StartStream()

Start a thread to start streaming data from the POD device.

StopStream()

Writes a command to the POD device to stop streaming data.

```
_Drop(currentTime: float, ti: float, data: list[PodApi.Packets.Packet | None]) → float
```

Updates the instance variables that store the streaming data. The data drops about every 1 second.

Parameters

- **currentTime** (*float*) Current start time in seconds.
- **ti** (*float*) Computer clock time at the start of the ~1 second drop.
- data (list[Packet | None]) Packets recieved when streaming.

Returns

updated current time in seconds for the next drop.

Return type

float

_Filter(*data: list*[PodApi.Packets.Packet | *None*], *timestamps: list*[*float*]) → bool Searches the data list for corrupted points, and deals with them according to the set filter method.

Parameters

- data (list[Packet / None]) List of Packets read from the POD device.
- **timestamps** (*list[float]*) List of timestamps in seconds for each Packet in the data list.

Returns

True when the corrupted data is filtered, False otherwise. A list containing only None cannot be filtered.

Return type

bool

_Filter_DoNothing(i: int, data: list[PodApi.Packets.Packet | None], timestamps: list[float])

Does nothing to the data and timestamps lists.

Parameters

- i (int) Index of corrupted data.
- data (list[Packet / None]) List of Packets read from the POD device.
- **timestamps** (*list[float]*) List of timestamps in seconds for each Packet in the data list.
- **_Filter_InsertValue**(*i: int, timestamps: list[float], data: list[*PodApi.Packets.Packet | *None]*) Replaces the data value at index i with a set value (class defaults to np.nan).

Parameters

- **i** (*int*) Index of corrupted data.
- data (list[Packet / None]) List of Packets read from the POD device.
- **timestamps** (*list[float]*) List of timestamps in seconds for each Packet in the data list.
- **_Filter_RemoveEntry**(*i: int, data: list*[PodApi.Packets.Packet.Packet | *None*], *timestamps: list*[*float*]) Removes a datapoint at index i from the data and timestamps lists.

Parameters

- **i** (*int*) Index of corrupted data.
- data (list[Packet | None]) List of Packets read from the POD device.
- **timestamps** (list[float]) List of timestamps in seconds for each Packet in the data list.
- _Filter_TakeFuture(i: int, data: list[PodApi.Packets.Packet | None], timestamps: list[float])

Replaces the data value at index i with the next Packet. If the index points to the last value in the data list, the data will be replaced with the previous Packet instead. If there are multiple currupted points in a row (data value is None), then all points will be replaced with the same future Packet.

Parameters

- i (int) Index of corrupted data.
- data (list[Packet / None]) List of Packets read from the POD device.

• timestamps (list[float]) - List of timestamps in seconds for each Packet in the data list.

_Filter_TakePast(i: int, data: list[PodApi.Packets.Packet | None], timestamps: list[float])

Replaces the data value at index i with the previous Packet. If the index points to the first value in the data list, the data will be replaced with the next Packet instead.

Parameters

- i (int) Index of corrupted data.
- data (list[Packet | None]) List of Packets read from the POD device.
- **timestamps** (*list[float]*) List of timestamps in seconds for each Packet in the data list.

_Flow()

Streams data from the POD device. The data drops about every 1 second. Streaming will continue until a "stop streaming" packet is recieved.

PodApi.Stream.DeviceValve module

```
class PodApi.Stream.DeviceValve.Valve(podDevice: Pod8206HR | Pod8401HR)
```

Bases: object

Simple class to start and stop streaming data from a POD device.

podDevice

POD device, such as an 8206-HR or 8401-HR.

Type

Pod

streamCmd

Command name/number for streaming data.

Type

str | int

streamPldStart

Payload to start streaming data.

Type

int | bytes | tuple[int | bytes]

streamPldStop

Payload to stop streaming data.

Type

int | bytes | tuple[int | bytes]

Close()

Write command to stop streaming

$Drip() \rightarrow Packet$

Reads one packet from the POD device.

Returns

POD packet beginning with STX and ending with ETX. This may be a standard packet, binary packet, or an unformatted packet (STX+something+ETX).

Packet

EmptyValve()

Reset the serial port buffer.

GetStartBytes() \rightarrow bytes

Gets the bytes string represeting a "start streaming data" packet.

Returns

Bytes string for a self.streamCmd command and a self.streamPldStart payload.

Return type

bytes

GetStopBytes() \rightarrow bytes

Gets the bytes string represeting a "stop streaming data" packet.

Returns

Bytes string for a self.streamCmd command and a self.streamPldStop payload.

Return type

bytes

Open()

Write command to start streaming.

Raises

Exception – Could not connect to this POD device.

PodApi.Stream.DrainToFile module

Bases: object

This class is used to save the streaming data from a POD device that was collected by a Bucket into a file.

self.drainToFile

Class to to drain the data drops collected by a Bucket into a text or EDF file.

Type

DrainToTXT, DrainToEDF

DrainBucketToFile($timeout_sec: float = 10.0, sleep: float = 0.25) \rightarrow Thread$

Starts a thread that starts draining data drops into the save file.

Parameters

- **timeout_sec** (*float*, *optional*) Quit saving data if no new data is recieved after timeout_sec time (in seconds). Defaults to 10.0.
- **sleep** (*float*, *optional*) Time duration in seconds to wait for the Bucket to collect more data before checking agian. Defaults to 0.25.

Returns

description

Return type

Thread

$IsCollecting() \rightarrow bool$

Checks if the Bucket is collected data from the Hose.

Returns

True if the Bucket is collecting data, False otherwise.

Return type

bool

IsDataAvailable() \rightarrow bool

Checks if data is available in the Bucket.

Returns

True if there are any drops in the Bucket, False otherwise.

Return type

bool

_DrainDropToFile($timeoutTicker: float, timeout_sec: float = 10.0, sleep: float = 0.25) <math>\rightarrow$ float _summary_

Parameters

- timeoutTicker (float) Counts the amout of time passed since the last drop was recieved.
- **timeout_sec** (*float*, *optional*) Quit saving data if no new data is recieved after timeout_sec time (in seconds). Defaults to 10.0.
- **sleep** (*float*, *optional*) Time duration in seconds to wait for the Bucket to collect more data before checking agian. Defaults to 0.25.

Raises

Exception – Timeout: no data found to be saved to file.

Returns

Updated timeoutTicker.

Return type

float

_ThreadedDrainBucketToFile(timeout_sec: float = 10.0, sleep: float = 0.25)

Opens a save file and starts saving data to it. The file is updated about every 1 sec. After the POD device stops streaming data, the file will close.

Parameters

- **timeout_sec** (*float*, *optional*) Quit saving data if no new data is recieved after timeout_sec time (in seconds). Defaults to 10.0.
- **sleep** (*float*, *optional*) Time duration in seconds to wait for the Bucket to collect more data before checking agian. Defaults to 0.25.

Module contents

1.1.2 Module contents

1.2 Setup package

1.2.1 Subpackages

1.2.1.1 Setup.Inputs package

Submodules

Setup.Inputs.GetUserInput module

class Setup.Inputs.GetUserInput.UserInput

Bases: object

UserInput contains several methods for getting user input for POD device setup.

static AskForBool(prompt: str) \rightarrow bool

Asks user for bool type input.

Parameters

prompt (*str*) – Statement requesting input from the user.

Returns

Boolean type input from user.

Return type

bool

$static AskForFloat(prompt: str) \rightarrow float$

Asks user for float type input.

Parameters

prompt (str) - Statement requesting input from the user.

Returns

Float type input from user.

Return type

float

static AskForFloatInList($prompt: str, goodInputs: list, badInputMessage: str | None = None) <math>\rightarrow$ float Asks the user for a float that exists in the list of valid options.

Parameters

- **prompt** (*str*) Statement requesting input from the user.
- **goodInputs** (*list*) List of valid input options.
- badInputMessage (str | None, optional) Error message to be printed if invalid input is given. Defaults to None.

Returns

User's choice from the options list as a float.

float

static AskForFloatInRange(prompt: str, minimum: float, maximum: float, thisIs: str = 'Input', unit: $str = '') \rightarrow float$

Asks the user for an float value that falls in a given range.

Parameters

- **prompt** (*str*) Statement requesting input from the user.
- **minimum** (*float*) Minimum value of range.
- maximum (float) Maximum value of range.
- **thisIs** (*str*, *optional*) Description of the input/what is being asked for. Used when printing the error message. Defaults to 'Input'.
- **unit** (*str*, *optional*) Unit of the requested value. Use when printing the error message. Defaults to ''.

Returns

Float value given by the user that falls in the given range.

Return type

float

static AskForInput(*prompt: str, append: str* = ': ') \rightarrow str

Asks user for input given a prompt. Will append a colon ':' to the end of prompt by default

Parameters

- **prompt** (str) Statement requesting input from the user
- append (str, optional) Appended to the end of the prompt. Defaults to ': '.

Returns

String of the user input

Return type

str

static AskForInt(prompt: str) \rightarrow int

Asks user for int type input.

Parameters

 ${\tt prompt}\ (str)$ — Statement requesting input from the user.

Returns

Integer type input from user.

Return type

int

static AskForIntInList($prompt: str, goodInputs: list, badInputMessage: str | None = None) <math>\rightarrow$ int Asks the user for an integer that exists in the list of valid options.

Parameters

- **prompt** (str) Statement requesting input from the user
- **goodInputs** (*1ist*) List of valid input options.
- badInputMessage (str | None, optional) Error message to be printed if invalid input is given. Defaults to None.

Returns

User's choice from the options list as an integer.

Return type

int

 $\textbf{static AskForIntInRange}(\textit{prompt: str, minimum: int, maximum: int, thisIs: str = 'Input', unit: str = '') \rightarrow int \\$

Asks the user for an integer value that falls in a given range.

Parameters

- **prompt** (*str*) Statement requesting input from the user.
- minimum (int) Minimum value of range.
- maximum (int) Maximum value of range.
- **thisIs** (*str*, *optional*) Description of the input/what is being asked for. Used when printing the error message. Defaults to 'Input'.
- **unit** (*str*, *optional*) Unit of the requested value. Use when printing the error message. Defaults to ".

Returns

Integer value given by the user that falls in the given range.

Return type

int

static AskForStrInList($prompt: str, goodInputs: list, badInputMessage: <math>str \mid None = None$) \rightarrow str Asks the user for a string that exists in the list of valid options.

Parameters

- **prompt** (*str*) Statement requesting input from the user.
- **goodInputs** (*list*) List of valid input options
- badInputMessage (str / None, optional) Error message to be printed if invalid input is given. Defaults to None.

Returns

User's choice from the options list as a string.

Return type

str

static AskForType(typecast: function, prompt: str) \rightarrow int | float | str

Ask user for input of a specific data type. If invalid input is given, an error message will print and the user will be prompted again.

Parameters

- **typecast** (*function*) Datatype to cast the user input (ex. _CastInt, _CastFloat, _CastStr)
- **prompt** (str) Statement requesting input from the user

Returns

Input from user as the requested type.

Return type

int|float|str

static AskForTypeInList($typecast: function, prompt: str, goodInputs: list, badInputMessage: str | None = None) <math>\rightarrow$ int | float | str

Asks the user for a value of a given type that exists in the list of valid options. If invalid input is given, an error message will print and the user will be prompted again.

Parameters

- **typecast** (*function*) Datatype to cast the user input (ex. _CastInt, _CastFloat, _CastStr).
- **prompt** (*str*) Statement requesting input from the user.
- **goodInputs** (*1ist*) List of valid input options.
- badInputMessage (str | None, optional) Error message to be printed if invalid input is given. Defaults to None.

Returns

User's choice from the goodInputs list as the given datatype.

Return type

int|float|str

static AskForTypeInRange(*typecast: function, prompt: str, minimum: int* | *float, maximum: int* | *float, thisIs: str* = 'Input', unit: str = '') \rightarrow int | float

Asks user for a numerical value that falls between two numbers. If invalid input is given, an error message will print and the user will be prompted again.

Parameters

- **typecast** (*function*) Datatype to cast the user input (ex. _CastInt, _CastFloat, _CastStr).
- prompt(str) Statement requesting input from the user.
- minimum (int | float) Minimum value of range.
- maximum (int | float) Maximum value of range.
- **thisIs** (*str*, *optional*) Description of the input/what is being asked for. Used when printing the error message. Defaults to 'Input'.
- **unit** (*str*, *optional*) Unit of the requested value. Use when printing the error message. Defaults to ".

Returns

Numerical value given by the user that falls in the given range.

Return type

int|float

static AskYN(question: str, append: str = '(y/n): ') \rightarrow bool

Asks the user a yes or no question. If invalid input is given, an error message will print and the user will be prompted again.

Parameters

- **question** (*str*) Statement requesting input from the user.
- append (str, optional) Appended to the end of the question. Defaults to '(y/n): '.

Returns

True for yes, false for no.

Return type

bool

static CastFloat(value) \rightarrow float

Casts the argument as an float.

Parameters

value – Value to type casted.

Returns

Value type casted as a float.

Return type

float

static CastInt(value) \rightarrow int

Casts the argument as an integer.

Parameters

value – Value to type casted.

Returns

Value type casted as an integer.

Return type

int

$static CastStr(value) \rightarrow str$

Casts the argument as an string.

Parameters

value – Value to type casted.

Returns

Value type casted as a string.

Return type

str

 $\textbf{static CheckFileExt}(\textit{f: str, flsExt: bool} = \textit{True}, \textit{goodExt: list[str]} = [\textit{'.txt'}], \textit{printErr: bool} = \textit{True}) \rightarrow \textit{bool}$

Checks if a file name has a valid extension.

Parameters

- $\mathbf{f}(str)$ file name or extension
- **fIsExt** (*bool*, *optional*) Boolean flag that is true if f is an extension, false otherwise. Defaults to True.
- **goodExt** (list[str], optional) List of valid file extensions. Defaults to ['.txt',].
- **printErr** (*bool*, *optional*) Boolean flag that, when true, will print an error statement. Defaults to True.

Returns

True if extension is in goodExt list, False otherwise.

Return type

bool

```
static GetFileName(goodExt: list[str] = ['.txt']) \rightarrow str
```

Asks the user for a filename. :param goodExt: List of valid file extensions. Defaults to ['.txt']. :type goodExt: list[str], optional

Returns

String of the file name and extension.

Return type

str

static GetFilePath($prompt: str \mid None = None, goodExt: list[str] = ['.txt']) \rightarrow str$

Asks the user for a file path and file name.

Parameters

- **prompt** (*str* / *None*, *optional*) Text to print to the user before requesting the path. Defaults to None.
- **goodExt** (list[str], optional) List of valid file extensions. Defaults to ['.txt'].

Returns

File path and name.

Return type

str

Module contents

1.2.1.2 Setup.SetupAllDevices package

Submodules

Setup.SetupAllDevices.Setup PodDevices module

```
 \textbf{class} \ \ \textbf{SetupAllDevices.SetupAll} (saveFileDict: \ dict[str, str] \mid None = None, \\ podParametersDict: \ dict[str, \ dict[int, \\ PodApi.Parameters.ParamsBasic.Params] \mid \\ None] \mid None = None)
```

Bases: object

Setup_PodDevices allows a user to set up and stream from any number of POD devices. The streamed data is saved to a file.

REQUIRES FIRMWARE 1.0.2 OR HIGHER.

_Setup_PodDevices

Dictionary containing the Setup_Interface subclasses for each POD device.

Type

dict[str,dict[int,Params_Interface]

_options

Dictionary listing the different options for the user to complete.

Type

dict[int,str]

GetOptions() \rightarrow dict[int, str]

Gets the dictionary of setup options.

Returns

Dictionary listing the different options for the user to complete (options).

Return type

dict[int,str]

$GetPODparametersInit() \rightarrow str$

Sets up each POD device type. Used in initialization. :returns: String representing a dictionary of all POD devices initialization. The keys are the device name and the entries are the initialization dictionaries. :rtype: str

${\tt GetSaveFileNames()} \rightarrow {\rm str}$

```
Run() \rightarrow None
```

Prints the options and askes the user what to do. Loops until 'Quit" is chosen.

SetupPODparameters (podParameters Dict: dict[str, dict[int, PodApi.Parameters.ParamsBasic.Params] | None]) \rightarrow None

Sets up each POD device type. Used in initialization.

Parameters

 $\label{lem:podParametersDict} \begin{subarray}{l} \textbf{Mone} \textbf{PodParametersDict} (dict[str,dict[int,Params_Interface] / None]) - Dictionary of all POD devices initialization. The keys are the device name and the entries are the initialization dictionaries. \\ \end{subarray}$

SetupSaveFile($saveFileDict: dict[str, str] \mid None = None) \rightarrow None$

Gets the path/file name from the user and stores it. Used in initialization.

Parameters

saveFile (dict[str,str|None] | None, optional) – String of the save file, which includes the directory path, filename, and file extension. Defaults to None.

_AskOption() \rightarrow int

Asks user which option to do.

Returns

Integer number representing an option key.

Return type

int

$\verb"_AskToStopStream"() \to None$

Asks user to press enter to stop streaming. The program will then prompt all POD devices to end stream.

static _AskUserForDevices()

Asks the user what POD devices they want to use.

_CheckForValidParams($podParametersDict: dict[str, None | dict]) <math>\rightarrow$ bool

Checks if the parameters are correctly formatted.

Parameters

podParametersDict (dict[str,None | dict]) – Dictionary with keys as the device names and values as None or the respective parameter dictionary.

Raises

• **Exception** – Parameters must be dictionary type.

- **Exception** Parameters dictionary is empty.
- **Exception** Invalid device name in paramater dictionary.

Returns

True if the parameters are correctly formatted.

Return type

bool

_ConnectNewDevice() \rightarrow None

Asks the user for the POD device type, then it sets up that device.

_DoOption(*choice: int*) \rightarrow None

Performs the methods associated with the user selected option.

Parameters

choice (int) – Integer number representing an option key.

$\texttt{_EditCheckConnect}() \rightarrow None$

Displays the POD devices parameters, asks the user to edit the device, and then reconnects the device for each POD device type.

_EditSaveFilePath() \rightarrow None

Asks the user for the POD device type, then asks the user for a new file name and path, then sets the value to the POD devices.

_GetChosenDeviceType(question: str) \rightarrow str

Asks the user which type of POD device they want.

Parameters

question (*str*) – Question to ask the user.

Returns

String of the user input (may be invalid POD device type).

Return type

str

_GetParams ($podParametersDict: None | dict[str, None]) <math>\rightarrow$ dict[str, dict | None]

If no parameters are give, this asks user which types of POD devices they want to use. Then it checks if the parameters are valid.

Parameters

podParametersDict (*None* / *dict[str,None]*) – Dictionary of all POD devices initialization. The keys are the device name and the entries are the initialization dictionaries.

Returns

Dictionary whose keys are the POD device name, and value the setup dictionary.

Return type

dict[str,dict|None]

_PrintInitCode() → None

Prints code that can be used to initialize and run SetupPodDevices with the current parameters.

_PrintOptions() \rightarrow None

Prints options available for user.

$_{\mathbf{Reconnect}}() \rightarrow bool$

Reconnects all POD devices.

Returns

Bool that is true if all devices were successfully connected. False otherwise.

Return type

bool

_RemoveDevice() \rightarrow None

Displays the POD devices parameters, asks the user which device ro remove, and then deletes that POD device.

$_$ Set $_$ Setup $_$ PodDevices $(podParametersDict: dict[str, dict | None]) <math>\rightarrow$ None

Sets the _Setup_PodDevices varaible to have keys as the POD device name and values as the setup class.

Parameters

podParametersDict (dict[str,dict | None]) – Dictionary with keys as the device names and values as None or the respective parameter dictionary.

$_$ ShowCurrentSettings() \rightarrow None

Displays the POD device settings for all devices, and then prints the save file name.

_Stream() \rightarrow float

Streams data from all POD devices and prints the execution time.

Returns

Float of the execution time in seconds.

Return type

float

$_{\mathtt{StreamAllDevices}}() \rightarrow \mathsf{None}$

Streams data from all the devices. User is asked to click enter to stop streaming. Data is saved to file. Uses threading.

static _TimeFunc(func: function) \rightarrow float

Runs a function and gets the calculated execultion time.

Parameters

func (function) – Function/method name.

Returns

Float of the execution time in seconds rounded to 3 decimal places.

Return type

float

Module contents

1.2.1.3 Setup.SetupOneDevice package

Submodules

Setup.SetupOneDevice.Setup 8206HR module

class Setup.SetupOneDevice.Setup_8206HR.Setup8206HR

Bases: SetupInterface

Setup 8206HR provides the setup functions for an 8206-HR POD device.

static GetDeviceName() \rightarrow str

Returns the name of the 8206-HR POD device.

Returns

8206-HR.

Return type

str

$StopStream() \rightarrow None$

Write a command to stop streaming data to all POD devices.

$static _ChoosePreampGain() \rightarrow int$

Asks user for the preamplifier gain of their POD device.

Returns

Integer 10 or 100 for the preamplifier gain.

Return type

int

_ConnectPODdevice($deviceNum: int, deviceParams: Params8206HR) <math>\rightarrow$ bool

Creates a POD_8206HR object and write the setup parameters to it.

Parameters

- **deviceNum** (*int*) Integer of the device's number.
- **deviceParams** (*Params_8206HR*) Device's parameters.

Returns

True if connection was successful, false otherwise.

Return type

bool

$_GetPODdeviceParameterTable() \rightarrow Texttable$

Builds a table containing the parameters for all POD devices.

Returns

Texttable containing all parameters.

Return type

Texttable

_GetParam_onePODdevice($forbiddenNames: list[str] = []) \rightarrow Params8206HR$

Asks the user to input all the device parameters.

Parameters

 $\textbf{forbiddenNames}\ (list[str]) - List\ of\ port\ names\ already\ used\ by\ other\ devices.$

Returns

Dictionary of device parameters.

Return type

dict[str,(str|int|dict[str,int])]

```
_OpenSaveFile_EDF(fname: str, devNum: int) \rightarrow EdfWriter
```

Opens EDF file and write header.

Parameters

- **fname** (*str*) String filename.
- **devNum** (*int*) Integer device number.

Returns

Opened file.

Return type

EdfWriter

_OpenSaveFile_TXT(fname: str) \rightarrow IOBase

Opens a text file and writes the column names. Writes the current date/time at the top of the txt file.

Parameters

fname (str) – String filename.

Returns

Opened file.

Return type

IOBase

_PHYSICAL_BOUND_uV: int = 2046

Class-level integer representing the max/-min physical value in uV. Used for EDF files.

_StreamThreading() \rightarrow dict[int, threading.Thread]

Opens a save file, then creates a thread for each device to stream and write data from.

Returns

Dictionary with keys as the device number and values as the started Thread.

Return type

dict[int,Thread]

_StreamUntilStop($pod: Pod8206HR, file: IOBase | EdfWriter, sampleRate: int) <math>\rightarrow$ None

Streams data from a POD device and saves data to file. Stops looking when a stop stream command is read. Calculates average time difference across multiple packets to collect a continuous time series data.

Parameters

- pod (POD_8206HR) POD device to read from.
- **file** (*IOBase* | *EdfWriter*) open file.
- sampleRate (int) Integer sample rate in Hz.

static _WriteDataToFile_EDF(file: EdfWriter, data: list[numpy.ndarray])

Writes data to an open EDF file.

Parameters

- **file** (*EdfWriter*) opened EDF file.
- data (list[np.ndarray]) List of 3 items, one for each channel.

static _WriteDataToFile_TXT(file: IOBase, data: list[numpy.ndarray], t: ndarray)

Writes data to an open text file.

Parameters

- **file** (*IOBase*) opened write file.
- data (list[np.ndarray]) List of 3 items, one for each channel.
- t (np.ndarray) list with the time stamps (in seconds).

Setup.SetupOneDevice.Setup_8229 module

class Setup.SetupOneDevice.Setup_8229.Setup8229

Bases: SetupInterface

Setup_8229 provides the setup functions for an 8229 POD device.

_streamMode

True when the user wants to stream data from an 8229 POD device, False otherwise.

Type

bool

static GetDeviceName() \rightarrow str

Returns the name of the POD device.

Returns

8229.

Return type

st

static GetSupportedFileExtensions() \rightarrow list[str]

Returns a list containing valid file extensions.

Returns

List of string file extensions.

Return type

list[str]

$StopStream() \rightarrow None$

Update the state flag to signal to stop streaming data.

_ConnectPODdevice($deviceNum: int, deviceParams: Params8229) <math>\rightarrow$ bool

Creates a 8229 POD device object and write the setup parameters to it.

Parameters

- **deviceNum** (*int*) Integer of the device's number.
- **deviceParams** (*Params_8229*) Device parameters.

Returns

True if connection was successful, false otherwise.

Return type

bool

_GetPODdeviceParameterTable() → Texttable

Builds a table containing the parameters for all POD devices.

Returns

Table containing all parameters.

Return type

Texttable

_GetParam_onePODdevice($forbiddenNames: list[str] = []) \rightarrow Params8229$

Asks the user to input all the device parameters.

Parameters

forbiddenNames (list[str], optional) – List of port names already used by other devices. Defaults to [].

Returns

Device parameters.

Return type

Params_8229

static _GetScheduleForWeek() → dict[str, tuple[int]]

Asks the user to input if the motor is on/off for each hour of each day of the week.

Returns

Dictionary with the schedule. The keys are the days of the week (Sunday, Monday, \dots). The values are a tuple of 24 items for each hour; the items are 1 if the motor is on or 0 if the motor is off

Return type

dict[str, tuple[int]]

_OpenSaveFile_EDF(fname: str, devNum: int)

EDF files are not supported for 8229 POD devices. Overwrites the parent's method, which would open an EDF file and write the header.

Parameters

- **fname** (*str*) String filename. Not used.
- **devNum** (*int*) Integer device number. Not used.

Raises

Exception – EDF filetype is not supported for 8229 POD devices.

_OpenSaveFile_TXT(fname: str) \rightarrow IOBase

Opens a text file and writes the column names. Writes the current date/time at the top of the txt file.

Parameters

fname (str) – String filename.

Returns

Opened file.

Return type

IOBase

$\verb|_StreamThreading()| \rightarrow dict[int, threading.Thread]|$

Opens a save file, then creates a thread for each device to stream and write data from.

Returns

Dictionary with keys as the device number and values as the started Thread.

Return type

dict[int,Thread]

```
_StreamUntilStop(pod: Pod8229, file: IOBase) \rightarrow None
```

Saves a log of all packets recieved from the 8229 POD device until the user decides to stop streaming.

Parameters

- pod (POD_8229) POD device to read from.
- **file** (*IOBase*) Opened text file to save data to.

Setup.SetupOneDevice.Setup_8401HR module

class Setup.SetupOneDevice.Setup_8401HR.Setup8401HR

```
Bases: SetupInterface
```

Setup_8401HR provides the setup functions for an 8206-HR POD device. REQUIRES FIRMWARE 1.0.2 OR HIGHER.

```
static GetDeviceName() \rightarrow str
```

returns the name of the 8401-HR POD device.

Returns

8401-HR.

Return type

str

$StopStream() \rightarrow None$

Write a command to stop streaming data to all POD devices.

```
static \_CodeDCmode(dcMode: str) \rightarrow int
```

gets the integer payload to use for 'SET DC MODE' commands given the mode.

Parameters

dcMode (str) – DC mode VBIAS or AGND.

Raises

Exception – DC Mode value is not supported.

Returns

Integer code representing the DC mode.

Return type

int

$\textbf{static _CodeHighpass}(\textit{highpass: float}) \rightarrow \text{int}$

Gets the integer payload to use for 'SET HIGHPASS' given a highpass value.

Parameters

highpass (*float*) – Highpass value in Hz.

Raises

Exception – High-pass value is not supported.

Returns

Integer code representing the highpass value.

Return type

int

_ConnectPODdevice($deviceNum: int, deviceParams: Params8401HR) <math>\rightarrow$ bool

Creates a POD_8206HR object and write the setup parameters to it.

Parameters

- **deviceNum** (*int*) Integer of the device's number.
- **deviceParams** (*Params_8401HR*) Device parameters.

Returns

True if connection was successful, false otherwise.

Return type

bool

$\texttt{_GetPODdeviceParameterTable}() \rightarrow \texttt{Texttable}$

Builds a table containing the parameters for all POD devices.

Returns

Texttable containing all parameters.

Return type

Texttable

_GetParam_onePODdevice($forbiddenNames: list[str] = []) \rightarrow Params8401HR$

Asks the user to input all the device parameters.

Parameters

forbiddenNames (list[str]) – List of port names already used by other devices.

Returns

Device parameters.

Return type

Params_8401HR

$_GetPreampDeviceName() \rightarrow str$

Asks the user to select a mouse/rat preamplifier.

Returns

String of the chosen preamplifier.

Return type

str

$\verb|_NiceABCDtableText| (abcdValues: list[int | str | None], channelMap: dict[str, str]) \rightarrow str$

Builds a string that formats the channel values to be input into the parameter table.

Parameters

- abcdValueDict (dict[str,int | str | None]) Dictionary with ABCD keys.
- **channelMap** (dict[str, str]) Maps the ABCD channels to the sensor's channel name.

Returns

String with "channel name: value newline..." for each channel.

Return type

str

_OpenSaveFile_EDF($fname: str, devNum: int) \rightarrow EdfWriter$

Opens EDF file and write header.

Parameters

- **fname** (str) String filename.
- **devNum** (*int*) Integer device number.

Returns

Opened file.

Return type

EdfWriter

_OpenSaveFile_TXT(fname: str) \rightarrow IOBase

Opens a text file and writes the column names. Writes the current date/time at the top of the txt file.

Parameters

fname (str) – String filename.

Returns

Opened file.

Return type

IOBase

_PHYSICAL_BOUND_uV: int = 2046

Class-level integer representing the max/-min physical value in uV. Used for EDF files.

static $_$ **SetBias**(channelName: str) \rightarrow float

Asks the user for the bias voltage in V (+/-2.048V).

Parameters

channelName (str) – Name of the channel.

Returns

A float for thebias voltage in V.

Return type

float

static $_$ **SetDCMode**(channelName: str) \rightarrow str

Asks the user for the DC mode (VBIAS or AGND).

Parameters

channelName (str) – Name of the channel.

Returns

String for the DC mode.

Return type

st

_SetForMappedChannels(message: str, channelMap: dict[str, str], func: function) → tuple[int | None] Asks the user to input values for all channels (excluding no-connects).

Parameters

- **message** (str) Message to ask the user.
- **channelMap** (*dict*[*str*, *str*]) Maps the ABCD channels to the sensor's channel name.
- func (function) a function that asks the user for an input. takes one string parameter
 and returns one value.

Returns

Tuple with user inputs for values for the ABCD channels

Return type

tuple[int|None]

static $_$ **SetHighpass**(channelName: str) \rightarrow float | None

Asks the user for the high-pass in Hz (0.5,1,10Hz, or DC).

Parameters

channelName (str) – Name of the channel.

Returns

A float for the high-pass frequency in Hz, or None if DC.

Return type

float|None

$static _SetLowpass(channelName: str) \rightarrow int | None$

Asks the user for the low-pass in Hz (21-15000Hz).

Parameters

channelName (str) – Name of the channel.

Returns

An integer for the low-pass frequency in Hz.

Return type

int|None

static $_$ **SetPreampGain**(channelName: str) \rightarrow int | None

Asks the user for the preamplifier gain.

Parameters

channelName (str) – Name of the channel.

Returns

An integer for the gain, or None if no gain.

Return type

int|None

static $_$ **SetSSGain**(channelName: str) \rightarrow int

Asks the user for the second stage gain.

Parameters

channelName (str) – Name of the channel.

Returns

An integer for the gain.

Return type

int

_StreamThreading() \rightarrow dict[int, threading.Thread]

Opens a save file, then creates a thread for each device to stream and write data from.

Returns

Dictionary with keys as the device number and values as the started Thread.

Return type

dict[int,Thread]

_StreamUntilStop(pod: Pod8401HR, file: IOBase | EdfWriter, sampleRate: int, devNum: int) → None Streams data from a POD device and saves data to file. Stops looking when a stop stream command is read. Calculates average time difference across multiple packets to collect a continuous time series data.

Parameters

- pod (POD_8206HR) POD device to read from.
- file (IOBase | EdfWriter) open file.
- sampleRate (int) Integer sample rate in Hz.

static _WriteDataToFile_EDF(file: EdfWriter, data: list[numpy.ndarray])

Writes data to an open EDF file.

Parameters

- **file** (*EdfWriter*) opened EDF file.
- data (list[np.ndarray]) List with one item for each channel.

static _WriteDataToFile_TXT(file: IOBase, data: list[numpy.ndarray], t: ndarray)

Writes data to an open text file.

Parameters

- **file** (*IOBase*) opened write file.
- **data** (*list* [*np.ndarray*]) List with one item for each channel.
- t (np.ndarray) list with the time stamps (in seconds).

Setup.SetupOneDevice.Setup 8480SC module

```
class Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
```

Bases: SetupInterface

Setup_8480SC provides the setup functions for an 8480-SC POD device.

_streamMode

Set to True when the user wants to start streaming data from an 8480 POD device, False otherwise.

Type

bool

static GetDeviceName() \rightarrow str

Returns the name of the POD device.

Returns

String of _NAME.

Return type

str

$\textbf{static GetSupportedFileExtensions()} \rightarrow list[str]$

Returns a list containing valid file extensions.

Returns

List of string file extensions.

Return type

list[str]

$StopStream() \rightarrow None$

Update the state flag to signal to stop streaming data.

$static _ChoosePeriod() \rightarrow tuple[int]$

Asks the user an input value for Stimulus Period, which is then seperated into Period_ms and Period_us. Seperation is required because the 'SET STIMULUS' requires 7 items in payload, and the second and third items is the Period ms and Period us.

Returns

Formatted period into millisecs and microsecs.

Return type

tuple[int]

static $_{chooseRepeat()} \rightarrow int$

Asks the user to input a value for the number of stimulus pulses to perform.

Returns

representing repeat count for command 'SET STIMULUS'.

Return type

int

static $_$ ChooseStimulusConfig() \rightarrow int

Asks the user to input values for Config format of Stimulus

Returns

Formatted Stimulus Config value.

Return type

(int)

$\textbf{static _ChooseSyncConfig()} \rightarrow int$

Asks the user to input values for Sync Config bits.

Returns

Value calculated from the input bits, this value would be given as payload.

Return type

int

$\textbf{static _ChooseWidth()} \rightarrow tuple[int]$

Asks the user an input value for Stimulus width, which is then seperated into width_ms and width_us. Seperation is required because the 'SET STIMULUS' requires 7 items in payload, and the fourth and fifth items is the width_ms and width_us.

Returns

Formatted given width into millisecs and microsecs

Return type

tuple[int]

_ConnectPODdevice(deviceNum: int, deviceParams: Params8480SC) → bool

Creates a POD_8206HR object and write the setup parameters to it.

Parameters

- **deviceNum** (*int*) Integer of the device's number.
- **deviceParams** (*Params_8480SC*) Device parameters.

Returns

True if connection was successful, false otherwise.

Return type

bool

_GetPODdeviceParameterTable() → Texttable

Builds a table containing the parameters for all POD devices.

Returns

Texttable containing all parameters.

Return type

Texttable

_GetParam_onePODdevice($forbiddenNames: list[str] = []) \rightarrow Params8480SC$

Asks the user to input all the device parameters.

Parameters

forbiddenNames (*list[str]*) – List of port names already used by other devices.

Returns

Device parameters.

Return type

Params 8480SC

_OpenSaveFile_EDF(fname: str, devNum: int)

EDF files are not supported for 8480 POD devices. Overwrites the parent's method, which would open an EDF file and write the header.

Parameters

- **fname** (*str*) String filename. Not used.
- devNum(int) Integer device number. Not used.

Raises

Exception – EDF filetype is not supported for 8480 POD devices.

_OpenSaveFile_TXT(fname: str) \rightarrow IOBase

Opens a text file and writes the column names. Writes the current date/time at the top of the txt file.

Parameters

fname (str) – String filename.

Returns

Opened file.

Return type

IOBase

_StreamThreading() \rightarrow dict[int, threading.Thread]

Opens a save file, then creates a thread for each device to stream and write data from.

Returns

Dictionary with keys as the device number and values as the started Thread.

Return type

dict[int,Thread]

_StreamUntilStop(pod: Pod8480SC, file: IOBase) \rightarrow None

Saves a log of all packets recieved from the 8480 POD device until the user decides to stop streaming.

Parameters

- **pod** (*POD_8480*) POD device to read from.
- **file** (*IOBase*) Opened text file to save data to.

static $_{\mathtt{TtlSetup}}() \rightarrow \mathrm{int}$

Asks the user to input values for Config format values of TTL Setup.

Returns

Formatted TTL Setup config value.

Setup.SetupOneDevice.Setup_PodInterface module

class Setup.SetupOneDevice.Setup_PodInterface.SetupInterface

Bases: object

Setup_Interface provides the basic interface of required methods for subclasses to implement. SetupPodDevices.py is designed to handle any of these children.

_podDevices

Instance-level dictionary of pod device objects. MUST have keys as device number.

Type

dict[int,POD Basics]

_podParametersDict

Instance-level dictionary of device information. MUST have keys as device number.

Type

dict[int,Params_Interface]

_saveFileName

Instance-level string filename: <path>/file.ext. The device name and number will be appended to the filename.

Type

str

AddPODdevice() \rightarrow None

Asks the user for the parameters for the new device. A new device number is generated.

AreDeviceParamsValid(paramDict: None | dict[int, PodApi.Parameters.ParamsBasic.Params])

Checks if the parameters dictionary is valid.

Parameters

paramDict (None | dict[int,Params_Interface]) - Dictionary of parameters for all POD devices.

Raises

- **Exception** Parameters must be contained in a dictionary.
- **Exception** Device keys must be integer type.
- **Exception** Device parameters must be dictionary type.
- **Exception** Device parameters dictionary is empty.

$ConnectAllPODdevices() \rightarrow bool$

Connects all setup POD devices.

Returns

True if all devices are successfully connected, false otherwise.

Return type

bool

$DisplayPODdeviceParameters() \rightarrow None$

Display all the pod device parameters in a table.

$static GetDeviceName() \rightarrow str$

returns the name of the POD device.

Returns

GENERIC.

Return type

str

${\tt GetPODparametersInit()} \rightarrow {\tt str}$

Gets a dictionary whose keys are the device number and the value is the device parameters dict. :returns: String representation of a dictionary of POD device parameters. The keys are the device number. :rtype: str

$GetSaveFileName() \rightarrow str$

Gets the path and filename where streaming data is saved to.

Returns

String of the save file name and path (_saveFileName).

Return type

str

$\textbf{static GetSupportedFileExtensions()} \rightarrow list[str]$

Returns a list containing valid file extensions.

Returns

List of string file extensions.

Return type

list[str]

$PrintSaveFile() \rightarrow None$

Prints the file path and name that data is saved to. Note that the device name and number will be appended to the end of the filename.

SetupFileName($fileName: str \mid None = None) \rightarrow None$

Gets the file path and name to save streaming data to. Note that the device name and number will be appended to the end of the filename.

Parameters

fileName (str | None, optional) - Name and path of the file, if known. Defaults to None.

$\label{eq:continuous} \textbf{SetupPODparameters}(podParametersDict:\ dict[int,\ PodApi.Parameters.ParamsBasic.Params]\ |\ None = None) \rightarrow None$

Sets the parameters for the POD devices.

Parameters

podParametersDict (dict[int,Params_Interface] | None, optional) - dictionary of the device parameters for all devices. Defaults to None.

$StopStream() \rightarrow None$

Write a command to stop streaming data to all POD devices.

$Stream() \rightarrow dict[int, threading.Thread]$

Tests that all devices are connected then starts streaming data.

Raises

Exception – Test connection failed.

Returns

Dictionary with integer device number keys and Thread values.

Return type

dict[int,Thread]

$ValidateParams() \rightarrow None$

Displays a table of the parameters of all devices, then asks the user if everything is correct. The user can then edit the parameters of a device.

_BuildFileName(devNum: int) \rightarrow str

Appends the device name and number to the end of the file name.

Parameters

devNum (int) – Integer of the device number.

Returns

String file name.

Return type

str

static _ChoosePort($forbidden: list[str] = []) \rightarrow str$

Systems checks user's Operating System, and chooses ports accordingly.

Parameters

forbidden (list[str], optional) – List of port names that are already used. Defaults to [].

Returns

String name of the port.

Return type

str

_ConnectPODdevice($deviceNum: int, deviceParams: Params) \rightarrow bool$

Creates a POD device object and write the setup parameters to it.

Parameters

- deviceNum(int) Integer of the device's number.
- **deviceParams** (*Params_Interface*) Device parameters.

Returns

True if connection was successful, false otherwise.

Return type

bool

_DisconnectAllPODdevices() \rightarrow None

Disconnects all POD devices by deleted all POD obejcts.

_EditParams() \rightarrow None

Asks the user which device to edit, and then asks them to re-input the device parameters.

```
_GetForbiddenNames(exclude: str \mid None = None) \rightarrow list[str]
```

Generates a list of port names used by the active pod devices. There is an option to exclude an additional name from the list.

Parameters

exclude (*str | None*, *optional*) – String port name to exclude from the returned list. Defaults to None.

Returns

List of string names of ports in use.

Return type

list[str]

$_GetPODdeviceParameterTable() \rightarrow Texttable$

Builds a table containing the parameters for all POD devices.

Returns

Table containing all parameters.

Return type

Texttable

_GetParam_allPoDdevices() \rightarrow dict[int, *PodApi.Parameters.ParamsBasic.Params*]

First gets the number of POD devices, then asks the user for the information for each device.

Returns

Dictionary with device numbers for keys and parameters for values.

Return type

dict[int,Params_Interface]

_GetParam_onePODdevice($forbiddenNames: list[str] = []) \rightarrow Params$

Asks the user to input all the device parameters.

Parameters

```
forbiddenNames (list[str]) – List of port names already used by other devices. Defaults to [].
```

Returns

Device parameters.

Return type

Params_Interface

static _GetTimeHeader_forTXT() \rightarrow str

Builds a string containing the current date and time to be written to the text file header.

Returns

String containing the date and time. Each line begins with '#' and ends with a newline.

Return type

str

_OpenSaveFile(devNum: int) \rightarrow IOBase | EdfWriter

Opens a save file for a given device.

Parameters

devNum (*int*) – Integer of the device number.

Returns

Opened file. IOBase for a text file, or EdfWriter for EDF file.

Return type

IOBase | EdfWriter

_OpenSaveFile_EDF($fname: str, devNum: int) \rightarrow EdfWriter$

Opens EDF file and write header.

Parameters

- **fname** (*str*) String filename.
- **devNum** (*int*) Integer device number.

Returns

Opened file.

Return type

EdfWriter

_OpenSaveFile_TXT(fname: str) \rightarrow IOBase

Opens a text file and writes the column names. Writes the current date/time at the top of the txt file.

Parameters

fname (*str*) – String filename.

Returns

Opened file.

Return type

IOBase

static $_$ PrintDeviceNumber(num: int) \rightarrow None

Prints a title with the device number.

Parameters

 $\mathbf{num}(int)$ – Integer of the device number.

_RemoveDevice() \rightarrow None

Asks the user for a device number to remove, then deletes that device. This will only remove a device if there are more than one options.

_SelectDeviceFromDict(action: str) \rightarrow int

Asks the user to select a valid device number. The input must be an integer number of an existing device.

Parameters

action (*str*) – Description of the action to be performed on the device.

Returns

Integer for the device number.

Return type

int

static _SetNumberOfDevices(name: str) \rightarrow int

Asks the user for how many devices they want to setup.

Parameters

name (str) – Name of the POD device type.

Returns

Integer number of POD devices desired by the user.

Return type

int

_StreamThreading() \rightarrow dict[int, threading.Thread]

Opens a save file, then creates a thread for each device to stream and write data from.

Returns

Dictionary with keys as the device number and values as the started Thread.

Return type

dict[int,Thread]

$static _TestDeviceConnection(pod: Pod) \rightarrow bool$

Tests if a POD device can be read from or written. Sends a PING command.

Parameters

pod (*POD_Basics*) – POD device to read to and write from.

Returns

True for successful connection, false otherwise.

Return type

bool

$\texttt{_TestDeviceConnection_All}() \rightarrow bool$

Tests the connection of all setup POD devices.

Returns

True when all devices are successfully connected, false otherwise

Return type

bool

static $_$ **uV**(voltage: float | int) \rightarrow float

Converts volts to microVolts, rounded to 6 decimal places.

Parameters

voltage (float | int) - number of volts.

Returns

voltage in of uV.

Return type

float

Module contents

1.2.2 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p	S
PodApi, 66	Setup, 92
PodApi.Commands, 5	Setup.Inputs, 71
PodApi.Commands.PodCommands, 1	Setup.Inputs.GetUserInput,66
PodApi.Devices, 28	Setup.SetupAllDevices, 74
PodApi.Devices.BasicPodProtocol,9	Setup.SetupAllDevices.Setup_PodDevices,71
PodApi.Devices.PodDevice_8206HR, 15	Setup.SetupOneDevice, 92
PodApi.Devices.PodDevice_8229, 16	Setup.SetupOneDevice.Setup_8206HR,74
PodApi.Devices.PodDevice_8401HR, 20	Setup.SetupOneDevice.Setup_8229,77
PodApi.Devices.PodDevice_8480SC, 25	Setup.SetupOneDevice.Setup_8401HR,79
PodApi.Devices.SerialPorts,9	Setup.SetupOneDevice.Setup_8480SC,83
PodApi.Devices.SerialPorts.PortAccess, 6	Setup.SetupOneDevice.Setup_PodInterface, 86
PodApi.Devices.SerialPorts.SerialComm, 7	
PodApi.Packets,44	
PodApi.Packets.Binary,28	
PodApi.Packets.Binary4,30	
PodApi.Packets.Binary5,33	
PodApi.Packets.Packet, 38	
PodApi.Packets.Standard,42	
PodApi.Parameters, 52	
PodApi.Parameters.Params8206HR,44	
PodApi.Parameters.Params8229,46	
PodApi.Parameters.Params8401HR,48	
PodApi.Parameters.Params8480SC, 49	
PodApi.Parameters.ParamsBasic, 51	
PodApi.Stream, 66	
PodApi.Stream.DataBucket, 58	
PodApi.Stream.DataHose, 60	
PodApi.Stream.DeviceValve, 63	
PodApi.Stream.Drain, 58	
PodApi.Stream.Drain.PodHandler, 56	
PodApi.Stream.Drain.PodHandler.Handle8206HR, 52	
PodApi.Stream.Drain.PodHandler.Handle8401HR,	
53	
PodApi.Stream.Drain.PodHandler.HandlerInterf	ace,
54	
PodApi.Stream.Drain.ToEDF, 56	
PodApi.Stream.Drain.ToFile, 56	
PodApi.Stream.Drain.ToText, 58	
PodApi.Stream.DrainToFile,64	

96 Python Module Index

INDEX

Symbols	(Setup.SetupOneDevice.Setup_84c	80SC.Setup8480SC
_AskOption() (Setup.SetupAllDevices.Setup_PodDevices.SetupA	All static method), 84	
method), 72Choc	oseWidth()(<i>Setup.SetupOneDevice.S</i>	Setup_8480SC.Setup8480SC
_AskToStopStream() (Setup.SetupAllDevices.Setup_PodDevice	s.SetHatiq method), 84	
method), 72Code	eDCmode() (Setup.SetupOneDevice.Se	tup_8401HR.Setup8401HR
AskUserForDevices()	static method), 79	
(Setup.SetupAllDevices.Setup_PodDevices.SetupAffCode	eDecimalAsHex()	(Po-
static method), 72	dApi.Devices.PodDevice_8229.Po	od8229
_BuildFileName() (Setup.SetupOneDevice.Setup_PodInterface.	Setustatic method), 18	
method), 88Code	eHighpass() (Setup.SetupOneDevice	.Setup_8401HR.Setup8401HR
CheckForValidParams()	static method), 79	
(Setup.SetupAllDevices.Setup_PodDevices.SetupAtlCol	lectDrop() (PodApi.Stream.Data	Bucket.Bucket
method) 72	memoa), 39	
_CheckParams() (<i>PodApi.Parameters.Params</i> 8206HR.Param882	lectForDuration()	(Po-
method), 45	dApi.Stream.DataBucket.Bucket	method),
_CheckParams() (PodApi.Parameters.Params8229.Params8229	59	
method) 47 _Col.	lectWhileOpen()	(Po-
_CheckParams() (PodApi.Parameters.Params8401HR.Params84	101HApi.Stream.DataBucket.Bucket	method),
method). 49	00	
_CheckParams() (PodApi.Parameters.Params8480SC.Params84	gectNewDevice()	
method), 50	(Setup.SetupAllDevices.Setup_Poo	dDevices.SetupAll
_CheckParams() (PodApi.Parameters.ParamsBasic.Params	method), 73	
method) 52 _Coni	nectPODdevice()	
_ChoosePeriod() (Setup.SetupOneDevice.Setup_8480SC.Setup	₈₄₈₀ &etup.SetupOneDevice.Setup_820	06HR.Setup8206HR
static method). 84	meinoa), 13	
$_$ ChoosePort() (Setup.SetupOneDevice.Setup $_$ PodInterface $\stackrel{ ext{Cond}}{:}$ Setup	nectPODdevice()	
static method), 88	(Setup.SetupOneDevice.Setup_822	29.Setup8229
ChoosePortLinux() (Po-	method), 77	
dApi.Devices.SerialPorts.PortAccess.FindPorts _Con	nectPODdevice()	
static method), 6	(Setup.SetupOneDevice.Setup_84)	01HR.Setup8401HR
_ChoosePortWindows() (Po-	method), 79	
dApi.Devices.SerialPorts.PortAccess.FindPorts _Con	nectPODdevice()	
static method), 6	(Setup.SetupOneDevice.Setup_84	80SC.Setup8480SC
ChoosePreamnGain()	method), 84	
(Setup.SetupOneDevice.Setup_8206HR.Setup8206HR	nectPODdevice()	
static method), 75	(Setup.SetupOneDevice.Setup_For	dInterface.SetupInterface
ChooseRepeat() (Setup Setup One Device Setup 8480SC Setup)	8480 31e thod), 88	
static method), 84Cus	comitoodiiiilbiioi ()	(Po-
_ChooseStimulusConfig()	dApi.Devices.PodDevice_8480SC	.Pod8480SC
(Setup.SetupOneDevice.Setup 8480SC.Setup8480SC	static method), 27	
static method), 84	tom109SETTTLSETUP()	(Po-
_ChooseSyncConfig()	dApi.Devices.PodDevice_8480SC	.Pod8480SC

static method), 27		(Setup.SetupOneDevice.Setup_8206HR.Setup8206HR
_Custom140SETTIME()	(Po-	method), 75
dApi.Devices.PodDevice_8229.Pod8229		_GetPODdeviceParameterTable()
static method), 18		(Setup.SetupOneDevice.Setup_8229.Setup8229
_CustomSTIMULUS()	(Po-	method), 77
dApi.Devices.PodDevice_8480SC.Pod84	80SC	_GetPODdeviceParameterTable()
static method), 27		(Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
_CustomSYNCCONFIG()	(Po-	method), 80
dApi.Devices.PodDevice_8480SC.Pod84	180SC	_GetPODdeviceParameterTable()
static method), 28		(Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
_DecodeDecimalAsHex()	(Po-	method), 85
dApi.Devices.PodDevice_8229.Pod8229		_GetPODdeviceParameterTable()
static method), 18		(Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
_DisconnectAllPODdevices()		method), 89
(Setup.SetupOneDevice.Setup_PodInterfo	ace.Setu	p <i>LiGetyBar</i> am_allPODdevices()
method), 88		(Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
_DoOption() (Setup.SetupAllDevices.Setup_PodD	evices.S	SetupAll method), 89
method), 73		_GetParam_onePODdevice()
_DrainDropToFile()	(<i>Po-</i>	(Setup.SetupOneDevice.Setup_8206HR.Setup8206HR
dApi. Stream. Drain To File. Drain Bucket		method), 75
method), 65		_GetParam_onePODdevice()
_Drop() (PodApi.Stream.DataHose.Hose method)	, 61	(Setup.SetupOneDevice.Setup_8229.Setup8229
_EditCheckConnect()		method), 78
(Setup.SetupAllDevices.Setup_PodDevice	es.Setup	<i>ALI</i> GetParam_onePODdevice()
method), 73		(Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
_EditParams()(Setup.SetupOneDevice.Setup_Po	odInterfa	uce.SetupIntenofthaed), 80
method), 89		_GetParam_onePODdevice()
_EditSaveFilePath()		(Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
(Setup.SetupAllDevices.Setup_PodDevice	es.Setup	
method), 73		_GetParam_onePODdevice()
_Filter() (PodApi.Stream.DataHose.Hose metho	od), 61	(Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
_Filter_DoNothing()	(Po-	method), 89
dApi.Stream.DataHose.Hose method), 62		_GetParams()(Setup.SetupAllDevices.Setup_PodDevices.SetupAll
_Filter_InsertValue()	(Po-	method), 73
dApi.Stream.DataHose.Hose method), 62		_GetPreampDeviceName()
_Filter_RemoveEntry()	(Po-	(Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
dApi.Stream.DataHose.Hose method), 62		method), 80
_Filter_TakeFuture()	(Po-	
dApi.Stream.DataHose.Hose method), 62		(Setup.SetupOneDevice.Setup_8229.Setup8229
_Filter_TakePast() (PodApi.Stream.DataHose	e.Hose	static method), 78
method), 63		_GetTimeHeader() (Po-
_FixABCDtype() (PodApi.Devices.PodDevice_84	01HR.P	
static method), 23		method), 58
_FixTypeInTuple()	(<i>Po-</i>	_GetTimeHeader_forTXT()
dApi.Parameters.ParamsBasic.Params	static	(Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
method), 52		static method), 89
_Flow() (PodApi.Stream.DataHose.Hose method)), 63	_IsDropAvailableInHose() (Po-
_GetChosenDeviceType()	~	dApi.Stream.DataBucket.Bucket method),
(Setup.SetupAllDevices.Setup_PodDevice	es.Setup	
method), 73		_NiceABCDtableText()
_GetForbiddenNames()	. ~	(Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
(Setup.SetupOneDevice.Setup_PodInterfo	ace.Setu	
method), 89		_OpenSaveFile() (Setup.SetupOneDevice.Setup_PodInterface.SetupInterj
_GetPODdeviceParameterTable()		method), 89

```
_OpenSaveFile_EDF()
                                                              dApi.Devices.BasicPodProtocol.Pod method),
        (Setup.SetupOneDevice.Setup_8206HR.Setup8206HR
                                                              13
        method), 75
                                                     _Read_ToETX() (PodApi.Devices.BasicPodProtocol.Pod
_OpenSaveFile_EDF()
                                                              method), 14
         (Setup.SetupOneDevice.Setup_8229.Setup8229
                                                     _Reconnect() (Setup.SetupAllDevices.Setup_PodDevices.SetupAll
        method), 78
                                                             method), 73
_OpenSaveFile_EDF()
                                                     _RemoveDevice() (Setup.SetupAllDevices.Setup PodDevices.SetupAll
        (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
                                                              method), 74
        method), 80
                                                     _RemoveDevice() (Setup.SetupOneDevice.Setup_PodInterface.SetupInter)
_OpenSaveFile_EDF()
                                                              method), 90
        (Setup.SetupOneDevice.Setup_8480SC.Setup8480SEelectDeviceFromDict()
        method), 85
                                                              (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
_OpenSaveFile_EDF()
                                                              method), 90
        (Setup.SetupOneDevice.Setup_PodInterface.Setup_Iset/Biras() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
        method), 90
                                                              static method), 81
_OpenSaveFile_TXT()
                                                     _SetDCMode() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
        (Setup.SetupOneDevice.Setup_8206HR.Setup8206HR
                                                              static method), 81
        method), 76
                                                     _SetForMappedChannels()
_OpenSaveFile_TXT()
                                                              (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
         (Setup.SetupOneDevice.Setup_8229.Setup8229
                                                              method), 81
        method), 78
                                                     _SetHighpass() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
_OpenSaveFile_TXT()
                                                             static method), 82
        (Setup.SetupOneDevice.Setup_8401HR.Setup8401HRetLowpass() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
        method), 81
                                                              static method), 82
_OpenSaveFile_TXT()
                                                     _SetNumberOfDevices()
        (Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
                                                              (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
        method), 85
                                                              static method), 90
_OpenSaveFile_TXT()
                                                     _SetPreampGain() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
        (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface static method), 82
                                                     _SetSSGain() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
        method), 90
_PHYSICAL_BOUND_uV (Setup.SetupOneDevice.Setup_8206HR.Setup8206HRethod), 82
        attribute), 76
                                                     _Set_Setup_PodDevices()
_PHYSICAL_BOUND_uV (Setup.SetupOneDevice.Setup_8401HR.Setup&$0thp18etupAllDevices.Setup_PodDevices.SetupAll
        attribute), 81
                                                              method), 74
_PrintDeviceNumber()
                                                     _Setup_PodDevices (Setup.SetupAllDevices.Setup_PodDevices.SetupAll
        (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface attribute), 71
        static method), 90
                                                     _ShowCurrentSettings()
_PrintInitCode() (Setup.SetupAllDevices.Setup_PodDevices.SetupAllDevices.SetupAllDevices.Setup_PodDevices.SetupAll
         method), 73
                                                              method), 74
_PrintOptions() (Setup.SetupAllDevices.Setup_PodDevices:StsteenamAl) (Setup.SetupAllDevices.Setup_PodDevices.SetupAll
        method), 73
                                                              method), 74
_ReadPODpacket_Recursive()
                                               (Po-
                                                     _StreamAllDevices()
        dApi.Devices.BasicPodProtocol.Pod method),
                                                              (Setup.SetupAllDevices.Setup_PodDevices.SetupAll
                                                              method), 74
_Read_Binary()(PodApi.Devices.BasicPodProtocol.Pod _StreamThreading()(Setup.SetupOneDevice.Setup_8206HR.Setup8206H
                                                              method), 76
         method), 13
_Read_Binary() (PodApi.Devices.PodDevice_8206HR.Pod&200HRThreading() (Setup.SetupOneDevice.Setup_8229.Setup8229
        method), 15
                                                              method), 78
_Read_Binary() (PodApi.Devices.PodDevice_8401HR.Pod.S401HAMThreading() (Setup.SetupOneDevice.Setup_8401HR.Setup8401H
                                                              method), 82
        method), 24
_Read_GetCommand()
                                               (Po-
                                                     _StreamThreading() (Setup.SetupOneDevice.Setup_8480SC.Setup8480S
        dApi.Devices.BasicPodProtocol.Pod method),
                                                             method), 85
                                                     _StreamThreading()(Setup.SetupOneDevice.Setup_PodInterface.SetupIn
```

method), 91

(Po-

_Read_Standard()

_StreamUntilStop()(Setup.SetupOneDevice.Set	ир_820	<i>6∰o\kagg8296₩</i> ndaryChannels()	(Po-
method), 76		dApi.Packets.Binary5.PacketBinary5	static
$\verb _StreamUntilStop() (Setup.SetupOneDevice.Set $	ир_822	9.Setup8229nethod), 38	
method), 78		_WriteDataToFile_EDF()	
$\verb _StreamUntilStop() (Setup.SetupOneDevice.Set $	ир_840	1HR.Setup& \$0th#К etupOneDevice.Setup_8206HR	.Setup8206HR
method), 82		static method), 76	
$\verb _StreamUntilStop() (Setup.SetupOneDevice.Set $	ир_848		
method), 85		(Setup.SetupOneDevice.Setup_8401HR	.Setup8401HR
_TestDeviceConnection()		static method), 83	
(Setup.SetupOneDevice.Setup_PodInterfa	ice.Setu		
static method), 91		(Setup.SetupOneDevice.Setup_8206HR	.Setup8206HR
_TestDeviceConnection_All()		static method), 76	
(Setup.SetupOneDevice.Setup_PodInterfa	ice.Setu	<i>pL</i> WrifueDataToFile_TXT()	
method), 91		(Setup.SetupOneDevice.Setup_8401HR	.Setup8401HR
_ThreadedDrainBucketToFile()	(Po-	static method), 83	
dApi. Stream. Drain To File. Drain Bucket		ARGUMENTS (PodApi.Commands.PodCommand	ls.CommandSet
method), 65		attribute), 5	
_TimeFunc() (Setup.SetupAllDevices.Setup_PodDestatic method), 74	evices.S	SetupBANARY (PodApi.Commands.PodCommands.Co attribute), 5	ommandSet
_TranslateTTLbyte_ASCII()	(Po-	BuildPortName()	(Po-
dApi.Devices.PodDevice_8206HR.Pod82	`	dApi.Devices.SerialPorts.SerialComm.	`
static method), 15	oom	method), 9	01110
_TtlSetup() (Setup.SetupOneDevice.Setup_8480S	SC Setu	· ·	(Po-
static method), 86	oc.sein _i	dApi.Devices.PodDevice_8401HR.Pod8	*
_ValidateChecksum()	(Po-	attribute), 24	7701111
dApi.Devices.BasicPodProtocol.Pod	static	DESCRIPTION (PodApi. Commands. PodComma	ands CommandSet
method), 14	sianc	attribute), 5	mas.commanasci
_ValidatePreampGain()	(Po-	NAME (PodApi.Commands.PodCommands.Com	mandSet
dApi.Devices.PodDevice_8401HR.Pod84		attribute), 5	Thereas et
static method), 24	01111	NOVALUE (PodApi.Commands.PodCommands.	CommandSet
_ValidateSsGain()	(Po-	attribute), 5	
dApi.Devices.PodDevice_8401HR.Pod84		RETURNS (PodApi.Commands.PodCommands.	CommandSet
static method), 24	01111	attribute), 5	
_Validate_Day()	(Po-	U16 (PodApi.Commands.PodCommands.Com	mandSet
dApi.Devices.PodDevice_8229.Pod8229	(10	attribute), 5	runuser
static method), 19		U32 (PodApi.Commands.PodCommands.Com	mandSet
_Validate_Hours()	(Po-	attribute), 5	
dApi.Devices.PodDevice_8229.Pod8229	(10	U8 (PodApi.Commands.PodCommands.Com	mandSet
static method), 19		attribute), 5	
_Validate_Schedule()	(Po-	commands (PodApi.Commands.PodCommands	.CommandSet
dApi.Devices.PodDevice_8229.Pod8229	(10	attribute), 1	· Communication
static method), 19		serialInst(PodApi.Devices.SerialPorts.Seri	alComm.PortIO
_Validate_Speed()	(Po-	attribute), 7	
dApi.Devices.PodDevice_8229.Pod8229	(10	_commands (PodApi.Devices.BasicPodProtocol.	Pod at-
static method), 20		tribute), 9	2000
_Voltage_PrimaryChannels()	(Po-	_commands (PodApi.Packets.Packet.Packet attrib	ute). 38
dApi.Packets.Binary5.PacketBinary5	static	_customPayload(PodApi.Packets.Standard.Pack	
method), 37		attribute), 42	w
_Voltage_PrimaryChannels_Biosensor()	(Po-	_customPayloadArgs	(Po-
dApi.Packets.Binary5.PacketBinary5	static	dApi.Packets.Standard.PacketStandard	at-
method), 37	Simil	tribute), 42	Ci i
_Voltage_PrimaryChannels_EEGEMG()	(Po-	_options(Setup.SetupAllDevices.Setup_PodDev	vices SetunAll
dApi.Packets.Binary5.PacketBinary5	static	attribute), 71	iccs.sciupini
method), 38	siuit	_podDevices (Setup.SetupOneDevice.Setup_Pod	Unterface SetunInterface
,,		_r - = 3. 2000 (Settip Strip Street Settip_1 00	jour 2 2 p 1 2 . jou C .

attribute), 86	AskForFloat() (Setup.Inputs.GetUserInput.UserInput
_podParametersDict(Setup.SetupOneDevice.Setup_Pod	
attribute), 86	AskForFloatInList()
_port (PodApi.Devices.BasicPodProtocol.Pod attribute), 9	(Setup.Inputs.GetUserInput.UserInput static method), 66
$\verb _preampGain (PodApi.Devices.PodDevice_8206HR.Pod82) PodApi.Devices.PodDevice_8206HR.Pod82 PodApi.Devices.PodDevice_8206HR.Pod82 PodApi.Devices.PodDevices.PodDevice_8206HR.Pod82 PodApi.Devices.PodDevice_8206HR.Pod82 PodApi.Devices.PodDevices_8206HR.Pod82 PodApi.Devices_8206HR.Pod82 PodApi.Devices_8206HR.Pod$	
attribute), 15	(Setup.Inputs.GetUserInput.UserInput static
_preampGain(PodApi.Devices.PodDevice_8401HR.Pod84	
attribute), 20	AskForInput() (Setup.Inputs.GetUserInput.UserInput
_preampGain (PodApi.Packets.Binary4.PacketBinary4	static method), 67
attribute), 30	AskForInt() (Setup.Inputs.GetUserInput.UserInput
_preampGain (PodApi.Packets.Binary5.PacketBinary5 attribute), 33	static method), 67 AskForIntInList() (Setup.Inputs.GetUserInput.UserInput
_saveFileName (Setup.SetupOneDevice.Setup_PodInterfa	ice SetunInterstricanethod) 67
attribute), 86	AskForIntInRange() (Setup.Inputs.GetUserInput.UserInput
_ssGain(PodApi.Devices.PodDevice_8401HR.Pod8401H	
attribute), 20	AskForStrInList() (Setup.Inputs.GetUserInput.UserInput
_ssGain (PodApi.Packets.Binary5.PacketBinary5 at-	static method), 68
tribute), 33	AskForType() (Setup.Inputs.GetUserInput.UserInput
$\verb _streamMode (Setup.SetupOneDevice.Setup_8229.Setup829) $	static method), 68
attribute), 77	${\tt AskForTypeInList()}\ ({\it Setup.Inputs.GetUserInput.UserInput}$
$\verb _streamMode (Setup.SetupOneDevice.Setup_8480SC.Setup$	
attribute), 83	AskForTypeInRange()
_uV() (PodApi.Stream.Drain.PodHandler.HandlerInterfac static method), 55	e.DrainDev[iseti]puld ar ts.GetUserInput.UserInput static method), 69
_uV() (Setup.SetupOneDevice.Setup_PodInterface.SetupIn static method), 91	tanglarian (Setup.Inputs.GetUserInput.UserInput static method), 69
Α	aTTL1 (PodApi.Packets.Binary5.PacketBinary5 at- tribute), 34
AddCommand() (PodApi.Commands.PodCommands.Comm method), 1	tribute), 34
AddPODdevice() (Setup.SetupOneDevice.Setup_PodInter	fæEBeiupInteoftApi.Packets.Binary5.PacketBinary5 at- tribute), 34
method), 86	aTTL4 (PodApi.Packets.Binary5.PacketBinary5 at-
aEXTO (PodApi.Packets.Binary5.PacketBinary5 at-	tribute), 34
tribute), 33 aEXT1 (PodApi.Packets.Binary5.PacketBinary5 at-	Home, 54
tribute), 34	В
AnalogEXT() (PodApi.Packets.Binary5.PacketBinary5 method), 34	bias (PodApi.Parameters.Params8401HR.Params8401HR attribute), 48
AnalogTTL() (PodApi.Packets.Binary5.PacketBinary5 method), 34	BinaryBytesToInt() (PodApi.Packets.Packet.Packet static method), 39
AreDeviceParamsValid()	BinaryBytesToInt_Split() (Po-
(Setup.SetupOneDevice.Setup_PodInterface.Setu	
method), 86	39
ArgumentHexChar() (Po-	BinaryBytesToVoltage() (Po-
dApi. Commands. Pod Commands. Command Set $method$), 2	dApi.Packets.Binary4.PacketBinary4 static method), 30
AsciiBytesToInt() (PodApi.Packets.Packet.Packet	binaryData (PodApi.Packets.Binary.PacketBinary at-
static method), 39	tribute), 28
ASCIIbytesToInt_Split() (Po-	binaryLength (PodApi.Packets.Binary.PacketBinary at-
dApi.Packets.Packet.Packet static method),	tribute), 28
39	BinaryLength() (PodApi.Packets.Binary.PacketBinary
AskForBool() (Setup.Inputs.GetUserInput.UserInput	method), 28
static method), 66	Bucket (class in PodApi.Stream.DataBucket), 58

BuildEmptySchedule() (Po- dApi.Parameters.Params8229.Params8229	Checksum() (PodApi.Devices.BasicPodProtocol.Pod static method), 10
static method), 47	ChoosePort() (PodApi.Devices.BasicPodProtocol.Pod
BuildPODpacket_Standard() (Po-	static method), 10
dApi.Devices.BasicPodProtocol.Pod static method), 9	ChoosePort() (PodApi.Devices.SerialPorts.PortAccess.FindPorts static method), 6
BuildSetDayScheduleArgument() (Po-	Close() (PodApi.Stream.DeviceValve.Valve method), 63
dApi.Devices.PodDevice_8229.Pod8229 static method), 16	CloseFile() (PodApi.Stream.Drain.ToEDF.DrainToEDF method), 56
C	CloseFile() (PodApi.Stream.Drain.ToFile.DrainToFile method), 57
CalculateBiasDAC_GetDACValue() (Po- dApi.Devices.PodDevice_8401HR.Pod8401HR	CloseFile() (PodApi.Stream.Drain.ToText.DrainToTXT method), 58
static method), 20	CloseSerialPort() (Po-
CalculateBiasDAC_GetVout() (Po-	dApi. Devices. Serial Ports. Serial Comm. Port IO
dApi.Devices.PodDevice_8401HR.Pod8401HR	method), 7
static method), 21	CodeDayOfWeek() (Po-
CastFloat() (Setup.Inputs.GetUserInput.UserInput static method), 70	dApi.Devices.PodDevice_8229.Pod8229 static method), 16
CastInt() (Setup.Inputs.GetUserInput.UserInput static	CodeDaySchedule() (Po-
method), 70	dApi.Devices.PodDevice_8229.Pod8229
CastStr() (Setup.Inputs.GetUserInput.UserInput static	static method), 16
method), 70	commandNumber (PodApi.Packets.Packet.Packet at-
Ch() (PodApi.Packets.Binary4.PacketBinary4 method), 30	tribute), 38 CommandNumber() (PodApi.Packets.Packet.Packet
ch0 (PodApi.Packets.Binary4.PacketBinary4 attribute),	method), 40
30	CommandNumberFromName() (Po-
ch1 (PodApi.Packets.Binary4.PacketBinary4 attribute), 30	dApi.Commands.PodCommands.CommandSet method), 2
ch2 (PodApi.Packets.Binary4.PacketBinary4 attribute), 30	CommandSet (class in Po- dApi.Commands.PodCommands), 1
Channel() (PodApi.Packets.Binary5.PacketBinary5	ConnectAllPODdevices()
method), 35	$(Setup. Setup One Device. Setup_Pod Interface. Setup Interface$
${\tt channelLabels} \ ({\it PodApi.Parameters.Params} 8401 HR. {\it Parameters.Params}) \ ({\it PodApi.Parameters.Params}) \ ({\it PodApi.Pa$	nms8401HRnethod), 86
attribute), 49	corruptedPointsRemoved (Po-
channels (PodApi.Packets.Binary5.PacketBinary5 attribute), 33	dApi.Stream.DataHose.Hose attribute), 60
CheckFileExt() (Setup.Inputs.GetUserInput.UserInput	D
static method), 70	data (PodApi.Stream.DataHose.Hose attribute), 60
CheckIfPacketIsValid() (Po-	dataBucket (PodApi.Stream.Drain.ToFile.DrainToFile
dApi.Packets.Binary.PacketBinary static	attribute), 56
method), 28	dataHose (PodApi.Stream.DataBucket.Bucket attribute),
CheckIfPacketIsValid() (Po-	58
dApi.Packets.Binary4.PacketBinary4 static method), 31	dcMode (PodApi.Parameters.Params8401HR.Params8401HR attribute), 49
CheckIfPacketIsValid() (Po-	DecodeChannelBitmask() (Po-
dApi.Packets.Binary5.PacketBinary5 static method), 35	dApi.Devices.PodDevice_8401HR.Pod8401HR static method), 21
CheckIfPacketIsValid() (Po-	DecodeDayAndSchedule() (Po-
dApi.Packets.Packet.Packet static method),	dApi.Devices.PodDevice_8229.Pod8229
40	static method), 17
CheckIfPacketIsValid() (Po-	DecodeDayOfWeek() (Po-
dApi.Packets.Standard.PacketStandard static	dApi.Devices.PodDevice_8229.Pod8229
method), 42	static method). 17

DecodeDaySchedule()	(Po-	dApi. Stream. Drain. Pod Handler. Handler Interface),
dApi.Devices.PodDevice_8229.Pod	a8229	54 Dual-Programme Table ()
static method), 17	(P	DrainDropToFile() (Po-
DecodeLCDSchedule()	(Po-	dApi.Stream.Drain.ToEDF.DrainToEDF
dApi.Devices.PodDevice_8229.Pod	a8229	method), 56
static method), 17	(P	DrainDropToFile() (Po-
DecodeSSConfigBitmask()	(Po-	dApi.Stream.Drain.ToFile.DrainToFile
dApi.Devices.PodDevice_8401HR	.Pod8401HR	method), 57
static method), 21	(P	DrainDropToFile() (Po-
DecodeStimulusConfigBits()	(Po-	dApi.Stream.Drain.ToText.DrainToTXT
dApi.Devices.PodDevice_8480SC.	Pod8480SC	method), 58
static method), 25		DrainToEDF (class in PodApi.Stream.Drain.ToEDF), 56
DecodeSyncConfigBits()	(Po-	DrainToFile (class in PodApi.Stream.Drain.ToFile), 56
dApi.Devices.PodDevice_8480SC.	Pod8480SC	drainToFile(PodApi.Stream.DrainToFile.DrainBucket.self
static method), 25		attribute), 64
DecodeTTLByte()	(Po-	DrainToTXT (class in PodApi.Stream.Drain.ToText), 58
dApi.Devices.PodDevice_8401HR	. <i>Pod8401HR</i>	Drip() (PodApi.Stream.DeviceValve.Valve method), 63
static method), 21		DripDrop() (PodApi.Stream.DataBucket.Bucket
<pre>DecodeTTlConfigBits()</pre>	(Po-	method), 59
dApi.Devices.PodDevice_8480SC.	Pod8480SC	drops (PodApi.Stream.DataBucket.Bucket attribute), 58
static method), 25		DropToDf() (PodApi.Stream.Drain.PodHandler.Handle8206HR.Drain820
DecodeTTLPayload()	(Po-	method), 52
dApi.Devices.PodDevice_8401HR	.Pod8401HR	DropToDf() (PodApi.Stream.Drain.PodHandler.Handle8401HR.Drain840
static method), 21		method), 53
<pre>DefaultPayload()</pre>	(Po-	DropToDf() (PodApi.Stream.Drain.PodHandler.HandlerInterface.DrainDe
dApi.Packets.Standard.PacketStand	dard	method), 54
method), 43		DropToListOfArrays() (Po-
Description() (PodApi.Commands.PodComethod), 2	ommands.Com	mandSet dApi.Stream.Drain.PodHandler.Handle8206HR.Drain8206HR method), 53
deviceHandler (PodApi.Stream.Drain.ToF	ile.DrainToFil	
attribute), 57		dApi.Stream.Drain.PodHandler.Handle8401HR.Drain8401HR
deviceValve (PodApi.Stream.DataHose	e.Hose at-	method), 54
tribute), 60		DropToListOfArrays() (Po-
DigitalMax() (PodApi.Stream.Drain.ToED	F.DrainToED	
static method), 56		method), 55
DigitalMin() (PodApi.Stream.Drain.ToED	F.DrainToED	
static method), 56		E
DisplayPODdeviceParameters()		EEG1() (PodApi.Parameters.Params8206HR.Params8206HR
(Setup.SetupOneDevice.Setup_Poo	Unterface Setu	IDInterface method) 45
method), 87	izmer jereen senn	EEG2() (PodApi.Parameters.Params8206HR.Params8206HR
DoesCommandExist()	(Po-	method), 45
dApi.Commands.PodCommands.C		EEG3_EMG() (PodApi.Parameters.Params8206HR.Params8206HR
method), 2	ommanasei	
Drain8206HR (class in	Po-	method), 45 EmptyBucket() (PodApi.Stream.DataBucket.Bucket
dApi.Stream.Drain.PodHandler.Ha		
52	<i>indic</i> 02001111)	memou), es
Drain8401HR (class in	Po-	EmptyHose() (PodApi.Stream.DataHose.Hose method),
dApi.Stream.Drain.PodHandler.He		61
53	maie0+0111K)	- · ·
DrainBucket (class in PodApi.Stream.Drai.	nToFile\ 64	method), 64
DrainBucketToFile()	(Po-	estimCurrent (PodApi.Parameters.Params8480SC.Params8480SC
dApi.Stream.DrainToFile.DrainBu	`	attribute), 50
method), 64	cive i	estimCurrent_CH0() (Po-
	in Po-	dApi.Parameters.Params8480SC.Params8480SC
(cmp)	10-	method), 51

static method), 22

estimCu	urrent_CH1() dApi.Parameters.Params8480SC.Param		GetChannels() (PodApi.Packets.Binary5.PacketBinary5 static method), 36
ETV()(method), 51 PodApi.Packets.Packet.Packet static metho		GetCommandNumber() (PodApi.Packets.Packet.Packet static method), 40
	гоалрі.1 аскеіз.1 аскеі.1 аскеі зіанс тето	<i>a</i>), 40	GetCommands() (PodApi.Commands.PodCommands.CommandSet
F			method), 3
file	(PodApi.Stream.Drain.ToEDF.DrainToED	F at-	GetCurrentTime() (Po-
	tribute), 56		dApi.Devices.PodDevice_8229.Pod8229
file	(PodApi.Stream.Drain.ToText.DrainToTX	T at-	static method), 17
	tribute), 58		GetDeviceColNames() (Po-
fileNa	me (PodApi.Stream.Drain.ToFile.DrainTo	File at-	dApi.Stream.Drain.PodHandler.Handle8206HR.Drain8206HR method), 53
£:1+0m	tribute), 56		GetDeviceColNames() (Po-
filter	* *	se at-	dApi.Stream.Drain.PodHandler.Handle8401HR.Drain8401HR
filter	tribute), 60	a at	method), 54
TITTEL	<pre>Method (PodApi.Stream.DataHose.Hos tribute), 60</pre>	se at-	GetDeviceColNamesList() (Po-
FindPo	***	Po-	dApi.Stream.Drain.PodHandler.Handle8206HR.Drain8206HR
rinaro	dApi.Devices.SerialPorts.PortAccess), 6		method), 53
Fluch() (PodApi.Devices.SerialPorts.SerialCom		GetDeviceColNamesList() (Po-
1 1 4 5 11 ()	method), 7	11.11 01110	dApi.Stream.Drain.PodHandler.Handle8401HR.Drain8401HR
FlushPo		col.Pod	method), 54
I I UDIII (method), 10	co ou	GetDeviceColNamesList() (Po-
	<i></i>		dApi.Stream.Drain.PodHandler.HandlerInterface.DrainDeviceHo
G			method), 55
Get Alli	PortNames()	(Po-	GetDeviceCommands() (Po-
GC CIII I	dApi.Devices.SerialPorts.PortAccess.Fit		$dApi.Devices.BasicPodProtocol.Pod\ method),$
	static method), 6		10
GetAna:	logEXT() (PodApi.Packets.Binary5.Packetstatic method), 35	etBinary5	GetDeviceName() (Setup.SetupOneDevice.Setup_8206HR.Setup8206HR static method), 75
GetAna:	logTTL() (PodApi.Packets.Binary5.Packe	tBinary5	GetDeviceName() (Setup.SetupOneDevice.Setup_8229.Setup8229
	static method), 35	·	static method), //
GetBas:	icCommands()	(Po-	GetDeviceName() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
	dApi.Commands.PodCommands.Comm	andSet	static method), 79
	static method), 2		GetDeviceName() (Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
GetBin	aryData()	(<i>Po-</i>	<pre>static method), 83 GetDeviceName() (Setup.SetupOneDevice.Setup_PodInterface.SetupInterf</pre>
	dApi.Packets.Binary.PacketBinary m	iethod),	static method), 87
	29		GetExtension() (PodApi.Stream.Drain.ToFile.DrainToFile
GetBina	aryLength()	(Po-	static method), 57
	dApi.Packets.Binary.PacketBinary	static	GetFileName() (Setup.Inputs.GetUserInput.UserInput
a . n.	method), 29	(P	static method), 70
GetBina	aryLength()	(Po-	GetFilePath() (Setup.Inputs.GetUserInput.UserInput
	dApi.Packets.Binary4.PacketBinary4	static	static method), 71
CotDin	<pre>method), 31 aryLength()</pre>	(Po-	GetHandlerForBucket() (Po-
Gerpin	dApi.Packets.Binary5.PacketBinary5	static	dApi.Stream.Drain.ToFile.DrainToFile static
	method), 36	sianc	method), 57
Co+Ch(memoa), 50) (PodApi.Packets.Binary4.PacketBinary4	1 static	GetInit() (PodApi.Parameters.Params8206HR.Params8206HR
de cerre,	method), 31	T SIUIIC	method), 45
Get Chai	nnelBitmask()	(Po-	<pre>GetInit() (PodApi.Parameters.Params8229.Params8229</pre>
JC CCIIAI	dApi.Devices.PodDevice_8401HR.Pod8	`	method), 47
	static method), 21		<pre>GetInit() (PodApi.Parameters.Params8401HR.Params8401HR</pre>
GetChai	nnelMapForPreampDevice()	(Po-	method), 49
	dApi.Devices.PodDevice_8401HR.Pod8	•	GetInit() (PodApi.Parameters.Params8480SC.Params8480SC
	static method) 22		method), 50

<pre>GetInit() (PodApi.Parameters.ParamsBasic.Par</pre>	rams	static method), 36
method), 51		GetStopBytes() (PodApi.Stream.DeviceValve.Valve
<pre>GetMinimumLength()</pre>	(Po-	method), 64
dApi.Packets.Binary.PacketBinary s	tatic	<pre>GetSupportedFileExtensions()</pre>
method), 29		(Setup.SetupOneDevice.Setup_8229.Setup8229
	(Po-	static method), 77
_	tatic	<pre>GetSupportedFileExtensions()</pre>
method), 31		(Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
	(Po-	static method), 83
	tatic	GetSupportedFileExtensions()
method), 36		(Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
GetMinimumLength() (PodApi.Packets.Packet.Packet.Packets.Packet.Packets.Packet	acket	static method), 87
static method), 40	ichci	GetSupportedPreampDevices() (Po-
	(Po-	dApi.Devices.PodDevice_8401HR.Pod8401HR
	tatic	static method), 22
method), 43	шис	
		GetTTL() (PodApi.Packets.Binary4.PacketBinary4 static
GetOptions() (Setup.SetupAllDevices.Setup_PodDe	evices.	
method), 71	(D-	GetTTLbitmask() (Po-
	(Po-	dApi.Devices.PodDevice_8401HR.Pod8401HR
	tatic	static method), 22
method), 32		GetU() (PodApi.Devices.BasicPodProtocol.Pod static
**	(Po-	method), 11
	tatic	GetVolumeOfDrops() (Po-
method), 36		dApi.Stream.DataBucket.Bucket method),
<pre>GetPayload() (PodApi.Packets.Standard.PacketStar</pre>	ıdard	59
static method), 43		11
Ca+DadEmamDualta+()	(Po-	

dApi.Stream.Drain.PodHandler.HandlerInt		
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55	terface	(PodApi.Packets.Packet.Packet method), 41
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55	terface	·ParsoonmeineMandbar() (PodApi.Packets.Packet.Packet
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55	terface	(PodApi.Packets.Packet.Packet method), 41
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtoco	terface	HastonmeineMandler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method),
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtoco method), 10	terface	**PastonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (Po-
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtoco method), 10 GetPODparametersInit()	terface	**PastonmeindNandbar() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (Po-
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72	terface	*PastonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72 GetPODparametersInit()	terface ol.Pod Setupe	*PastonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72	terface ol.Pod Setupe	Has Commend Mandbar() (PodApi.Packets.Packet.Packet method), 41 Has Commands () (PodApi.Packets.Packet.Packet method), 41 Has Custom Payload () (PodApi.Packets.Standard.PacketStandard method), 43 Has Payload () (PodApi.Packets.Standard.PacketStandard method), 43 Has Payload () (PodApi.Packets.Standard.PacketStandard method), 43
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72 GetPODparametersInit() (Setup.SetupOneDevice.Setup_PodInterface method), 87	terface ol.Pod Setup e.Setup	Has Command () (PodApi. Packets. Packet method), 41 Has Commands () (PodApi. Packets. Packet method), 41 Has Custom Payload () (PodApi. Packets. Standard. Packet Standard method), 43 Has Payload () (PodApi. Packets. Standard. Packet Standard pointerface method), 43 high Pass (PodApi. Parameters. Params 8401 HR. Params 8401 HR
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72 GetPODparametersInit() (Setup.SetupOneDevice.Setup_PodInterface method), 87 GetPortName() (PodApi.Devices.SerialPorts.Serial	terface ol.Pod Setup e.Setup	**RastonmeindNandbar() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard plnterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72 GetPODparametersInit() (Setup.SetupOneDevice.Setup_PodInterface method), 87 GetPortName() (PodApi.Devices.SerialPorts.Serial method), 7	verface ol.Pod Setupe e.Setup Comm	**RasyonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard plnterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices.method), 72 GetPODparametersInit() (Setup.SetupOneDevice.Setup_PodInterface method), 87 GetPortName() (PodApi.Devices.SerialPorts.Serial method), 7 GetSampleRate() (PodApi.Stream.DataHose.BasicPodParametersInit())	verface ol.Pod Setupe e.Setup Comm	HasCommands() (PodApi.Packets.Packet nethod), 41 HasCommands() (PodApi.Packets.Packet nethod), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard olinterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR.PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	terface ol.Pod Setup? e.Setup Comm Hose	HasCommands() (PodApi.Packets.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard olnterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	erface ol.Pod Setup e.Setup Comm Hose _PodIr	**RastonmeindNandbar() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard onterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR.PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 nterface.SetupInterface
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	erface ol.Pod Setup e.Setup Comm Hose _PodIr	HasCommands() (PodApi.Packets.Packet method), 41 HasCommands() (PodApi.Packets.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard olinterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 interface.SetupInterface DivictorSecurity (PodApi.Packets.Packet.Packet
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupa Setupa Setupa Comm Hose _PodIn	HasCommindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard otherface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 interface.SetupInterface DivictSascuribytes() (PodApi.Packets.Packet.Packet static method), 41
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupa Setupa Comm Hose _PodIn (Po-	**ParsoommeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR.PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 hterface.SetupInterface Prices SetupInterface Prices SetupInterface method), 41 isCollecting (PodApi.Stream.DataBucket.Bucket at-
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupa Setupa Comm Hose _PodIn (Po-	**RastonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 sterface.SetupInterface DeniceSonSetupInterface DeniceSonSetupInterface
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit() (Setup.SetupAllDevices.Setup_PodDevices. method), 72 GetPODparametersInit() (Setup.SetupOneDevice.Setup_PodInterface method), 87 GetPortName() (PodApi.Devices.SerialPorts.Serial method), 7 GetSampleRate() (PodApi.Stream.DataHose.Bastatic method), 61 GetSaveFileName() (Setup.SetupOneDevice.Setup_method), 87 GetSaveFileNames() (Setup.SetupAllDevices.Setup_method), 72 GetSelectPortNames() (Setup.SetupAllDevices.Setup_method), 72 GetSelectPortNames() (Setup.SetupAllDevices.Setup_method), 6	Setup Setup Setup Comm Hose PodIr PodIr (Po-	**RasyonmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard obliterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR.PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 iterface.SetupInterface Office(SastupInterface) Office(SastupInterface) Office(SastupInterface) IsCollecting (PodApi.Stream.DataBucket.Bucket attribute), 59 IsCollecting() (PodApi.Stream.DrainToFile.DrainBucket)
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupe Setupe Setupe Comm Hose Podle (Po- Ports (Po-	**RasyonmindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard olinterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 merface.SetupInterface Office(SastupInterface) (PodApi.Packets.Packet.Packet static method), 41 isCollecting (PodApi.Stream.DataBucket.Bucket attribute), 59 IsCollecting() (PodApi.Stream.DrainToFile.DrainBucket method), 64
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupe Setupe Setupe Comm Hose Podle (Po- Ports (Po-	**RasbohmeindNambler() (PodApi.Packets.Packet.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.Packets.Standard.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard olinterface method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 atterface.SetupInterface Drivero SetupInterface
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupe Setupe Setupe Comm Hose Podle (Po- Ports (Po- IHR	**HasCommands() (PodApi.Packets.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 atterface.SetupInterface DriveroasCurrent Continue Conti
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupe Setupe Setupe Comm Hose Podle (Po- Ports (Po- IHR	**Parsocomment Mandbar**() (PodApi.Packets.Packet.Packet method), 41 **HasCommands*() (PodApi.Packets.Packet.Packet method), 41 **HasCustomPayload*() (PodApi.Packets.Standard.PacketStandard method), 43 **HasPayload*() (PodApi.Packets.Standard.PacketStandard obliterface method), 43 **highPass*(PodApi.Parameters.Params8401HR.Params8401HR.PortIO attribute), 48 **Hose*(class in PodApi.Stream.DataHose), 60 **hoursPerDay*(PodApi.Parameters.Params8229.Params8229 attribute), 47 **atterface.SetupInterface** DrictOsSchipAll**(tes() (PodApi.Packets.Packet.Packet static method), 41 isCollecting*(PodApi.Stream.DataBucket.Bucket attribute), 59 IsCollecting*() (PodApi.Stream.DrainToFile.DrainBucket method), 64 IsCommandBinary*() (PodApi.Commands.CommandSet method), 3
dApi.Stream.Drain.PodHandler.HandlerInt static method), 55 GetPODpacket() (PodApi.Devices.BasicPodProtocol method), 10 GetPODparametersInit()	Setupa Setupa Setupa Comm Hose _PodIn (Po- Ports (Po- I HR	**HasCommands() (PodApi.Packets.Packet method), 41 HasCommands() (PodApi.Packets.Packet.Packet method), 41 HasCustomPayload() (PodApi.PacketStandard method), 43 HasPayload() (PodApi.Packets.Standard.PacketStandard method), 43 highPass (PodApi.Parameters.Params8401HR.Params8401HR .PortIO attribute), 48 Hose (class in PodApi.Stream.DataHose), 60 hoursPerDay (PodApi.Parameters.Params8229.Params8229 attribute), 47 atterface.SetupInterface DriveroasCurrelly* tes() (PodApi.Packets.Packet.Packet static method), 41 isCollecting (PodApi.Stream.DataBucket.Bucket attribute), 59 IsCollecting() (PodApi.Stream.DrainToFile.DrainBucket method), 64 IsCommandBinary() (PodApi.CommandSet)

method), 65	PodApi.Stream, 66
<pre>IsPreampDeviceSupported()</pre>	PodApi.Stream.DataBucket,58
dApi.Devices.PodDevice_8401HR.Pod8401HR	PodApi.Stream.DataHose, 60
static method), 23	PodApi.Stream.DeviceValve, 63
IsSerialClosed() (Po-	PodApi.Stream.Drain, 58
dApi.Devices.SerialPorts.SerialComm.PortIO	PodApi.Stream.Drain.PodHandler,56
method), 7	PodApi.Stream.Drain.PodHandler.Handle8206HR,
<pre>IsSerialOpen() (PodApi.Devices.SerialPorts.SerialCom</pre>	m.PortIO 52
method), 7	PodApi.Stream.Drain.PodHandler.Handle8401HR, 53
L	PodApi.Stream.Drain.PodHandler.HandlerInterface,
ledCurrent (PodApi.Parameters.Params8480SC.Params8	3480SC 54
attribute), 50	PodApi.Stream.Drain.ToEDF, 56
ledCurrent_CHO() (Po-	PodApi.Stream.Drain.ToFile, 56
dApi.Parameters.Params8480SC.Params8480SC	PodApi.Stream.Drain.ToText, 58
method), 51	PodApi.Stream.DrainToFile,64
ledCurrent_CH1() (Po-	Setup, 92
dApi.Parameters.Params8480SC.Params8480SC	Setup.Inputs, 71
method), 51	Setup.Inputs.GetUserInput,66
lowPass (PodApi.Parameters.Params8206HR.Params8206	GHR Setup.SetupAllDevices,74
attribute), 45	<pre>Setup.SetupAllDevices.Setup_PodDevices,</pre>
lowPass (PodApi.Parameters.Params8401HR.Params840)	1HR 71
attribute), 48	Setup.SetupOneDevice,92
${\tt lowPassLabels} (\textit{PodApi.Parameters.Params} 8206 HR. \textit{Parameters.Params} 8206 HR. Parame$	ams82 %etyp p.SetupOneDevice.Setup_82 0 6HR,74
attribute), 46	Setup.SetupOneDevice.Setup_8229,77
	Setup.SetupOneDevice.Setup_8401HR,79
M	Setup.SetupOneDevice.Setup_8480SC,83
mode (PodApi.Parameters.Params8229.Params8229 at-	<pre>Setup.SetupOneDevice.Setup_PodInterface,</pre>
<i>tribute</i>), 46	86
module	motorDirection(PodApi.Parameters.Params8229.Params8229
PodApi, 66	attribute), 46
PodApi.Commands, 5	motorSpeed(PodApi.Parameters.Params8229.Params8229
PodApi.Commands.PodCommands, 1	attribute), 46
PodApi.Devices, 28	muxMode (PodApi.Parameters.Params8401HR.Params8401HR
PodApi.Devices.BasicPodProtocol,9	attribute), 48
PodApi.Devices.PodDevice_8206HR, 15	NI
PodApi.Devices.PodDevice_8229, 16	N
PodApi.Devices.PodDevice_8401HR, 20	${\tt NoValue()}\ (Pod Api. Commands. Pod Commands. Command Set$
PodApi.Devices.PodDevice_8480SC, 25	static method), 3
PodApi.Devices.SerialPorts,9	numDrops (PodApi.Stream.DataHose.Hose attribute), 60
PodApi.Devices.SerialPorts.PortAccess, 6	
PodApi.Devices.SerialPorts.SerialComm, 7	0
PodApi.Packets, 44	Open() (PodApi.Stream.DeviceValve.Valve method), 64
PodApi.Packets.Binary,28	OpenFile() (PodApi.Stream.Drain.ToEDF.DrainToEDF
PodApi.Packets.Binary4,30	method), 56
PodApi.Packets.Binary5,33	OpenFile() (PodApi.Stream.Drain.ToFile.DrainToFile
PodApi.Packets.Packet,38	method), 57
PodApi.Packets.Standard,42	OpenFile() (PodApi.Stream.Drain.ToText.DrainToTXT
PodApi.Parameters, 52	method), 58
PodApi.Parameters.Params8206HR,44	OpenSerialPort() (Po-
PodApi.Parameters.Params8229,46	dApi.Devices.SerialPorts.SerialComm.PortIO
PodApi.Parameters.Params8401HR,48	method), 8
PodApi.Parameters.Params8480SC, 49	
PodApi.Parameters.ParamsBasic, 51	

P	module, 15
Packet (class in PodApi.Packets.Packet), 38	PodApi.Devices.PodDevice_8229
PacketBinary (class in PodApi.Packets.Binary), 28	module, 16
PacketBinary4 (class in PodApi.Packets.Binary4), 30	PodApi.Devices.PodDevice_8401HR
PacketBinary5 (class in PodApi.Packets.Binary5), 33	module, 20
packetNumber (PodApi.Packets.Binary4.PacketBinary4	PodApi.Devices.PodDevice_8480SC
attribute), 30	module, 25
packetNumber (PodApi.Packets.Binary5.PacketBinary5	PodApi.Devices.SerialPorts
attributa) 33	module, 9
PacketNumber() (PodApi.Packets.Binary4.PacketBinary4	<pre>PodApi.Devices.SerialPorts.PortAccess</pre>
method), 32	module, 6
PacketNumber() (PodApi.Packets.Binary5.PacketBinary.	5 PodApi.Devices.SerialPorts.SerialComm
method), 36	module, 7
PacketStandard (class in PodApi.Packets.Standard), 42	PodApi.Packets
Params (class in PodApi.Parameters.ParamsBasic), 51	module, 44
Params8206HR (class in Po-	PodApi.Packets.Binary
dApi.Parameters.Params8206HR), 44	module, 28
Params8229 (class in PodApi.Parameters.Params8229),	PodApi.Packets.Binary4
46	module, 30
Params8401HR (class in Po-	PodApi.Packets.Binary5
dApi.Parameters.Params8401HR), 48	module, 33
Params8480SC (class in Po-	PodApi.Packets.Packet
dApi.Parameters.Params8480SC), 49	module, 38
payload (PodApi.Packets.Standard.PacketStandard at-	PodApi.Packets.Standard
tribute), 42	module, 42
Payload() (PodApi.Packets.Standard.PacketStandard	PodApi.Parameters
method), 43	module, 52
PayloadToBytes() (Po-	PodApi.Parameters.Params8206HR
dApi.Devices.BasicPodProtocol.Pod static	module, 44
method), 11	PodApi.Parameters.Params8229
PhysicalBound() (Po-	module, 46
dApi. Stream. Drain. To EDF. Drain To EDF	PodApi.Parameters.Params8401HR
static method), 56	module, 48
PickFilterMethod() (PodApi.Stream.DataHose.Hose	PodApi.Parameters.Params8480SC
method), 61	module, 49
Pod (class in PodApi.Devices.BasicPodProtocol), 9	PodApi.Parameters.ParamsBasic
Pod8206HR (class in Po-	module, 51
dApi.Devices.PodDevice_8206HR), 15	PodApi.Stream
Pod8229 (class in PodApi.Devices.PodDevice_8229), 16	module, 66
Pod8401HR (class in Po-	PodApi.Stream.DataBucket
dApi.Devices.PodDevice_8401HR), 20	module, 58
Pod8480SC (class in Po-	PodApi.Stream.DataHose module, 60
dApi.Devices.PodDevice_8480SC), 25	PodApi.Stream.DeviceValve
PodApi	module, 63
module, 66	PodApi.Stream.Drain
PodApi.Commands	module, 58
module, 5	PodApi.Stream.Drain.PodHandler
PodApi.Commands.PodCommands	module, 56
module, 1	PodApi.Stream.Drain.PodHandler.Handle8206HR
PodApi.Devices	module, 52
module, 28	PodApi.Stream.Drain.PodHandler.Handle8401HR
PodApi.Devices.BasicPodProtocol	module, 53
module, 9 PodApi.Devices.PodDevice 8206HR	PodApi.Stream.Drain.PodHandler.HandlerInterface

module, 54	ReadPODpacket() (Po-
PodApi.Stream.Drain.ToEDF	dApi.Devices.PodDevice_8229.Pod8229
module, 56	method), 17
PodApi.Stream.Drain.ToFile	ReadPODpacket() (Po-
module, 56	dApi.Devices.PodDevice_8401HR.Pod8401HR
PodApi.Stream.Drain.ToText	method), 23
module, 58	ReadPODpacket() (Po-
PodApi.Stream.DrainToFile	dApi.Devices.PodDevice_8480SC.Pod8480SC
module, 64	method), 25
podDevice (PodApi.Stream.DeviceValve.Valve attribute),	<pre>ReadUntil() (PodApi.Devices.SerialPorts.SerialComm.PortIO</pre>
port (PodApi.Parameters.Params8206HR.Params8206HR	
attribute), 44	dApi.Commands.PodCommands.CommandSet
port (PodApi.Parameters.Params8229.Params8229 at-	method), 3
tribute), 46	RestoreBasicCommands() (Po-
port (PodApi.Parameters.Params8401HR.Params8401HR	dApi.Commands.PodCommands.CommandSet
attribute), 48	method), 3
port (PodApi.Parameters.Params8480SC.Params8480SC	ReturnHexChar() (Po-
attribute), 49	dApi.Commands.PodCommands.CommandSet
port (PodApi.Parameters.ParamsBasic.Params at-	method), 3
tribute), 51	reverseBaseTime (Po-
PortIO (class in Po-	dApi.Parameters.Params8229.Params8229
dApi.Devices.SerialPorts.SerialComm), 7	attribute), 46
preamp (PodApi.Parameters.Params8480SC.Params8480Sc attribute), 50	CreverseVarTime (PodApi.Parameters.Params8229.Params8229 attribute), 46
	nR840(I)HRetup.SetupAllDevices.Setup_PodDevices.SetupAll
attribute), 48	method), 72
preampDevice (PodApi.Stream.Drain.PodHandler.Handle attribute), 53	
preampDevice (PodApi.Stream.Drain.ToFile.DrainToFile	
attribute), 56	sampleRate (PodApi.Parameters.Params8206HR.Params8206HR attribute), 44
preampGain (PodApi.Parameters.Params8401HR.Params8	Stant ReRate (PodApi.Parameters.Params8401HR.Params8401HR
attribute), 48	attribute), 48
preamplifierGain (Po-	sampleRate (PodApi.Stream.DataHose.Hose attribute),
dApi.Parameters.Params8206HR.Params8206HR	
attribute), 45 PrintSaveFile() (Setup.SetupOneDevice.Setup_PodInter	· · · · · · · · · · · · · · · · · · ·
method), 87	Search() (PodApi.Commands.PodCommands.CommandSet method), 4
R	SetBaudrate() (PodApi.Devices.SerialPorts.SerialComm.PortIO
randomReverse (PodApi.Parameters.Params8229.Params	
attribute), 46	SetBaudrateOfDevice() (Po-
rawPacket (PodApi.Packets.Packet.Packet attribute), 38	dApi.Devices.BasicPodProtocol.Pod method),
Read() (PodApi.Devices.SerialPorts.SerialComm.PortIO	11
method), 8	SetCustomPayload() (Po-
ReadLine() (PodApi.Devices.SerialPorts.SerialComm.Pormethod), 8	
ReadPODpacket() (Po-	SetFilterInsertValue() (Po-
dApi.Devices.BasicPodProtocol.Pod method),	dApi.Stream.DataHose.Hose method), 61
11	Setup
ReadPODpacket() (Po-	module, 92
dApi.Devices.PodDevice_8206HR.Pod8206HR	Setup.Inputs
method), 15	module, 71
memou), 13	Setup.Inputs.GetUserInput

			JA D D . JD 0490CC D . J9490CC
module, 66			dApi.Devices.PodDevice_8480SC.Pod8480SC
Setup.SetupAllDevic	es		static method), 26
module, 74			StopCollecting() (PodApi.Stream.DataBucket.Bucket
	es.Setup_PodDevices		method), 59
module, 71			StopStream() (PodApi.Stream.DataHose.Hose
Setup.SetupOneDevic	e		method), 61
module, 92			StopStream() (Setup.SetupOneDevice.Setup_8206HR.Setup8206HR
Setup.SetupOneDevic	e.Setup_8206HR		method), 75
module, 74			StopStream() (Setup.SetupOneDevice.Setup_8229.Setup8229
Setup.SetupOneDevic	e.Setup_8229		method), 77
module, 77			StopStream() (Setup.SetupOneDevice.Setup_8401HR.Setup8401HR
Setup.SetupOneDevic	e.Setup_8401HR		method), 79
module, 79			StopStream() (Setup.SetupOneDevice.Setup_8480SC.Setup8480SC
Setup.SetupOneDevic	e.Setup_8480SC		method), 83
module, 83	-		StopStream() (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
	e.Setup_PodInterface		method), 88
module, 86	•-		Stream() (Setup.SetupOneDevice.Setup_PodInterface.SetupInterface
Setup8206HR	(class	in	method), 88
_	eDevice.Setup_8206HR), 74		streamCmd (PodApi.Stream.DeviceValve.Valve attribute),
	up.SetupOneDevice.Setup_822	29)	63
77	p.seinponeBevice.seinp <u>-</u> 021		streamPldStart (PodApi.Stream.DeviceValve.Valve at-
Setup8401HR	(class	in	tribute), 63
-	eDevice.Setup_8401HR), 79	ırı	streamPldStop (PodApi.Stream.DeviceValve.Valve at-
	* — · · ·	<i>:-</i> .	
Setup8480SC	(class	in	tribute), 63
	eDevice.Setup_8480SC), 83		STX() (PodApi.Packets.Packet.Packet static method), 41
	.SetupAпDevices.Setup_Poat	Devic	esymcConfig (PodApi.Parameters.Params8480SC.Params8480SC
71			attribute), 50
C . T'1 N () (C .		77 .	
	up.SetupOneDevice.Setup_Po	dInte	Sync Scoupfing Bifase() (Po-
method), 87			r fsyna CScompffingBrifacs() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC
<pre>method), 87 SetupInterface</pre>	(class	in	(Po-dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26
method), 87 SetupInterface Setup.SetupOn		in	r Sync Scompfing Britise() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229
method), 87 SetupInterface Setup.SetupOn 86	(class eDevice.Setup_PodInterface)	in	(Po-dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26
<pre>method), 87 SetupInterface</pre>	(class eDevice.Setup_PodInterface)	in ,	*SyncScomffingBrface() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46
<pre>method), 87 SetupInterface</pre>	(class eDevice.Setup_PodInterface)	in ,	*SyncScomffingBrface() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46
<pre>method), 87 SetupInterface</pre>	(class eDevice.Setup_PodInterface)	in ,	r Sync Scompfing Britise() (Po- dApi. Devices. Pod Device_8480 SC. Pod 8480 SC static method), 26 systemID (Pod Api. Parameters. Params 8229. Params 8229 attribute), 46
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl	(class eDevice.Setup_PodInterface) () lDevices.Setup_PodDevices.S	in ,	r Sayne Scompfing Britises() (Po- dApi. Devices. PodDevice_8480SC. Pod8480SC static method), 26 systemID (PodApi. Parameters. Params 8229. Params 8229 attribute), 46 All TestConnection() (Po-
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((class eDevice.Setup_PodInterface) () lDevices.Setup_PodDevices.S	in , Setup	rsyncScompfingBrituse() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method),
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87	(class eDevice.Setup_PodInterface) () lDevices.Setup_PodDevices.S () neDevice.Setup_PodInterface	in , Setupa .Setup	rsyncScompfingPrfates() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), bInterface 12 totalDropsCollected (Po-
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87	(class eDevice.Setup_PodInterface) () lDevices.Setup_PodDevices.S () neDevice.Setup_PodInterface	in , Setupa .Setup	rsyncScompfingPrfates() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), bInterface 12 totalDropsCollected (Po-
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu	(class eDevice.Setup_PodInterface) () lDevices.Setup_PodDevices.S () neDevice.Setup_PodInterface	in , Setupa .Setup	rsyncScompfingBrituse() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 ATI TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), pInterface 12 totalDropsCollected (Po- dces.SetupAlApi.Stream.DataBucket.Bucket attribute),
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod	in , Setup .Setup dDevi	rSyncScompfingPrints() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), plnterface 12 totalDropsCollected (Po- dees.SetupAllApi.Stream.DataBucket.Bucket attribute), 58
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod	in , Setup .Setup dDevi	rsyncScompfingPrints() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- ices.SetupAllApi.Stream.DataBucket.Bucket attribute), 58 HaranslateAll() (PodApi.Packets.Binary.PacketBinary
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setup .Setup dDevi	rSyncScompfingPrints() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- ices.SetupAilApi.Stream.DataBucket.Bucket attribute), 58 TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain(PodApi.Parame attribute), 48 StartCollecting()	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setup .Setup dDevi 3401F	**SyncScompfingPrinted**() (Po-dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 **TestConnection() (PodApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Podes.SetupAlApi.Stream.DataBucket.Bucket attribute), 58 **TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setup .Setup dDevi 3401F	**SyncScompfingBrifacs() (Po-dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 ATT TestConnection() (PodApi.Devices.BasicPodProtocol.Pod method), plnterface 12 totalDropsCollected (Podes.SetupAllApi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setupa .Setupa dDevi dDevi (Po- od),	**SyncScompfingBritase() (Po-dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 ATT TestConnection() (PodApi.Devices.BasicPodProtocol.Pod method), plnterface 12 totalDropsCollected (Podapi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream()	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setupa .Setupa dDevi dDevi (Po- od),	rsyncScompfingPrints() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), plnterface 12 totalDropsCollected (Po- ices.SetupAllApi.Stream.DataBucket.Bucket attribute), 58 RranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain(PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61	(class eDevice.Setup_PodInterface) E) IDevices.Setup_PodDevices.S E) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8 (ataBucket.Bucket methol (PodApi.Stream.DataHose.H	in , Setup .Setup dDevi 3401F (Po- od),	rSyncScompfingPrints() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 All TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- ices.SetupAlfApi.Stream.DataBucket.Bucket attribute), 58 FranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Packet.Packet
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain(PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.S (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8	in , Setupa .Setupa dDevi dDevi (Po- od),	**SyncScompfingBrites() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 ATI TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- ces.SetupAlApi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Packet.Packet method), 41
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain(PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack tribute), 33	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.Setup_PodInterface) (neDevice.Setup_PodInterface) up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8 (ataBucket.Bucket method (PodApi.Stream.DataHose.Hets.Binary5.PacketBinary5	in , Setupa .Setup dDevi dDevi dOse at-	rsyncCompfingDifue() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- des.SetupAlApi.Stream.DataBucket.Bucket attribute), 58 TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Packet.Packet method), 41 TranslateAll() (PodApi.Packets.Standard.PacketStandard
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack tribute), 33 Status() (PodAp	(class eDevice.Setup_PodInterface) E) IDevices.Setup_PodDevices.S E) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8 (ataBucket.Bucket methol (PodApi.Stream.DataHose.H	in , Setupa .Setup dDevi dDevi dOse at-	**SyncCompfligDifuse() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 **TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), **DInterface 12 totalDropsCollected (Po- des.SetupAdApi.Stream.DataBucket.Bucket attribute), 58 **ParanslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 41 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 44
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack tribute), 33 Status() (PodAp method), 37	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.Setup_PodInterface) (c) neDevice.Setup_PodInterface up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8 (ataBucket.Bucket metho (PodApi.Stream.DataHose.H ets.Binary5.PacketBinary5 i.Packets.Binary5.PacketBinar	in , Setupa .Setupa dDevi 3401F (Po- od), lose at- ury5	**SyncCompfligDifuce() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 AT TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- dess.SetupAdJApi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 41 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 44 TranslateBinaryTTLbyte() (Po-
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack tribute), 33 Status() (PodAp method), 37 stimulus (PodApi.Parame method), 37	(class eDevice.Setup_PodInterface) (c) lDevices.Setup_PodDevices.Setup_PodInterface) (neDevice.Setup_PodInterface) up.SetupAllDevices.Setup_Pod ters.Params8401HR.Params8 (ataBucket.Bucket method (PodApi.Stream.DataHose.Hets.Binary5.PacketBinary5	in , Setupa .Setupa dDevi 3401F (Po- od), lose at- ury5	**SyncCompfligDifuce() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 AT TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), bInterface 12 totalDropsCollected (Po- dees.SetupAdApi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 41 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 44 TranslateBinaryTTLbyte() (Po-
method), 87 SetupInterface Setup.SetupOn 86 SetupPODparameters((Setup.SetupAl method), 72 SetupPODparameters((Setup.SetupOn method), 87 SetupSaveFile() (Setu method), 72 ssGain (PodApi.Parame attribute), 48 StartCollecting() dApi.Stream.D 59 StartStream() method), 61 status (PodApi.Pack tribute), 33 Status() (PodAp method), 37	(class eDevice.Setup_PodInterface) (c) IDevices.Setup_PodDevices.Setup_PodInterface) IDevices.Setup_PodInterface IDevice.Setup_PodInterface IDEVICE.Setup_PodInterface IDEVICE.Setup_PodInterface IDEVICES.Setup_PodInterface IDEV	in , Setupa .Setupa dDevi 3401F (Po- od), lose at- ury5	**SyncCompfligDifuce() (Po- dApi.Devices.PodDevice_8480SC.Pod8480SC static method), 26 systemID (PodApi.Parameters.Params8229.Params8229 attribute), 46 AT TestConnection() (Po- dApi.Devices.BasicPodProtocol.Pod method), oInterface 12 totalDropsCollected (Po- dess.SetupAdJapi.Stream.DataBucket.Bucket attribute), 58 #*TranslateAll() (PodApi.Packets.Binary.PacketBinary method), 29 TranslateAll() (PodApi.Packets.Binary4.PacketBinary4 method), 32 TranslateAll() (PodApi.Packets.Binary5.PacketBinary5 method), 37 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 41 TranslateAll() (PodApi.Packets.Standard.PacketStandard method), 44 TranslateBinaryTTLbyte() (Po-

```
ttl (PodApi.Packets.Binary4.PacketBinary4 attribute), WriteRead()
                                                                     (PodApi.Devices.BasicPodProtocol.Pod
                                                              method), 12
Ttl() (PodApi.Packets.Binary4.PacketBinary4 method),
         32
TtlConfigBits()
        dApi.Devices.PodDevice 8480SC.Pod8480SC
        static method), 26
ttlPUllups (PodApi.Parameters.Params8480SC.Params8480SC
         attribute), 50
\verb+ttlSetup+ (PodApi.Parameters.Params 8480 SC.Params 8480 SC)
        attribute), 50
TwosComplement()
                        (PodApi.Packets.Packet.Packet
        static method), 42
U
U16() (PodApi.Commands.PodCommands.CommandSet
        static method), 4
U32() (PodApi.Commands.PodCommands.CommandSet
        static method), 4
U8()
      (PodApi.Commands.PodCommands.CommandSet
        static method), 4
UnpackAll()
                 (PodApi.Packets.Binary.PacketBinary
        method), 29
UnpackAll() (PodApi.Packets.Binary4.PacketBinary4
        method), 33
UnpackAll() (PodApi.Packets.Binary5.PacketBinary5
        method), 37
UnpackAll() (PodApi.Packets.Packet.Packet method),
UnpackAll() (PodApi.Packets.Standard.PacketStandard
        method), 44
UserInput (class in Setup.Inputs.GetUserInput), 66
V
ValidateCommand()
                                               (Po-
        dApi. Commands. Pod Commands. Command Set\\
        method), 4
ValidateParams() (Setup.SetupOneDevice.Setup PodInterface.SetupInterface
        method), 88
Valve (class in PodApi.Stream.DeviceValve), 63
W
week (PodApi.Parameters.Params8229.Params8229 at-
        tribute), 47
Write() (PodApi.Devices.SerialPorts.SerialComm.PortIO
        method), 9
WritePacket() (PodApi.Devices.BasicPodProtocol.Pod
        method), 12
WritePacket() (PodApi.Devices.PodDevice_8229.Pod8229
        method), 18
WritePacket() (PodApi.Devices.PodDevice_8401HR.Pod8401HR
        method), 23
WritePacket() (PodApi.Devices.PodDevice_8480SC.Pod8480SC
        method), 27
```