3-Space API

Sensor Documentation here:
https://yostlabs.com/wp/wp-content/uploads/pdf/3-Space-Sensor-Users-Manual-1.pdf

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 threespace_api Namespace Reference

Classes

- class _TSBase

  Base class for sensors.
- class _TSSensor

  Standard base for all other sensors.
- class Broadcaster

  The Broadcaster class allows the programmer to easily enable asynchronous commands for a cluster of sensors all at once.
- class TSBTSensor

  The TSBTSensor class is an interface layer for 3-Space Sensor Bluetooth units that communicate through the USB port.
- class TSDLSensor

  The TSDLSensor class is an interface layer for 3-Space Sensor Data-logging units that communicate through the USB port.
- class TSDongle

  The TSDongle class is an interface layer for 3-Space Sensor Dongle units that communicate through the USB port and acts as an autonomous object that handles communication between the dongle and wireless 3-Space Sensor units.
- class TSEMSensor

  The TSEMSensor class is an interface layer for 3-Space Sensor Embedded units that communicate through the USB port or RS-232 port.
- class TSLXSensor

  The TSLXSensor class is an interface layer for 3-Space Sensor LX units that communicate through the USB port.
- class TSNANOSensor

  The TSNANOSensor class is an interface layer for 3-Space Sensor NANO units that communicate through the USB port.
- class TSUSBSensor

  The TSUSBSensor class is an interface layer for 3-Space Sensor USB units that communicate through the USB port.
- class TSWLSensor

  The TSWLSensor class is an interface layer for 3-Space Sensor Wireless units that communicate through the USB port or a 3-Space Sensor Dongle.

Functions

- def **generateAxisDirections** (axis_order, neg_x=False, neg_y=False, neg_z=False)
- def **getDefaultCreateDeviceBaudRate** ()
- def **getSystemWirelessRetries** ()
- def **padProtocolHeader69** (header_data, sys_timestamp)
- def **padProtocolHeader71** (header_data)
- def **padProtocolHeader85** (header_data, sys_timestamp)
- def **padProtocolHeader87** (header_data)
- def **parseAxisDirections** (axis_byte)
- def **setDefaultCreateDeviceBaudRate** (new_baudrate)
- def **setSystemWirelessRetries** (retries)

Variables

- global_broadcaster = Broadcaster()

  A global object meant for communicating with all USB/RS232 connected sensors at once.

3.1.1 Detailed Description

```
This module is an API module for ThreeSpace devices.

   The ThreeSpace API module is a collection of classes, functions, structures,
   and static variables use exclusivly for ThreeSpace devices. This module can
   be used with a system running Python 3 and newer.

   For all numbered Commands reference the Sensor Manual for details.
   https://yostlabs.com/wp/wp-content/uploads/pdf/3-Space-Sensor-Users-Manual-1.pdf
```

3.2 threespace_utils Namespace Reference

Functions

- def **checkSoftwareVersionFromPort** (serial_port)
- def **pyTryPort** (port_name, conn)
- def **tryPort** (port_name, use_subprocess=False)

3.2.1 Detailed Description

```
This module is a utility module used in the ThreeSpace API.

   The ThreeSpace Utils module is a collection of functions, structures, and
   static variables to be use exclusivly with the ThreeSpace API module to find
   available ThreeSpace devices on the host system and information on them.
   This module can be used with a system running 3 and newer
```

3.3 win32_threespace_utils Namespace Reference

Classes

- class BLUETOOTH_ADDRESS

  Bluetooth structure.
- class BLUETOOTH_DEVICE_INFO

  Bluetooth structure.
- class BLUETOOTH_DEVICE_SEARCH_PARAMS

  Bluetooth structure.
- class GUID

  COM Port stucture.
- class SP_DEVICE_INTERFACE_DATA

  COM Port stucture.
- class SP_DEVINFO_DATA

  COM Port stucture.
- class SYSTEMTIME

  Time structure.

Functions

- def getComPorts (filter=TSS_FIND_ALL)
- def getDeviceInfoFromComPort (port_name, poll_device=True)
- def **toLong** (number, base=None)

Variables

- **argtypes**
- **errcheck**
- **GUID_DEVINTERFACE_COMPORT** = GUID(toLong(0x86E0D1E0), 0x8089, 0x11D0, (BYTE * 8)(0x9C, 0xE4, 0x08, 0x00, 0x3E, 0x30, 0x1F, 0x73))
- **GUID_DEVINTERFACE_SERENUM_BUS_ENUMERATOR** = GUID(toLong(0x4D36E978), 0xE325, 0x11↩CE, (BYTE * 8)(0xBF, 0xC1, 0x08, 0x00, 0x2B, 0xE1, 0x03, 0x18))
- **PSP_DEVICE_INTERFACE_DATA** = ctypes.POINTER(SP_DEVICE_INTERFACE_DATA)
- **PSP_DEVICE_INTERFACE_DETAIL_DATA** = ctypes.c_void_p
- PSP_DEVINFO_DATA = ctypes.POINTER(SP_DEVINFO_DATA)

  Structures/Class Pointers ###.
- **RegCloseKey** = advapi32.RegCloseKey
- **RegQueryValueEx** = advapi32.RegQueryValueExA
- **restype**
- **SetupDiDestroyDeviceInfoList** = setupapi.SetupDiDestroyDeviceInfoList
- **SetupDiEnumDeviceInterfaces** = setupapi.SetupDiEnumDeviceInterfaces
- **SetupDiGetClassDevs** = setupapi.SetupDiGetClassDevsA
- **SetupDiGetDeviceInterfaceDetail** = setupapi.SetupDiGetDeviceInterfaceDetailA
- **SetupDiGetDeviceRegistryProperty** = setupapi.SetupDiGetDeviceRegistryPropertyA
- **SetupDiOpenDevRegKey** = setupapi.SetupDiOpenDevRegKey

3.3.1 Detailed Description

```
This module is a utility module for Windows.

    The Win32 ThreeSpace Utils module is a collection of classes, functions,
    structures, and static variables use exclusivly for Windows. All functions
    in this module are used to scan for available ThreeSpace devices on the host
    system and information on them. This module can be used with a system
    running Python 3 and newer.
```

3.3.2 Function Documentation

3.3.2.1 getComPorts()

```
def win32_threespace_utils.getComPorts (
              filter = TSS_FIND_ALL )
```

```
Queries the system for all available serial COM ports and returns a list
    of them.

    Args:
        filter: An interger denoting a flag of what 3-Space Sensors device
            type to be found (default is TSS_FIND_ALL)

    Returns:
        A list of all known serial COM ports. Each element of the list is a
            tuple formatted as such:
                (COM_PORT_NAME, FRIENDLY_NAME, YEI_TECH_DEVICE_TYPE)
        Note:
            YEI_TECH_DEVICE_TYPE will be an empty string if the port's
                driver's vendor and product IDs do not match any known YEI
                Techology products.
            Possible YEI_TECH_DEVICE_TYPE strings are:
                '???' - Unknown
                'BTL' - Bootloader (No Firmware)
                'USB' - USB
                'DNG' - Dongle
                'WL' - Wireless
                'EM' - Embedded
                'DL' - Data-logging
                'BT' - Bluetooth
```

3.3.2.2 getDeviceInfoFromComPort()

```
def win32_threespace_utils.getDeviceInfoFromComPort (
              port_name,
              poll_device = True )
```

```
Analyzes a serial COM port of a 3-Space Sensor and returns details about
   the device.

   Args:
       port_name: A string representing the name of the serial COM port to
           analyze.
       poll_device: An optional boolean that controls whether the named COM
           port is written to and queried for information about the device.
           If this value is True, please take caution as the COM port's
           device will be written to and may produce undesired effects if
           the device is unknown or not a 3-Space Sensor (default is True)

   Returns:
       A list of 5 values describing various details about the COM port's
       device:
           Friendly name,
           3-Space Type,
           3-Space ID,
           3-Space Firmware Version String,
           3-Space Hardware Version String,
           isInBootloader

   Raises:
       No explicit exceptions are raised.
```

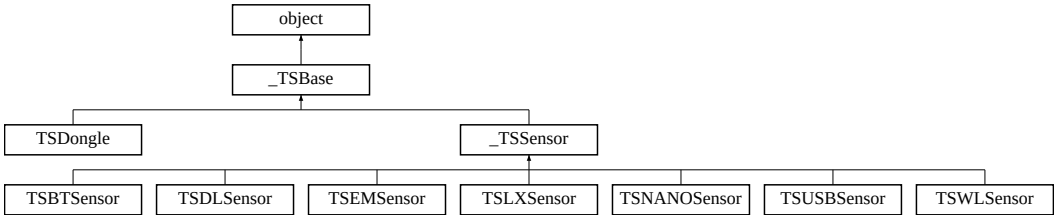
Chapter 4

Class Documentation

4.1 _TSBase Class Reference

Base class for sensors.

Inheritance diagram for _TSBase:



Public Member Functions

- def **__init__** (self, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def **__repr__** (self)
- def **__str__** (self)
- def **close** (self)
- def commitSettings (self, timestamp=False)

  225(0xe1)
- def **f7WriteRead** (self, command, input_list=None)
- def **f9WriteRead** (self, command, input_list=None)
- def getFirmwareVersionString (self, timestamp=False)

  223(0xdf)
- def getHardwareVersionString (self, timestamp=False)

  230(0xe6)
- def getJoystickAndMousePresentRemoved (self, timestamp=False)

  254(0xfe)
- def getLEDColor (self, timestamp=False)

  239(0xef)
- def getLEDMode (self, timestamp=False)

  200(0xc8)
- def getSerialNumber (self, timestamp=False)

237(0xed)

- def **isConnected** (self, try_reconnect=False)
- def **reconnect** (self)
- def setJoystickAndMousePresentRemoved (self, joystick, mouse, timestamp=False)

  253(0xfd)
- def setLEDColor (self, rgb, timestamp=False)

  238(0xee)
- def setLEDMode (self, mode, timestamp=False)

  196(0xc4)
- def startStreaming (self)

  86(0x56)
- def stopStreaming (self)

  85(0x55)
- def updateCurrentTimestamp (self, time, timestamp=False)

  95(0x5f)

Public Attributes

- **baudrate**
- **data_loop**
- **header_idx_lst**
- **protocol_args**
- **read_dict**
- **read_lock**
- **read_queue**
- **read_thread**
- **record_data**
- **serial_number_hex**
- **serial_port**
- **stream_data**
- **stream_last_data**
- **stream_parse**
- **stream_slot_cmds**
- **stream_timing**
- **timestamp_mode**

Static Public Attributes

- dictionary **command_dict**
- def **writeRead** = f9WriteRead

4.1.1 Detailed Description

Base class for sensors.

should not be used directly.

4.1.2 Member Data Documentation

4.1.2.1 command_dict

```
dictionary command_dict  [static]
```

Initial value:

```
=  {
        'checkLongCommands': (0x19, 1, '>B', 0, None, 1),
        'startStreaming': (0x55, 0, None, 0, None, 1),
        'stopStreaming': (0x56, 0, None, 0, None, 1),
        'updateCurrentTimestamp': (0x5f, 0, None, 4, '>I', 1),
        'setLEDMode': (0xc4, 0, None, 1, '>B', 1),
        'getLEDMode': (0xc8, 1, '>B', 0, None, 1),
        '_setWiredResponseHeaderBitfield': (0xdd, 0, None, 4, '>I', 1),
        '_getWiredResponseHeaderBitfield': (0xde, 4, '>I', 0, None, 1),
        'getFirmwareVersionString': (0xdf, 12, '>12s', 0, None, 1),
        'commitSettings': (0xe1, 0, None, 0, None, 1),
        'softwareReset': (0xe2, 0, None, 0, None, 1),
        'getHardwareVersionString': (0xe6, 32, '>32s', 0, None, 1),
        'getSerialNumber': (0xed, 4, '>I', 0, None, 1),
        'setLEDColor': (0xee, 0, None, 12, '>fff', 1),
        'getLEDColor': (0xef, 12, '>fff', 0, None, 1),
        'setJoystickAndMousePresentRemoved': (0xfd, 0, None, 2, '>BB', 1),
        'getJoystickAndMousePresentRemoved': (0xfe, 2, '>B', 0, None, 1),
        'null': (0xff, 0, None, 0, None, 1)
    }
```

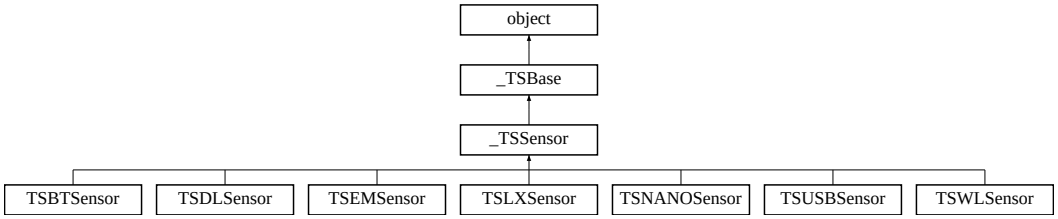
The documentation for this class was generated from the following file:

- threespace_api.py

4.2 _TSSensor Class Reference

Standard base for all other sensors.

Inheritance diagram for _TSSensor:



Public Member Functions

- def **__init__** (self, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def beginGyroscopeAutoCalibration (self, timestamp=False)

  165(0xa5)
- def clearOrthoCalibrationData (self, timestamp=False)

  175(0xaf)
- def **clearRecordingData** (self)
- def disableAxis (self, hid_type, config_axis)
- def disableButton (self, hid_type, button_idx)
- def getAccelerometerCalibrationCoefficients (self, timestamp=False)

  163(0xa3)
- def getAccelerometerEnabledState (self, timestamp=False)

141(0x8d)

- def getAccelerometerRange (self, timestamp=False)

  148(0x94)
- def getAccelerometerReferenceVector (self, timestamp=False)

  134(0x86)
- def getAccelerometerResistanceThreshold (self, timestamp=False)

  158(0x9e)
- def getAccelerometerTrustValues (self, timestamp=False)

  130(0x82)
- def getAllCorrectedComponentSensorData (self, timestamp=False)

  37(0x25)
- def getAllNormalizedComponentSensorData (self, timestamp=False)

  32(0x20)
- def getAllRawComponentSensorData (self, timestamp=False)

  64(0x40)
- def getAxisDirections (self, timestamp=False)

  143(0x8f)
- def getCalibrationMode (self, timestamp=False)

  170(0xaa)
- def getCompassCalibrationCoefficients (self, timestamp=False)

  162(0xa2)
- def getCompassEnabledState (self, timestamp=False)

  142(0x8e)
- def getCompassRange (self, timestamp=False)

  155(0x9b)
- def getCompassReferenceVector (self, timestamp=False)

  133(0x85)
- def getCompassTrustValues (self, timestamp=False)

  131(0x83)
- def getConfidenceFactor (self, timestamp=False)

  45(0x2d)
- def getControlData (self, control_class, control_index, handler_index, timestamp=False)

  247(0xf7)
- def getControlMode (self, control_class, control_index, timestamp=False)

  246(0xf6)
- def getCorrectedAccelerometerVector (self, timestamp=False)

  39(0x27)
- def getCorrectedCompassVector (self, timestamp=False)

  40(0x28)
- def getCorrectedGyroRate (self, timestamp=False)

  38(0x26)
- def getCorrectedLinearAccelerationInGlobalSpace (self, timestamp=False)

  41(0x29)
- def getCurrentUpdateRate (self, timestamp=False)

  132(0x84)
- def getDesiredUpdateRate (self, timestamp=False)

  146(0x92)
- def getDifferenceQuaternion (self, timestamp=False)

  5(0x05)
- def getEulerAngleDecompositionOrder (self, timestamp=False)

  156(0x9c)

- def getFilterMode (self, timestamp=False)

  152(0x98)
- def getGyroscopeCalibrationCoefficients (self, timestamp=False)

  164(0xa4)
- def getGyroscopeEnabledState (self, timestamp=False)

  140(0x8c)
- def getGyroscopeRange (self, timestamp=False)

  154(0x9a)
- def getJoystickEnabled (self, timestamp=False)

  242(0xf2)
- def **getLatestStreamData** (self, timeout)
- def getMagnetoresistiveThreshold (self, timestamp=False)

  157(0x9d)
- def getMouseAbsoluteRelativeMode (self, timestamp=False)

  252(0xfc)
- def getMouseEnabled (self, timestamp=False)

  243(0xf3)
- def getNormalizedAccelerometerVector (self, timestamp=False)

  34(0x22)
- def getNormalizedCompassVector (self, timestamp=False)

  35(0x23)
- def getNormalizedGyroRate (self, timestamp=False)

  33(0x21)
- def getOffsetOrientationAsQuaternion (self, timestamp=False)

  159(0x9f)
- def getOrthoCalibrationDataPoint (self, type, index, timestamp=False)

  173(0xad)
- def getOversampleRate (self, timestamp=False)

  144(0x90)
- def getRawAccelerometerData (self, timestamp=False)

  66(0x42)
- def getRawCompassData (self, timestamp=False)

  67(0x43)
- def getRawGyroscopeRate (self, timestamp=False)

  65(0x41)
- def getRunningAverageMode (self, timestamp=False)

  153(0x99)
- def getRunningAveragePercent (self, timestamp=False)

  145(0x91)
- def getSleepMode (self, timestamp=False)

  228(0xe4)
- def getStreamingBatch (self, timestamp=False)

  84(0x54)
- def getStreamingSlots (self)

  81(0x51)
- def getStreamingTiming (self, timestamp=False)

  83(0x53)
- def getTareAsQuaternion (self, timestamp=False)

  128(0x80)
- def getTareAsRotationMatrix (self, timestamp=False)

  129(0x81)

- def getTaredOrientationAsAxisAngle (self, timestamp=False)

  3(0x03)
- def getTaredOrientationAsEulerAngles (self, timestamp=False)

  1(0x01)
- def getTaredOrientationAsQuaternion (self, timestamp=False)

  generated functions USB and WL_ and EM_ and DL_ and BT_ 0(0x00)
- def getTaredOrientationAsRotationMatrix (self, timestamp=False)

  2(0x02)
- def getTaredOrientationAsTwoVector (self, timestamp=False)

  4(0x04)
- def getTaredTwoVectorInSensorFrame (self, timestamp=False)

  11(0x0b)
- def getTemperatureC (self, timestamp=False)

  43(0x2b)
- def getTemperatureF (self, timestamp=False)

  44(0x2c)
- def getUntaredOrientationAsAxisAngle (self, timestamp=False)

  9(0x09)
- def getUntaredOrientationAsEulerAngles (self, timestamp=False)

  7(0x07)
- def getUntaredOrientationAsQuaternion (self, timestamp=False)

  6(0x06)
- def getUntaredOrientationAsRotationMatrix (self, timestamp=False)

  8(0x08)
- def getUntaredOrientationAsTwoVector (self, timestamp=False)

  10(0x0a)
- def getUntaredTwoVectorInSensorFrame (self, timestamp=False)

  12(0x0c)
- def offsetWithCurrentOrientation (self, timestamp=False)

  19(0x13)
- def offsetWithQuaternion (self, quaternion, timestamp=False)

  21(0x15)
- def performOrthoCalibration (self, timestamp=False)

  174(0xae)
- def **queueWriteRead** (self, rtn_dict, rtn_key, retries, command, input_list=None)
- def resetBaseOffset (self, timestamp=False)

  20(0x14)
- def resetKalmanFilter (self, timestamp=False)

  120(0x78)
- def setAccelerometerCalibrationCoefficients (self, matrix, bias, timestamp=False)

  161(0xa1)
- def setAccelerometerEnabled (self, enabled, timestamp=False)

  108(0x6c)
- def setAccelerometerRange (self, accelerometer_range_setting, timestamp=False)

  121(0x79)
- def setAccelerometerReferenceVector (self, reference_vector, timestamp=False)

  119(0x77)
- def setAccelerometerResistanceThreshold (self, threshold, lockout_decay, timestamp=False)

  18(0x12)
- def setAxisDirections (self, axis_direction_byte, timestamp=False)

  116(0x74)

- def setBaseOffsetWithCurrentOrientation (self, timestamp=False)

  22(0x16)
- def setCalibrationMode (self, mode, timestamp=False)

  169(0xa9)
- def setCompassCalibrationCoefficients (self, matrix, bias, timestamp=False)

  160(0xa0)
- def setCompassEnabled (self, enabled, timestamp=False)

  109(0x6d)
- def setCompassRange (self, compass_range_setting, timestamp=False)

  126(0x7e)
- def setCompassReferenceVector (self, reference_vector, timestamp=False)

  118(0x76)
- def setConfidenceAccelerometerTrustValues (self, min_trust_value, max_trust_value, timestamp=False)

  100(0x64)
- def setConfidenceCompassTrustValues (self, min_trust_value, max_trust_value, timestamp=False)

  102(0x66)
- def setControlData (self, control_class, control_index, data_point_index, data_point, timestamp=False)

  245(0xf5)
- def setControlMode (self, control_class, control_index, handler_index, timestamp=False)

  244(0xf4)
- def setDesiredUpdateRate (self, update_rate, timestamp=False)

  103(0x67)
- def setEulerAngleDecompositionOrder (self, angle_order, timestamp=False)

  16(0x10)
- def setFilterMode (self, mode, timestamp=False)

  123(0x7b)
- def setGlobalAxis (self, hid_type, config_axis, local_axis, global_axis, deadzone, scale, power)
- def setGyroscopeCalibrationCoefficients (self, matrix, bias, timestamp=False)

  166(0xa6)
- def setGyroscopeEnabled (self, enabled, timestamp=False)

  107(0x6b)
- def setGyroscopeRange (self, gyroscope_range_setting, timestamp=False)

  125(0x7d)
- def setJoystickEnabled (self, enabled, timestamp=False)

  240(0xf0)
- def setMagnetoresistiveThreshold (self, threshold, trust_frames, lockout_decay, perturbation_detection_↩ value, timestamp=False)

  17(0x11)
- def setMouseAbsoluteRelativeMode (self, mode, timestamp=False)

  251(0xfb)
- def setMouseEnabled (self, enabled, timestamp=False)

  241(0xf1)
- def **setNewDataCallBack** (self, callback)
- def setOrientationButton (self, hid_type, button_idx, local_axis, global_axis, max_dist)
- def setOrthoCalibrationDataPointFromCurrentOrientation (self, timestamp=False)

  171(0xab)
- def setOrthoCalibrationDataPointFromVector (self, type, index, vector, timestamp=False)

  172(0xac)
- def setOversampleRate (self, samples_per_iteration, timestamp=False)

  106(0x6a)
- def setPhysicalButton (self, hid_type, button_idx, button_bind)

- def setReferenceVectorMode (self, mode, timestamp=False)

  105(0x69)
- def setRunningAverageMode (self, mode, timestamp=False)

  124(0x7c)
- def setRunningAveragePercent (self, running_average_percent, timestamp=False)

  117(0x75)
- def setScreenPointAxis (self, hid_type, config_axis, dist_from_screen, dist_on_axis, collision_component, sensor_dir, button_halt)
- def setShakeButton (self, hid_type, button_idx, threshold)
- def setSleepMode (self, mode, timestamp=False)

  227(0xe3)
- def setStaticAccelerometerTrustValue (self, trust_value, timestamp=False)

  99(0x63)
- def setStaticCompassTrustValue (self, trust_value, timestamp=False)

  101(0x65)
- def setStreamingSlots (self, slot0='null', slot1='null', slot2='null', slot3='null', slot4='null', slot5='null', slot6='null', slot7='null')

  80(0x50)
- def setStreamingTiming (self, interval, duration, delay, timestamp=False)

  82(0x52)
- def setupSimpleJoystick (self, deadzone, scale, power, shake_threshold, max_dist)
- def setupSimpleLightgun (self, diagonal_size, dist_from_screen, aspect_ratio, is_relative=True)
- def setupSimpleMouse (self, diagonal_size, dist_from_screen, aspect_ratio, is_relative=True)
- def **startRecordingData** (self)
- def startStreaming (self, start_record=False)

  86(0x56)
- def **stopRecordingData** (self)
- def stopStreaming (self)

  85(0x55)
- def tareWithCurrentOrientation (self, timestamp=False)

  96(0x60)
- def tareWithQuaternion (self, quaternion, timestamp=False)

  97(0x61)
- def tareWithRotationMatrix (self, rotation_matrix, timestamp=False)

  98(0x62)

Public Attributes

- **baudrate**
- **callback_func**
- **latest_lock**
- **new_data**
- **protocol_args**
- **record_data**
- **stream_data**
- **stream_last_data**
- **stream_parse**
- **stream_slot_cmds**
- **stream_timing**
- **timestamp_mode**

Static Public Attributes

- **command_dict** = _TSBase.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.2.1 Detailed Description

Standard base for all other sensors.

4.2.2 Member Function Documentation

4.2.2.1 disableAxis()

```
def disableAxis (
              self,
              hid_type,
              config_axis )


 Disables an axis on the passed in device.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param config_axis: A string whose value may be either 'X' or 'Y'
        for a mouse or 'X', 'Y', or 'Z' for a joystick. This string
        defines what axis of the device is to be configured.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.2 disableButton()

```
def disableButton (
              self,
              hid_type,
              button_idx )


 Disables a button on the passed in emulated device.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param button_idx: An integer whose value defines which button on
        the emulated device to configure. Default range is 0 through 7.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.3 setGlobalAxis()

```
def setGlobalAxis (
              self,
              hid_type,
              config_axis,
              local_axis,
              global_axis,
              deadzone,
              scale,
              power )
```

```
 Sets an axis of the desired emulated input device as a 'Global Axis'
    style axis. Axis operating under this style use a reference vector
    and a consitent local vector to determine the state of the device's
    axis. As the local vector rotates, it is projected onto the global
    vector. Once the distance of that projection on the global vector
    exceeds the inputted "deadzone", the device will begin tranmitting
    non-zero values for the device's desired axis.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param config_axis: A string whose value may be either 'X' or 'Y'
        for a mouse or 'X', 'Y', or 'Z' for a joystick. This string
        defines what axis of the device is to be configured.

    @param local_axis: A list of 3 Floats whose value is a normalized
        Vector3. This vector represents the sensor's local vector to
        track.

    @param global_axis: A list of 3 Floats whose value is a normalized
        Vector3. This vector represents the global vector to project the
        local vector onto (should be orthoginal to the local vector).

    @param deadzone: A float that defines the minimum distance necessary
        for the device's axis to read a non-zero value.

    @param scale: A float that defines the linear scale for the values
        being returned for the axis.

    @param power: A float whose value is an exponental power used to
        further modify data being returned from the sensor.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.4 setOrientationButton()

```
def setOrientationButton (
              self,
              hid_type,
              button_idx,
              local_axis,
              global_axis,
              max_dist )
```

```
 Sets up a device's button such that it is 'pressed' when a reference
    vector aligns itself with a local vector.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param button_idx: An integer whose value defines which button on
        the emulated device to configure. Default range is 0 through 7.

    @param local_axis: A list of 3 floats whose value represents a
        normalized Vector3. This vector represents the sensor's local
        vector to track.

    @param global_axis: A list of 3 floats whose value is a normalized
        Vector3. This vector represents the global vector to move the
        local vector towards for "pressing" (should not be colinear to
        the local vector).

    @param max_dist: A float whose value defines how close the local
        vector's orientation must be to the global vector for the button
        to be 'pressed'.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.5 setPhysicalButton()

```
def setPhysicalButton (
              self,
              hid_type,
              button_idx,
              button_bind )
```

```
 Binds a sensor's physical button to an emulated device's button.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param button_idx: An integer whose value defines which button on
        the emulated device to configure. Default range is 0 through 7.

    @param button_bind: An integer whose value defines which physical
        button to bind to the emulated device's button to as defined by
        button_idx, either TSS_BUTTON_LEFT or TSS_BUTTON_RIGHT.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.6 setScreenPointAxis()

```
def setScreenPointAxis (
              self,
              hid_type,
              config_axis,
              dist_from_screen,
              dist_on_axis,
              collision_component,
              sensor_dir,
              button_halt )
```

```
 Sets an axis of the desired emulated input device as a 'Screen Point
    Axis' style axis. An axis operating under this style projects a
    vector along the sensor's direction vector into a mathmatical plane.
    The collision point on the plane is then used to determine what the
    device's axis's current value is. The direction vector is rotated
    based on the orientation of the sensor.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param config_axis: A string whose value may be either 'X' or 'Y'
        for a mouse or 'X', 'Y', or 'Z' for a joystick. This string
        defines what axis of the device is to be configured.

    @param dist_from_screen: A float whose value is the real world
        distance the sensor is from the user's screen. Must be the same
        units as dist_on_axis.

    @param dist_on_axis: A float whose value is the real world length of
        the axis along the user's screen (width of screen for x-axis,
        height of screen for y-axis). Must be the same units as
        dist_from_screen.

    @param collision_component: A string whose value may be 'X', 'Y', or
        'Z'. This string defines what component of the look vector's
        collision point on the virtual plane to use for manipulating the
        device's axis.

    @param sensor_dir: A string whose value may be 'X', 'Y', or 'Z'.
        This string defines which of the sensor's local axis to use for
        creating the vector to collide with the virtual plane.

    @param button_halt: A float whose value is a pause time in
        milliseconds. When a button is pressed on the emulated device,
        transmission of changes to the axis is paused for the inputted
        amount of time to prevent undesired motion detection when
        pressing buttons.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.7 setShakeButton()

```
def setShakeButton (
              self,
              hid_type,
              button_idx,
              threshold )
```

```
 Sets up an emulated device's button such that it is 'pressed' when
    the sensor is shaken.

    @param hid_type: An integer whose value defines whether the device
        in question is a TSS_JOYSTICK or TSS_MOUSE.

    @param button_idx: An integer whose value defines which button on
        the emulated device to configure. Default range is 0 through 7.

    @param threshold: A float whose value defines how many Gs of force
        must be experienced by the sensor before the button is
        'pressed'.

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.8 setupSimpleJoystick()

```
def setupSimpleJoystick (
              self,
              deadzone,
              scale,
              power,
              shake_threshold,
              max_dist )
```

```
 Creates a simple emulated joystick device using the features of the
    sensor. The left and right physical buttons on the sensor act as
    buttons 0 and 1 for the joystick. Button 2 is a shake button.
    Buttons 3 and 4 are pressed when the sensor is rotated +-90 degrees
    on the Z-axis. Rotations on the sensor's Y and X axis correspond to
    movements on the joystick's X and Y axis.

    @param deadzone: A float that defines the minimum distance necessary
        for the device's axis to read a non-zero value.

    @param scale: A float that defines the linear scale for the values
        being returned for the axis.

    @param power:A float whose value is an exponental power used to
        further modify data being returned from the sensor.

    @param shake_threshold: A float whose value defines how many Gs of
        force must be experienced by the sensor before the button 2 is
        'pressed'.

    @param max_dist: A float whose value defines how close the local
        vector's orientation must  be to the global vector for buttons 3
        and 4 are "pressed".

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.9 setupSimpleLightgun()

```
def setupSimpleLightgun (
              self,
              diagonal_size,
              dist_from_screen,
              aspect_ratio,
              is_relative = True )
```

```
 Creates a simple emulated mouse based lightgun device using the
    features of the sensor. Left button of the sensor emulates the
    mouse's left button. Shaking the sensor emulates the mouse's right
    button. This configuration uses the sensor as a pointing device with
    the front of the device facing forward the screen will move the
    mouse cursor.

    @param diagonal_size: A float whose value is the real world diagonal
        size of the user's screen.

    @param dist_from_screen: A float whose value is the real world
        distance the sensor is from the user's screen. Must be the same
        units as diagonal_size.
```

```
    @param aspect_ratio: A float whose value is the real world aspect
        ratio of the user's screen.

    @param is_relative: A boolean whose value expresses whether the
        mouse is to operate in relative mode (True) or absolute mode
        (False).

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

4.2.2.10 setupSimpleMouse()

```
def setupSimpleMouse (
              self,
              diagonal_size,
              dist_from_screen,
              aspect_ratio,
              is_relative = True )
```

```
 Creates a simple emulated mouse device using the features of the
    sensor. Left button and right button emulate the mouse's left and
    right buttons respectivly and using the sensor as a pointing device
    with the front of the device facing towards the screen will move the
    mouse cursor.

    @param diagonal_size: A float whose value is the real world diagonal
        size of the user's screen.

    @param dist_from_screen: A float whose value is the real world
        distance the sensor is from the user's screen. Must be the same
        units as diagonal_size.

    @param aspect_ratio: A float whose value is the real world aspect
        ratio of the user's screen.

    @param is_relative: A boolean whose value expresses whether the
        mouse is to operate in relative mode (True) or absolute mode
        (False).

    @return: True if the command was successfuly written to the device.
        False if the command was not written.
```

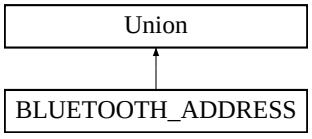
The documentation for this class was generated from the following file:

- threespace_api.py

4.3 BLUETOOTH_ADDRESS Class Reference

Bluetooth structure.

Inheritance diagram for BLUETOOTH_ADDRESS:



Public Member Functions

- def **__eq__** (self, other)
- def **__repr__** (self)
- def **__str__** (self)

4.3.1 Detailed Description

Bluetooth structure.

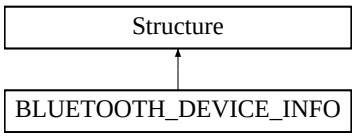
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.4 BLUETOOTH_DEVICE_INFO Class Reference

Bluetooth structure.

Inheritance diagram for BLUETOOTH_DEVICE_INFO:



Public Member Functions

- def **__str__** (self)

Public Attributes

- **fAuthenticated**
- **fConnected**
- **fRemembered**

4.4.1 Detailed Description

Bluetooth structure.

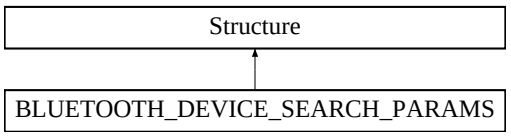
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.5 BLUETOOTH_DEVICE_SEARCH_PARAMS Class Reference

Bluetooth structure.

Inheritance diagram for BLUETOOTH_DEVICE_SEARCH_PARAMS:



4.5.1 Detailed Description

Bluetooth structure.

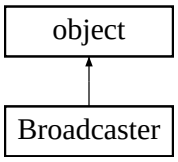
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.6 Broadcaster Class Reference

The Broadcaster class allows the programmer to easily enable asynchronous commands for a cluster of sensors all at once.

Inheritance diagram for Broadcaster:



Public Member Functions

- def **__init__** (self)
- def **broadcastMethod** (self, method, default=None, args=[ ], filter=None, callback_func=None)
- def **debugPrint** (self, broadcast_dict)
- def **getStreamingSlots** (self, filter=None, callback_func=None)
- def **sequentialWriteRead** (self, command, input_list=None, filter=None)
- def **setRetries** (self, retries=10)
- def **setStreamingSlots** (self, slot0='null', slot1='null', slot2='null', slot3='null', slot4='null', slot5='null', slot6='null', slot7='null', filter=None, callback_func=None)
- def **setStreamingTiming** (self, interval, duration, delay, delay_offset, filter=None, callback_func=None)
- def **startRecordingData** (self, filter=None, callback_func=None)
- def **startStreaming** (self, record_data=False, filter=None, callback_func=None)
- def **stopRecordingData** (self, filter=None, callback_func=None)
- def **stopStreaming** (self, filter=None, callback_func=None)
- def **writeRead** (self, command, input_list=None, filter=None)

Public Attributes

- **retries**

4.6.1 Detailed Description

The Broadcaster class allows the programmer to easily enable asynchronous commands for a cluster of sensors all at once.

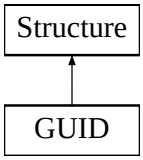
The documentation for this class was generated from the following file:

- threespace_api.py

4.7 GUID Class Reference

COM Port stucture.

Inheritance diagram for GUID:



Public Member Functions

- def **__str__** (self)

4.7.1 Detailed Description

COM Port stucture.

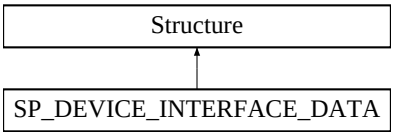
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.8 SP_DEVICE_INTERFACE_DATA Class Reference

COM Port stucture.

Inheritance diagram for SP_DEVICE_INTERFACE_DATA:



Public Member Functions

- def **__str__** (self)

4.8.1 Detailed Description

COM Port stucture.

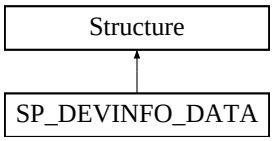
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.9 SP_DEVINFO_DATA Class Reference

COM Port stucture.

Inheritance diagram for SP_DEVINFO_DATA:



Public Member Functions

- def **__str__** (self)

4.9.1 Detailed Description

COM Port stucture.

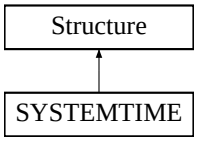
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.10 SYSTEMTIME Class Reference

Time structure.

Inheritance diagram for SYSTEMTIME:



Public Member Functions

- def **__str__** (self)

4.10.1 Detailed Description

Time structure.

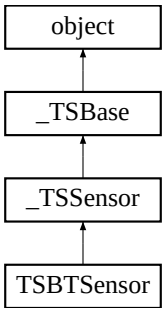
The documentation for this class was generated from the following file:

- win32_threespace_utils.py

4.11 TSBTSensor Class Reference

The TSBTSensor class is an interface layer for 3-Space Sensor Bluetooth units that communicate through the USB port.

Inheritance diagram for TSBTSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def getBatteryPercentRemaining (self, timestamp=False)

  202(0xca)
- def getBatteryStatus (self, timestamp=False)

  203(0xcb)
- def getBatteryVoltage (self, timestamp=False)

  generated functions BT_ 201(0xc9)
- def getButtonState (self, timestamp=False)

  250(0xfa)
- def getUARTBaudRate (self, timestamp=False)

  232(0xe8)
- def setUARTBaudRate (self, baud_rate, timestamp=False)

  231(0xe7)

Public Attributes

- **baudrate**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.11.1 Detailed Description

The TSBTSensor class is an interface layer for 3-Space Sensor Bluetooth units that communicate through the USB port.

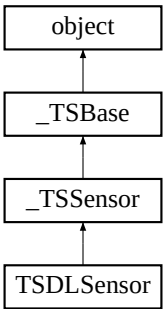
The documentation for this class was generated from the following file:

- threespace_api.py

4.12 TSDLSensor Class Reference

The TSDLSensor class is an interface layer for 3-Space Sensor Data-logging units that communicate through the USB port.

Inheritance diagram for TSDLSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def beginDataLoggingSession (self, timestamp=False)

  60(0x3c)
- def endDataLoggingSession (self, timestamp=False)

  61(0x3d)
- def formatAndInitializeSDCard (self, timestamp=False)

  59(0x3b)
- def getBatteryPercentRemaining (self, timestamp=False)

  202(0xca)
- def getBatteryStatus (self, timestamp=False)

  203(0xcb)
- def getBatteryVoltage (self, timestamp=False)

  201(0xc9)
- def getButtonState (self, timestamp=False)

  250(0xfa)

- def getClockValues (self, timestamp=False)

  63(0x3f)
- def setClockValues (self, month, day, year, hour, minute, second, timestamp=False)

  62(0x3e)
- def turnOffMassStorage (self, timestamp=False)

  58(0x3a)
- def turnOnMassStorage (self, timestamp=False)

  generated functions DL_ 57(0x39)

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

Additional Inherited Members

4.12.1 Detailed Description

The TSDLSensor class is an interface layer for 3-Space Sensor Data-logging units that communicate through the USB port.

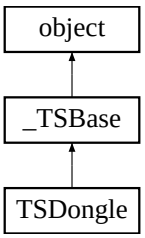
The documentation for this class was generated from the following file:

- threespace_api.py

4.13 TSDongle Class Reference

The TSDongle class is an interface layer for 3-Space Sensor Dongle units that communicate through the USB port and acts as an autonomous object that handles communication between the dongle and wireless 3-Space Sensor units.

Inheritance diagram for TSDongle:



Public Member Functions

- def **__getitem__** (self, idx)
- def **__init__** (self, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def broadcastSynchronizationPulse (self, timestamp=False)

  182(0xb6)
- def commitWirelessSettings (self, timestamp=False)

  197(0xc5)
- def **f8WriteRead** (self, logical_id, command, input_list=None)
- def faWriteRead (self, logical_id, command, input_list=None)

  Wireless New Protocol WriteRead.
- def getJoystickLogicalID (self, timestamp=False)

  242(0xf2)
- def getMouseLogicalID (self, timestamp=False)

  243(0xf3)
- def getReceptionBitfield (self, timestamp=False)

  183(0xb7)
- def **getSensorFromDongle** (self, idx)
- def getSerialNumberAtLogicalID (self, logical_id, timestamp=False)

  208(0xd0)
- def getSignalStrength (self, timestamp=False)

  214(0xd6)
- def getWirelessAddress (self, timestamp=False)

  198(0xc6)
- def getWirelessChannel (self, timestamp=False)

  194(0xc2)
- def getWirelessChannelNoiseLevels (self, timestamp=False)

  210(0xd2)
- def getWirelessHIDAsynchronousMode (self, timestamp=False)

  218(0xda)
- def getWirelessHIDUpdateRate (self, timestamp=False)

  216(0xd8)
- def getWirelessPanID (self, timestamp=False)

  192(0xc0)
- def getWirelessRetries (self, timestamp=False)

  212(0xd4)
- def getWirelessSlotsOpen (self, timestamp=False)

  213(0xd5)
- def getWirelessStreamingAutoFlushMode (self, timestamp=False)

  177(0xb1)
- def **reconnect** (self)
- def setJoystickLogicalID (self, logical_id, timestamp=False)

  240(0xf0)
- def setMouseLogicalID (self, logical_id, timestamp=False)

  241(0xf1)
- def **setSensorToDongle** (self, idx, hw_id)
- def setSerialNumberAtLogicalID (self, logical_id, serial_number, timestamp=False)

  209(0xd1)
- def setWirelessChannel (self, channel, timestamp=False)

  195(0xc3)

- def setWirelessHIDAsynchronousMode (self, mode, timestamp=False)
  217(0xd9)
- def setWirelessHIDUpdateRate (self, update_rate, timestamp=False)
  215(0xd7)
- def setWirelessPanID (self, PanID, timestamp=False)
  193(0xc1)
- def setWirelessRetries (self, retries, timestamp=False)
  211(0xd3)
- def setWirelessStreamingAutoFlushMode (self, mode, timestamp=False)
  generated functions DNG 176(0xb0)

Public Attributes

- **baudrate**
- **header_idx_lst**
- **protocol_args**
- **serial_number_hex**
- **serial_port**
- **timestamp_mode**
- **wireless_table**

Static Public Attributes

- **command_dict** = _TSBase.command_dict.copy()
- **wl_command_dict** = TSWLSensor.command_dict.copy()

4.13.1 Detailed Description

The TSDongle class is an interface layer for 3-Space Sensor Dongle units that communicate through the USB port and acts as an autonomous object that handles communication between the dongle and wireless 3-Space Sensor units.

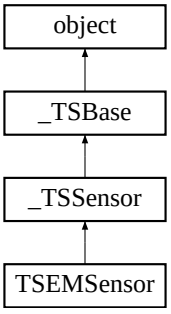
The documentation for this class was generated from the following file:

- threespace_api.py

4.14 TSEMSensor Class Reference

The TSEMSensor class is an interface layer for 3-Space Sensor Embedded units that communicate through the USB port or RS-232 port.

Inheritance diagram for TSEMSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def getInterruptStatus (self, timestamp=False)

  31(0x1f)
- def getPinMode (self, timestamp=False)

  30(0x1e)
- def getUARTBaudRate (self, timestamp=False)

  232(0xe8)
- def setPinMode (self, mode, pin, timestamp=False)

  generated functions EM_ 29(0x1d)
- def setUARTBaudRate (self, baud_rate, timestamp=False)

  231(0xe7)

Public Attributes

- **baudrate**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.14.1 Detailed Description

The TSEMSensor class is an interface layer for 3-Space Sensor Embedded units that communicate through the USB port or RS-232 port.

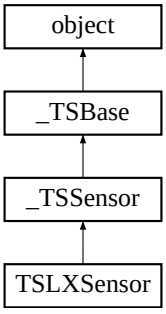
The documentation for this class was generated from the following file:

- threespace_api.py

4.15 TSLXSensor Class Reference

The TSLXSensor class is an interface layer for 3-Space Sensor LX units that communicate through the USB port.

Inheritance diagram for TSLXSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def getInterruptStatus (self, timestamp=False)

  31(0x1f)
- def getUARTBaudRate (self, timestamp=False)

  232(0xe8)
- def readInterruptType (self, timestamp=False)

  30(0x1e)
- def setInterruptType (self, mode, pin, timestamp=False)

  generated functions EM_ 29(0x1d)
- def setUARTBaudRate (self, baud_rate, timestamp=False)

  231(0xe7)

Public Attributes

- **baudrate**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.15.1 Detailed Description

The TSLXSensor class is an interface layer for 3-Space Sensor LX units that communicate through the USB port.

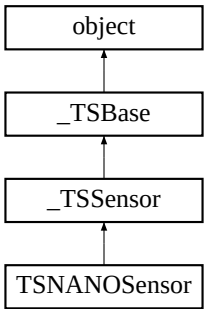
The documentation for this class was generated from the following file:

- threespace_api.py

4.16 TSNANOSensor Class Reference

The TSNANOSensor class is an interface layer for 3-Space Sensor NANO units that communicate through the USB port.

Inheritance diagram for TSNANOSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def getInterruptStatus (self, timestamp=False)

  31(0x1f)
- def getUARTBaudRate (self, timestamp=False)

  232(0xe8)
- def readInterruptType (self, timestamp=False)

  30(0x1e)
- def setInterruptType (self, mode, pin, timestamp=False)

  generated functions EM_ 29(0x1d)
- def setUARTBaudRate (self, baud_rate, timestamp=False)

  231(0xe7)

Public Attributes

- **baudrate**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.16.1 Detailed Description

The TSNANOSensor class is an interface layer for 3-Space Sensor NANO units that communicate through the USB port.

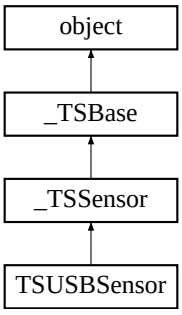
The documentation for this class was generated from the following file:

- threespace_api.py

4.17 TSUSBSensor Class Reference

The TSUSBSensor class is an interface layer for 3-Space Sensor USB units that communicate through the USB port.

Inheritance diagram for TSUSBSensor:



Public Member Functions

- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR)
- def getButtonState (self, timestamp=False)

  250(0xfa)
- def getUARTBaudRate (self, timestamp=False)

  232(0xe8)
- def setUARTBaudRate (self, baud_rate, timestamp=False)

  231(0xe7)

Public Attributes

- **baudrate**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.17.1 Detailed Description

The TSUSBSensor class is an interface layer for 3-Space Sensor USB units that communicate through the USB port.

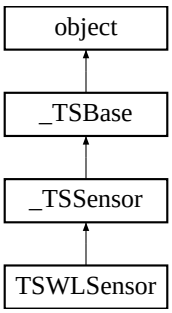
The documentation for this class was generated from the following file:

- threespace_api.py

4.18 TSWLSensor Class Reference

The TSWLSensor class is an interface layer for 3-Space Sensor Wireless units that communicate through the USB port or a 3-Space Sensor Dongle.

Inheritance diagram for TSWLSensor:



Public Member Functions

- def **__init__** (self, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR, logical_id=None, dongle=None)
- def **__new__** (cls, com_port=None, baudrate=_baudrate, timestamp_mode=TSS_TIMESTAMP_SENSOR, logical_id=None, dongle=None)
- def **close** (self)
- def commitWirelessSettings (self, timestamp=False)

  generated functions WL_ 197(0xc5)
- def getBatteryPercentRemaining (self, timestamp=False)

  202(0xca)
- def getBatteryStatus (self, timestamp=False)

  203(0xcb)
- def getBatteryVoltage (self, timestamp=False)

  201(0xc9)
- def getButtonState (self, timestamp=False)

  250(0xfa)
- def getWirelessAddress (self, timestamp=False)

  198(0xc6)
- def getWirelessChannel (self, timestamp=False)

  194(0xc2)
- def getWirelessPanID (self, timestamp=False)

  192(0xc0)
- def setWirelessChannel (self, channel, timestamp=False)

  195(0xc3)
- def setWirelessPanID (self, PanID, timestamp=False)

  193(0xc1)
- def **switchToWiredMode** (self)
- def **switchToWirelessMode** (self)

Public Attributes

- **baudrate**
- **callback_func**
- **latest_lock**
- **new_data**
- **protocol_args**
- **timestamp_mode**
- **wireless_com**
- **writeRead**

Static Public Attributes

- **command_dict** = _TSSensor.command_dict.copy()
- **reverse_command_dict** = dict(map(lambda x: [x[1][0], x[0]], command_dict.items()))

4.18.1 Detailed Description

The TSWLSensor class is an interface layer for 3-Space Sensor Wireless units that communicate through the USB port or a 3-Space Sensor Dongle.

The documentation for this class was generated from the following file:

- threespace_api.py

Index