PDO

class **ingeniamotion.pdo.PDOPoller**(*mc*, *servo*, *refresh_time*, *watchdog_timeout*, *buffer_size*)

Poll register values using PDOs

start()

Start the poller

Return type: None

stop()

Stop the poller

Return type: None

property **data**

Get the poller data. After the data is retrieved, the data buffers are cleared.

Return type: Tuple[List[float], List[List[Union[int, float]]]]

Returns: A tuple with a list of the readings timestamps and a list of lists with the readings values.

add_channels(*registers*)

Configure the PDOs with the registers to be read.

Parameters: **registers** (List[Dict[str, Union[int, str]]]) – list of registers to add to the Poller.

Return type: None

subscribe_to_exceptions(*callback*)

Get notified when an exception occurs on the PDO thread.

Parameters: **callback** (`Callable` [[`IMException`], `None`]) – Function to be called when an exception occurs.

Return type: `None`

property **available_samples**

Number of samples in the buffer.

Return type: `int`

---

class **ingeniamotion.pdo.PDONetworkManager**(*motion_controller*)

Manage all the PDO functionalities.

Parameters: **mc** – The MotionController.

class **ProcessDataThread**(*net*, *refresh_rate*, *watchdog_timeout*, *notify_send_process_data=None*, *notify_receive_process_data=None*, *notify_exceptions=None*)

Manage the PDO exchange.

Parameters:

- **net** (`EthercatNetwork`) – The EthercatNetwork instance where the PDOs will be active.
- **refresh_rate** (`Optional` [`float`]) – Determines how often (seconds) the PDO values will be updated.
- **watchdog_timeout** (`Optional` [`float`]) – The PDO watchdog time. If not provided it will be set proportional to the refresh rate.
- **notify_send_process_data** (`Optional` [`Callable` [[], `None`]]) – Callback to notify when process data is about to be sent.
- **notify_receive_process_data** (`Optional` [`Callable` [[], `None`]]) – Callback to notify when process data is received.
- **notify_exceptions** (`Optional` [`Callable` [[`IMException`], `None`]]) – Callback to notify when an exception is raised.

Raises: ValueError – If the provided refresh rate is unfeasible.

run()

Start the PDO exchange

Return type: `None`

stop()

Stop the PDO exchange

Return type: `None`

***static* high_precision_sleep(*duration*)**

Replaces the time.sleep() method in order to obtain more precise sleeping times.

Return type: `None`

create_pdo_item(*register_uid*, *axis=1*, *servo='default'*, *value=None*)

Create a PDOMapItem by specifying a register UID.

Parameters:
- **register_uid** (`str`) – Register to be mapped.
- **axis** (`int`) – servo axis. `1` by default.
- **servo** (`str`) – servo alias to reference it. `default` by default.
- **value** (`Union` [`float` , `int` , `None`]) – Initial value for an RPDO register.

Return type: `Union` [`RPDOMapItem` , `TPDOMapItem`]

Returns: Mappable PDO item.

Raises:
- ValueError – If there is a type mismatch retrieving the register object.
- AttributeError – If an initial value is not provided for an RPDO register.

create_pdo_maps(*rpdo_map_items*, *tpdo_map_items*)

Create the RPDO and TPDO maps from PDOMapItems.

Parameters:
- **rpdo_map_items** (`Union` [`RPDOMapItem` , `List` [`RPDOMapItem`]]) – The RPDOMapItems to be added to a RPDOMap.
- **tpdo_map_items** (`Union` [`TPDOMapItem` , `List` [`TPDOMapItem`]]) – The TDOMapItems to be added to a TPDOMap.

Return type: `Tuple` [`RPDOMap` , `TPDOMap`]

Returns: RPDO and TPDO maps.

***static* add_pdo_item_to_map(*pdo_map_item*, *pdo_map*)**

Add a PDOMapItem to a PDOMap.

Parameters:
- **pdo_map_item** (`Union` [`RPDOMapItem` , `TPDOMapItem`]) – The PDOMapItem.
- **pdo_map** (`Union` [`RPDOMap` , `TPDOMap`]) – The PDOMap to add the PDOMapItem.

Raises:
- ValueError – If an RPDOItem is tried to be added to a TPDOMap.
- ValueError – If an TPDOItem is tried to be added to a RPDOMap.

Return type: `None`

static `create_empty_rpdo_map()`

Create an empty RPDOMap.

Return type: `RPDOMap`

Returns: The empty RPDOMap.

static `create_empty_tpdo_map()`

Create an empty TPDOMap.

Return type: `TPDOMap`

Returns: The empty TPDOMap.

`set_pdo_maps_to_slave(`*rpdo_maps*, *tpdo_maps*, *servo='default'*`)`

Map the PDOMaps to the slave.

Parameters:
- **rpdo_maps** (`Union` [`RPDOMap` , `List` [`RPDOMap`]]) – The RPDOMaps to be mapped.
- **tpdo_maps** (`Union` [`TPDOMap` , `List` [`TPDOMap`]]) – he TPDOMaps to be mapped.
- **servo** (`str`) – servo alias to reference it. `default` by default.

Raises:
- ValueError – If there is a type mismatch retrieving the drive object.
- ValueError – If not all elements of the RPDO map list are instances of a RPDO map.
- ValueError – If not all elements of the TPDO map list are instances of a TPDO map.

Return type: `None`

clear_pdo_mapping(*servo='default'*)

Clear the PDO mapping within the servo.

Parameters: **servo** (`str`) – servo alias to reference it. `default` by default.

Raises: ValueError – If there is a type mismatch retrieving the drive object.

Return type: `None`

remove_rpdo_map(*servo='default'*, *rpdo_map=None*, *rpdo_map_index=None*)

Remove a RPDOMap from the RPDOMap list. The RPDOMap instance or the index of the map in the RPDOMap list should be provided.

Parameters:
- **servo** (`str`) – servo alias to reference it. `default` by default.
- **rpdo_map** (`Optional` [`RPDOMap`]) – The RPDOMap instance to be removed.
- **rpdo_map_index** (`Optional` [`int`]) – The index of the RPDOMap list to be removed.

Raises:
- ValueError – If the RPDOMap instance is not in the RPDOMap list.
- IndexError – If the index is out of range.

Return type: `None`

remove_tpdo_map(*servo='default'*, *tpdo_map=None*, *tpdo_map_index=None*)

Remove a TPDOMap from the TPDOMap list. The TPDOMap instance or the index of the map in the TPDOMap list should be provided.

Parameters:
- **servo** (`str`) – servo alias to reference it. `default` by default.
- **tpdo_map** (`Optional` [`TPDOMap`]) – The TPDOMap instance to be removed.
- **tpdo_map_index** (`Optional` [`int`]) – The index of the TPDOMap list to be removed.

Raises:
- ValueError – If the TPDOMap instance is not in the TPDOMap list.
- IndexError – If the index is out of range.

Return type: `None`

start_pdos(*network_type=None*, *refresh_rate=None*, *watchdog_timeout=None*)

Start the PDO exchange process.

Parameters:
- **network_type** (`Optional` [`COMMUNICATION_TYPE`]) – Network type (EtherCAT or CANopen) on which to start the PDO exchange.
- **refresh_rate** (`Optional` [`float`]) – Determines how often (seconds) the PDO values will be updated.
- **watchdog_timeout** (`Optional` [`float`]) – The PDO watchdog time. If not provided it will be set proportional to the refresh rate.

Raises:
- ValueError – If the refresh rate is too high.
- ValueError – If the MotionController is connected to more than one Network.
- ValueError – If network_type argument is invalid.
- IMException – If the MotionController is connected to more than one Network.
- ValueError – If there is a type mismatch retrieving the network object.
- IMException – If the PDOs are already active.

Return type: `None`

`stop_pdos()`

Stop the PDO exchange process.

Raises: IMException – If the PDOs are not active yet.

Return type: `None`

property `is_active`

Check if the PDO thread is active.

Return type: `bool`

Returns: True if the PDO thread is active. False otherwise.

`subscribe_to_send_process_data(`*callback*`)`

Subscribe be notified when the RPDO values will be sent.

Parameters: **callback** (`Callable` [[], `None`]) – Callback function.

Return type: `None`

`subscribe_to_receive_process_data(`*callback*`)`

Subscribe be notified when the TPDO values are received.

Parameters: **callback** (`Callable` [[], `None`]) – Callback function.

Return type: `None`

subscribe_to_exceptions(*callback*)

Subscribe be notified when there is an exception in the PDO process data thread.

If a callback is subscribed, the PDO exchange process is paused when an exception is raised. It can be resumed using the *resume_pdos* method.

Parameters: **callback** (`Callable` [[`IMException`], `None`]) – Callback function.

Return type: `None`

unsubscribe_to_send_process_data(*callback*)

Unsubscribe from the send process data notifications.

Parameters: **callback** (`Callable` [[], `None`]) – Subscribed callback function.

Return type: `None`

unsubscribe_to_receive_process_data(*callback*)

Unsubscribe from the receive process data notifications.

Parameters: **callback** (`Callable` [[], `None`]) – Subscribed callback function.

Return type: `None`

create_poller(*registers*, *servo='default'*, *sampling_time=0.125*, *buffer_size=100*, *watchdog_timeout=None*, *start=True*)

Create a register Poller using PDOs.

Parameters:

- **registers** (`List` [`Dict` [`str` , `Union` [`int` , `str`]]]) – list of registers to add to the Poller. Dicts should have the follow format:

```
[
    { # Poller register one
        "name": "CL_POS_FBK_VALUE",  # Register name.
        "axis": 1  # Register axis.
        # If it has no axis field, by default axis 1.
    },
    { # Poller register two
        "name": "CL_VEL_FBK_VALUE",  # Register name.
        "axis": 1  # Register axis.
        # If it has no axis field, by default axis 1.
    }
]
```

- **servo** (`str`) – servo alias to reference it. `default` by default.
- **sampling_time** (`float`) – period of the sampling in seconds. By default `0.125` seconds.
- **watchdog_timeout** (`Optional` [`float`]) – The PDO watchdog time. If not provided it will be set proportional to the refresh rate.
- **buffer_size** (`int`) – number maximum of sample for each data read. `100` by default.
- **start** (`bool`) – if `True` , function starts poller, if `False` poller should be started after. `True` by default.

Return type: `PDOPoller`

Returns: The poller instance.

`unsubscribe_to_exceptions(callback)`

Unsubscribe from the exceptions in the process data notifications.

Parameters: **callback** (`Callable` [[`IMException`], `None`]) – Subscribed callback function.

Return type: `None`