ptb-drivers Documentation

Release 0.1

Robert Jördens

Jul 04, 2019

CONTENTS

Contents:

CHAPTER

ONE

SYNTHESIZER

1.1 ptb.synth_protocol module

class ptb.synth_protocol.**SynthProtocol**
Protocol for the PTB synthesizer (ADF4350-based)

get(*key*)
Get a synthesizer configuration setting (instance attribute).

async locked()
Return the reference lock status.

Returns: bool: True if locked

async save(*regs=None*)
Save the six registers to the EEPROM. That data is loaded on boot of the synthesizer.

Args:

regs (list(int), optional): 32 bit register values (reg5 down to reg0). If no register values are passed, then the ones calculated by `set_frequency()` are used.

set(***kwargs*)
Configure synthesizer settings.

See datasheet and `ADF4350` for fields and values.

This does not update the synthesizer settings or registers. This method only sets instance attributes that affect the calculation of register values during `set_frequency()`.

async start(*regs=None*)
Send the six registers to the synthesizer.

Args:

regs (list(int), optional): 32 bit register values (reg5 down to reg0). If no register values are passed, then the ones calculated by `set_frequency()` are used.

async version()
Return the hardware/firmware version.

Returns: str: Version string.

1.2 ptb.synth_tcp module

class ptb.synth_tcp.**SynthTCP**(*reader*, *writer*)

1.3 ptb.adf4350 module

class ptb.adf4350.**ADF4350**

From the dtasheet with analogies of naming from https://github.com/analogdevicesinc/no-OS/blob/master/drivers/adf4350/adf4350.h

set_frequency(*f_out*)

Set output frequency.

Args: f_out (float): Desired frequency

Returns: Actual output frequency set

CHAPTER

TWO

TEMPERATURE SENSOR

2.1 ptb.temp_protocol module

class ptb.temp_protocol.**TempProtocol**
Protocol for the PTB multi-channel temperature sensor

async get(*channel*)
Measure the temperature on one channel.

Args: channel (int): Channel index to measure on

Return: float: Temperature

async get_all()
Measure and return the temperatures on all channels.

Returns: list(float): Temperatures on all channels

async version()
Return the hardware/firmware version.

Returns: str: Version string.

2.2 ptb.temp_tcp module

class ptb.temp_tcp.**TempTCP**(*reader*, *writer*)

CHAPTER

THREE

VOLTAGE SOURCE

3.1 ptb.voltage_protocol module

class ptb.voltage_protocol.**VoltageProtocol**

Protocol for the PTB multi-channel voltage source

factory()

Reset gains and offsets to default values

async get_data(*channels=None*)

Get raw channel output values. Values are given in DAC LSBs (integers).

Args: channels (list(int)): Target channels. Defaults to 1...8

Returns: list(int)): DAC values, one for each target channel.

async get_temperature()

Get temperature data for controller and amplifier boards.

Returns: list(float): Raw ADC values for the NTCs

ldac()

Pulse LDAC to all DACs to load values into active registers.

async set_data(*values*, *channels=None*)

Set raw channel output values. Values are given in DAC LSBs (integers). Data becomes active only after *ldac()*.

Args: values (list(int)): DAC values, one for each target channel. channels (list(int)): Target channels. Defaults to 1...len(values)

Returns: list(int): Actual values returned by the device.

async set_gain(*values*, *channels=None*)

Set channel gains. Channel gains are processed within the microcontroller and become active on `set_volt()` and a subsequent *ldac()*.

Gains are given in *2**16/full_scale* where *full_scale = u_max - u_min*.

Args: values (list(float)): Gains, one for each target channel. channels (list(int)): Target channels. Defaults to 1...len(values)

Returns: list(float): Actual values returned by the device.

async set_offset(*values*, *channels=None*)

Set channel offsets. Channel gains are processed within the microcontroller and become active on `set_volt()` and a subsequent *ldac()*.

Offsets are given in DAC LSBs (integers).

Args: values (list(int)): Offsets, one for each target channel. channels (list(int)): Target channels. Defaults to 1...len(values)

Returns: list(int): Actual values returned by the device.

async set_voltage(*values*, *channels=None*)
Set output voltages. Voltages become active only after `ldac()`.

Args: values (list(float)): Voltages, one for each target channel. channels (list(int)): Target channels. Defaults to 1...len(values)

Returns: list(float): Actual values returned by the device.

async version()
Return the hardware/firmware version.

Returns: str: Version string.

3.2 ptb.voltage_tcp module

class ptb.voltage_tcp.**VoltageTCP**(*reader*, *writer*)

CHAPTER

FOUR

SHUTTER CONTROLLER

4.1 ptb.shutter_protocol module

class ptb.shutter_protocol.**ShutterProtocol**
Protocol for the PTB multi-channel shutter controller

async clear()
Clear all error flags.

Returns:

tuple(bool): Error flag on all channels after clearing. *True* means that there is an error.

async passthrough(*shutter*, *cmd*)
Execute a low level command.

Args: shutter (int): Shutter index (1-3) cmd (bytes(1)): Single character command (e.g. W)

Return: bytes: Response

async status()
Return the error flags on all channels.

Returns:

tuple(bool): Error flag on all channels. *True* means that there is an error.

async version()
Return the hardware/firmware version.

Returns: str: Version string.

4.2 ptb.shutter_tcp module

class ptb.shutter_tcp.**ShutterTCP**(*reader*, *writer*)

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX