Unified Report: DPO4054 Python Class for Tektronix Oscilloscope Automation

Introduction

The `DPO4054` Python class provides a robust, high-level interface for automating the Tektronix DPO4054 oscilloscope using SCPI commands via PyVISA. It supports full instrument configuration, trigger and acquisition control, measurement setup and retrieval, waveform and image data export, and integration with scientific Python libraries for advanced analysis (including FFT).

---

1. Instrument Initialization and Basic Control

Initialization

- **Purpose:** Connect to the oscilloscope via VISA.
- **Usage:**

```
osc = DPO4054('USB0::0x0699::0x0401::C020132::INSTR')
print("Connected to:", osc.identify())
```

Basic Commands

- `identify()`: Returns instrument ID.
- `reset()`: Resets the instrument.
- `clear_status()`: Clears status and error queues.
- `self_test()`: Runs a self-test and returns result.
- `get_error()`: Gets all current error messages.

---

2. Channel (Vertical) Control

Display, Scaling, and Position

- `set_channel_display(channel, display)`: Show/hide channel.
- `set_channel_scale(channel, volts_per_div)`: Set V/div.
- `get_channel_scale(channel)`: Query V/div.
- `set_channel_position(channel, position)`: Set vertical position.
- `get_channel_position(channel)`: Query vertical position.

Coupling and Bandwidth

- `set_channel_coupling(channel, coupling)`: Set coupling (DC/AC/GND).
- `get_channel_coupling(channel)`: Query coupling.
- `set_channel_bandwidth_limit(channel, enable)`: Enable/disable bandwidth limit.
- `get_channel_bandwidth_limit(channel)`: Query bandwidth limit.

Example:

```
osc.set_channel_display(1, True)
osc.set_channel_scale(1, 0.5)
osc.set_channel_coupling(1, 'DC')
```

---

3. Timebase (Horizontal) Control

Time/Div and Trigger Position

- `set_timebase_scale(seconds_per_div)`: Set time/div.
- `get_timebase_scale()`: Query time/div.
- `set_timebase_position(seconds)`: Set horizontal trigger position (offset).
- `get_timebase_position()`: Query horizontal trigger position.

Example:

```
osc.set_timebase_scale(1e-3)  # 1 ms/div
osc.set_timebase_position(0.0)  # Center trigger
```

---

4. Trigger Control

Mode, Source, Slope, and Level

- `set_trigger_mode(mode)`: Set trigger mode (EDGE, GLITch, etc.).
- `get_trigger_mode()`: Query trigger mode.
- `set_trigger_edge_source(channel)`: Set trigger source channel.
- `get_trigger_edge_source()`: Query trigger source.
- `set_trigger_edge_slope(slope)`: Set trigger slope (RISE/FALL/EITH).
- `get_trigger_edge_slope()`: Query slope.
- `set_trigger_level(channel, level_volts)`: Set trigger level.
- `get_trigger_level(channel)`: Query trigger level.
- `get_trigger_status()`: Get current trigger state.

Example:

```
osc.set_trigger_mode('EDGE')
osc.set_trigger_edge_source(1)
osc.set_trigger_edge_slope('RISE')
osc.set_trigger_level(1, 1.0)
```

---

5. Acquisition Control

- `run_acquisition()`: Start acquisition.
- `stop_acquisition()`: Stop acquisition.
- `single_sequence(timeout, sampling_interval)`: Run a single sequence acquisition and wait for completion.
- `set_acquire_continuous()`: Set to continuous acquisition mode.
- `wait_for_acquisition_complete(timeout)`: Wait for acquisition to finish.

---

6. Measurement Management

Add/Remove/Clear Measurements

- `add_measurement(meas_type, channel, slot=1, source=1)`: Add a measurement to a slot.
- `remove_measurement(channel, slot=1, source=1)`: Remove measurement from slot.
- `clear_all_measurements(slot=1)`: Clear all measurement slots.

Retrieve Measurements

- `get_measurement(channel, slot=1, source=1)`: Get value from a slot.
- `get_all_measurements()`: Get all enabled measurement values as a dictionary.

Example:

```
osc.add_measurement('FREQ', 1, slot=1)
value = osc.get_measurement(1, slot=1)
print(f"Frequency: {value} Hz")
```

---

7. Math Functions

- `enable_math(func)`: Define and enable a math function.
- `disable_math()`: Disable math display.

---

8. Waveform Data Transfer and FFT Analysis

Fetching Waveform Data

- `fetch_waveform(channel)`: Returns time and voltage arrays for a channel.

Exporting and Plotting Data

- `export_waveform_to_csv(times, voltages, filename)`: Export data to CSV.
- `export_and_plot_all_waveforms(image=False, image_path="waveforms.png", image_format="png")`: Fetch and plot all displayed channels.

FFT Analysis Example

After fetching data, use NumPy for FFT:

```
import numpy as np
import matplotlib.pyplot as plt

times, volts = osc.fetch_waveform(1)
dt = times[^1] - times[^0]
fs = 1.0 / dt
fft_vals = np.fft.fft(volts)
fft_freqs = np.fft.fftfreq(len(volts), dt)
plt.plot(fft_freqs[fft_freqs >= 0], np.abs(fft_vals[fft_freqs >= 0]))
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.title("FFT of Channel 1")
plt.show()
```

---

9. Screen Capture and File Transfer

- `save_image_to_scope(filepath_on_scope, image_format="PNG")`: Save a screen image to scope or USB.
- `read_file_from_scope(filepath_on_scope, filename_on_pc)`: Fetch a file (e.g., image) from the scope to the PC.
- `save_and_fetch_screen_image_usb(filename_on_usb, save_to_pc, img_format="PNG")`: Save and fetch screen image from USB.

---

10. Utility and Automation

- `autoset_and_wait(timeout=10)`: Run autoset and wait for completion.
- `save_setup(filename="setup.stp")`: Save current setup to internal memory.
- `recall_setup(filename="setup.stp")`: Recall setup from internal memory.

---

11. Measurement Statistics

- `get_channel_stats(channel)`: Return mean, min, max, and stddev for a channel's waveform.

---

12. Example: Automated Measurement and FFT Script

```
osc = DPO4054('USB0::0x0699::0x0401::C020132::INSTR')
osc.reset()
osc.set_channel_display(1, True)
osc.set_channel_scale(1, 0.2)
osc.set_timebase_scale(1e-3)
osc.set_trigger_mode('EDGE')
osc.set_trigger_edge_source(1)
osc.set_trigger_edge_slope('RISE')
osc.set_trigger_level(1, 0.5)
osc.add_measurement('FREQ', 1, slot=1)
osc.single_sequence(timeout=10)
freq = osc.get_measurement(1, slot=1)
print(f"Measured frequency: {freq} Hz")
times, volts = osc.fetch_waveform(1)
# FFT
dt = times[^1] - times[^0]
fs = 1.0 / dt
fft_vals = np.fft.fft(volts)
fft_freqs = np.fft.fftfreq(len(volts), dt)
plt.plot(fft_freqs[fft_freqs >= 0], np.abs(fft_vals[fft_freqs >= 0]))
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.title("FFT of Channel 1")
plt.show()
osc.export_and_plot_all_waveforms(image=True)
osc.set_acquire_continuous()
osc.close()
```

---

13. Summary Table

Functionality	**Methods (non-overlapping)**
Identification & Status	`identify`, `reset`, `clear_status`, `self_test`, `get_error`
Channel Control	`set_channel_display`, `set_channel_scale`, `get_channel_scale`, `set_channel_position`,
	`get_channel_position`, `set_channel_coupling`, `get_channel_coupling`,
	`set_channel_bandwidth_limit`, `get_channel_bandwidth_limit`
Timebase (Horizontal)	`set_timebase_scale`, `get_timebase_scale`, `set_timebase_position`, `get_timebase_position`
Trigger Control	`set_trigger_mode`, `get_trigger_mode`, `set_trigger_edge_source`, `get_trigger_edge_source`,
	`set_trigger_edge_slope`, `get_trigger_edge_slope`, `set_trigger_level`, `get_trigger_level`,
	`get_trigger_status`
Acquisition Control	`run_acquisition`, `stop_acquisition`, `single_sequence`, `set_acquire_continuous`,
	`wait_for_acquisition_complete`
Measurement Management	`add_measurement`, `remove_measurement`, `clear_all_measurements`,
	`get_measurement`, `get_all_measurements`

Functionality	**Methods (non-overlapping)**
Math Functions	`enable_math`, `disable_math`
Waveform Data	`fetch_waveform`, `export_waveform_to_csv`, `export_and_plot_all_waveforms`
Screen Capture & Files	`save_image_to_scope`, `read_file_from_scope`, `save_and_fetch_screen_image_usb`
Automation & Utility	`autoset_and_wait`, `save_setup`, `recall_setup`
Statistics	`get_channel_stats`

Conclusion

The `DPO4054` Python class is a comprehensive toolkit for automating the Tektronix DPO4054 oscilloscope. It enables full remote control, robust measurement and data export, advanced analysis (including FFT), and seamless integration into laboratory and research workflows. Each function is carefully mapped to SCPI commands, providing both power and flexibility for users.