# Ergänzung zum Beitrag in FA 6/22, S. 454 f. „Kennlinienschreiber für Halbleiter-Leistungsbauteile“

Ergänzend zum Beitrag präsentieren wir hier noch eine Zusammenstellung nutzbarer Befehle, die in der gedruckten Ausgabe keinen Platz mehr fanden.

## Befehlsübersicht

Im Prinzip kommen im Python-Script *Curve_Tracer.py* drei Befehle zum Einsatz. Mit diesen kann der Benutzer den Kennlinienschreiber nach eigenen Bedürfnissen steuern, weil er die volle Kontrolle über die Hardware hat.

Der Befehl *set_set_points(set_point_1, voltage_or_current_1, set_point_2, voltage_or_current_2)* legt fest, auf welchen Wert (Strom oder Spannung) die beiden Kanäle regeln sollen, führt aber noch keinen Puls aus.
Parameter:
– set_point_1: Wert, für Kanal 1 (kann Strom oder Spannung sein). Fließkommazahl in Python.
– voltage_or_current_1: Legt fest, ob Kanal 1 auf Strom oder Spannung regelt. String in Python, entweder "voltage" oder "current".
– set_point_2: Wert, für Kanal 2 (kann Strom oder Spannung sein). Fließkommazahl in Python.
– voltage_or_current_2: Legt fest, ob Kanal 2 auf Strom oder Spannung regelt. String in Python, entweder "voltage" oder "current".
Rückgabewert: keiner

Der Befehl *; pulse()* führt einen Pulse aus, so wie zuvor mit dem Befehl set_set_points(...) definiert. Wartezeit im Skript nötig, bevor mit read_all_data_regs() anschließend die Ergebnisse des Pulses von der Hardware heruntergeladen werden können.
Parameter: keine
Rückgabewert: keiner

Der Befehl *read_all_data_regs()* liest dies Daten in ein Array ein.
Parameter: keine
Rückgabewert: Tupel von 8 Arrays ADC_1_V, ADC_1_C, DAC_1_high, DAC_1_low, ADC_2_V, ADC_2_C, DAC_2_high, DAC_2_low) welche alle Samples der vier ADCs und der vier DACs beinhalten
– ADC_1_V: Array, welches alle Samples des ADC von Kanal 1, Spannungsmessung beinhaltet. Werte können im Skript zur Ergebnisdarstellung verwendet werden.
– ADC_1_C: wie ADC_1_V, aber Kanal 1, Strommessung
– ADC_2_V: wie ADC_1_V, aber Kanal 2, Spannungsmessung
– ADC_2_C: wie ADC_1_V, aber Kanal 2, Strommessung
– DAC_1_high: Array, welches alle Samples des DAC für den Highside-MOSFET von Kanal 1 beinhaltet. Werte werden für die Ergebnisdarstellung nicht benötigt, sie dienen nur der Information, wie die Regelung zur Ausführung des Pulses gearbeitet hat.
– DAC_1_low: wie DAC_1_high für den Lowside-MOSFET von Kanal 1
– DAC_2_high: wie DAC_1high für den Highside-MOSFET von Kanal 2
– DAC_2_low: wie DAC_2_low für den Lowside-MOSFET von Kanal 2.

Zusätzlich gibt es noch folgende Befehle, die aber nur selten zum Einsatz kommen und daher nur der Vollständigkeit halber aufgeführt sind:
*__init__(serial_port_name, serial_port_baud_rate)*
*load_calibration()*
*code_to_physical(channel, voltage_current, code)*
*physical_to_code(channel, voltage_current, physical)*
*set_constants(N_samples)*
*reset()*
*set_parameters(channel, Kp, Ki, Kd, Kf, Mp, Mi, Md, Mf, bias_high, bias_low, limit)*
*set_default_parameters()*
*set_set_points(set_point_1, voltage_or_current_1, set_point_2, voltage_or_current_2)*
*calibrate()*
*pulse()*
*write_reg(address, data)*
*read_reg(address)*
*read_data_reg(data_reg_offset, start_index = 0)*
*read_all_data_regs(converted = True)*
*plot_results(plot_mode = "converted", show_current = "yes")*
*write_test_reg(addr, data)*