Detaillierte (KI generierte) Erklärungen des Programmcodes von Test-01.py

Dieser Python-Code überwacht ein GPIO-Signal und registriert Änderungen des Signalstatus, indem er einen Interrupt-Handler verwendet. Der Code läuft auf Mikrocontroller-Plattformen wie dem Raspberry Pi Pico oder ähnlichen Geräten.

Hier eine Erklärung des Codes:

Überblick

- Der Code verwendet das Modul `machine` für GPIO-Operationen und `utime` für die Zeitmessung
- Eine LED auf dem Board wird als Statusanzeige verwendet.
- Ein Interrupt-Handler registriert Statusänderungen an einem bestimmten GPIO-Pin (GPIO15) und protokolliert die Zeiten, zu denen das Signal an- und ausgeht, um die Dauer der Signale zu messen.

Erklärungen der Codeabschnitte

1. **Importierte Module**:

```python
from machine import Pin
import utime
```

Diese Module werden verwendet, um die GPIO-Pins zu steuern und Zeitstempel zu erzeugen.

2. **LED-Steuerung**:

```python
led_onboard = machine.Pin('LED', machine.Pin.OUT, value=0)
```

Hier wird die Onboard-LED als `Pin`-Objekt definiert, das als Ausgang genutzt wird und initial auf `AUS` (Wert `0`) gesetzt ist.

3. **Interrupt-Handler `Signal_INT`**:

  ```python
def Signal_INT(pin):
...
```
- Wird aufgerufen, wenn sich der Signalzustand ändert.
- Deaktiviert den Interrupt-Handler kurzzeitig mit `Signal.irq(handler=None)`, um zu verhindern, dass er während der Verarbeitung erneut aufgerufen wird.
- Erkennt, ob das Signal von `AUS` auf `AN` oder von `AN` auf `AUS` wechselt, und führt entsprechende Aktionen aus:

  - **Bei `AN`**
  - Setzt `Signal_Status` auf 1.
  - Schaltet die LED ein.
  - Speichert die aktuelle Zeit in `tck1`.

  - **Bei `AUS`**
  - Setzt `Signal_Status` auf 0.
  - Schaltet die LED aus.
  - Speichert die aktuelle Zeit in `tck2`.
  - Berechnet die Periodendauer, falls `tck2_v` gesetzt ist.
  - Aktiviert den Interrupt-Handler wieder am Ende.

4. **GPIO-Konfiguration**:

  ```python

  Signal = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL_DOWN)

  Signal.irq(trigger=machine.Pin.IRQ_RISING | machine.Pin.IRQ_FALLING, handler=Signal_INT)

  ```

Der Pin 15 wird als Eingang mit Pull-Down-Widerstand konfiguriert. Der Interrupt-Handler wird so eingestellt, dass er bei steigenden und fallenden Flanken (`IRQ_RISING | IRQ_FALLING`) ausgelöst wird.

5. **Initialisierung der Variablen**:

  ```python
Signal_Status = Signal.value()
tck1 = 0; tck2 = 0; tck2_v = -1; periode = -1

  ```

Die Variablen werden initialisiert, um Signalstatus und Zeitstempel zu speichern.

6. **Hauptschleife**:

  ```python
while True:
    print("counter = ", counter, "\t Pulsdauer = ", tck2-tck1, 'ms')
    utime.sleep(2)
   counter += 1
```
Eine Endlosschleife, die alle 2 Sekunden eine Ausgabe der aktuellen Zählvariable und der Pulsdauer ausgibt. Hier könnten weitere Aktivitäten hinzugefügt werden.

Zusammenfassung

Der Code dient zur Überwachung von Signalen an einem GPIO-Pin und zur Messung der Zeit zwischen Signaländerungen. Dies könnte in Anwendungen wie Taktmessungen oder Erkennung von Signalpulsen nützlich sein.