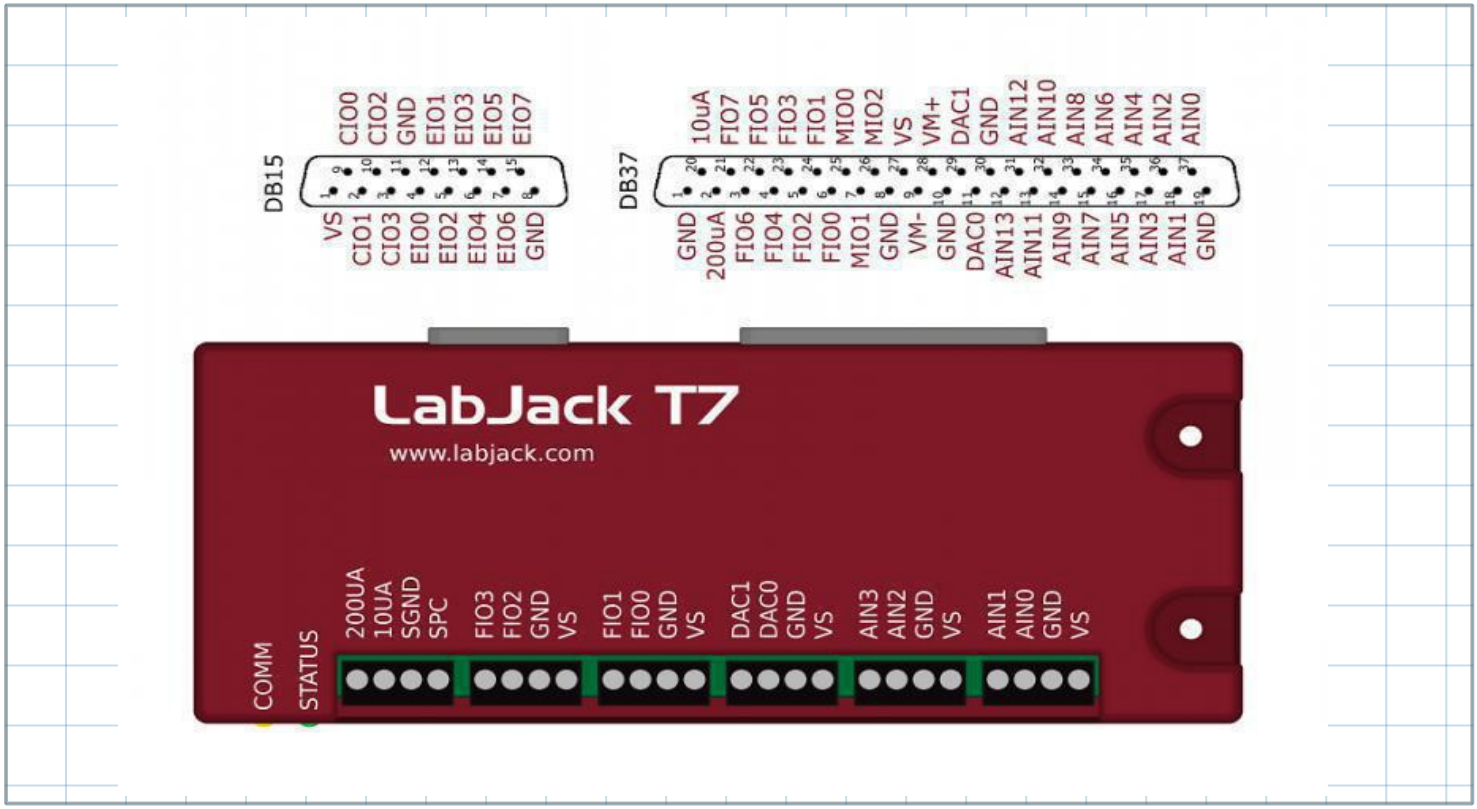
Python Configuration of LabJack T7 devices in MatDeck using GUIs

In this document, we will illustrate how Python can be used in MatDeck documents to configure LabJack devices. In addition LabJack devices can be configured in the MatDeck Script which has a style similar to C++. Alternatively, you can use simple and intuitive GUI plug-ins which MatDeck provides for the configuration of LabJack devices as illustrated in other examples.

Use of the LabJack T7 device in this experiment

In this experiment, the LabJack T7 device is used for several roles. First, the pulse width modulation (PWM) is produced using the DIO extended features, PWM outputs at DIO0(FIO0) and a high-speed counter at the DIO18 output. These extended features are explained in detail in the following links:
https://labjack.com/support/datasheets/t7/digital-io/extended-features
https://labjack.com/support/datasheets/t7/digital-io/extended-features/pwm-out
https://labjack.com/support/datasheets/t7/digital-io/extended-features/high-speed-counter
Python is used to configure these extended features.

In order to configure and use the device, the LabJack T7 device should be opened. The Python function ljm.openS() returns the handle of the opened device.



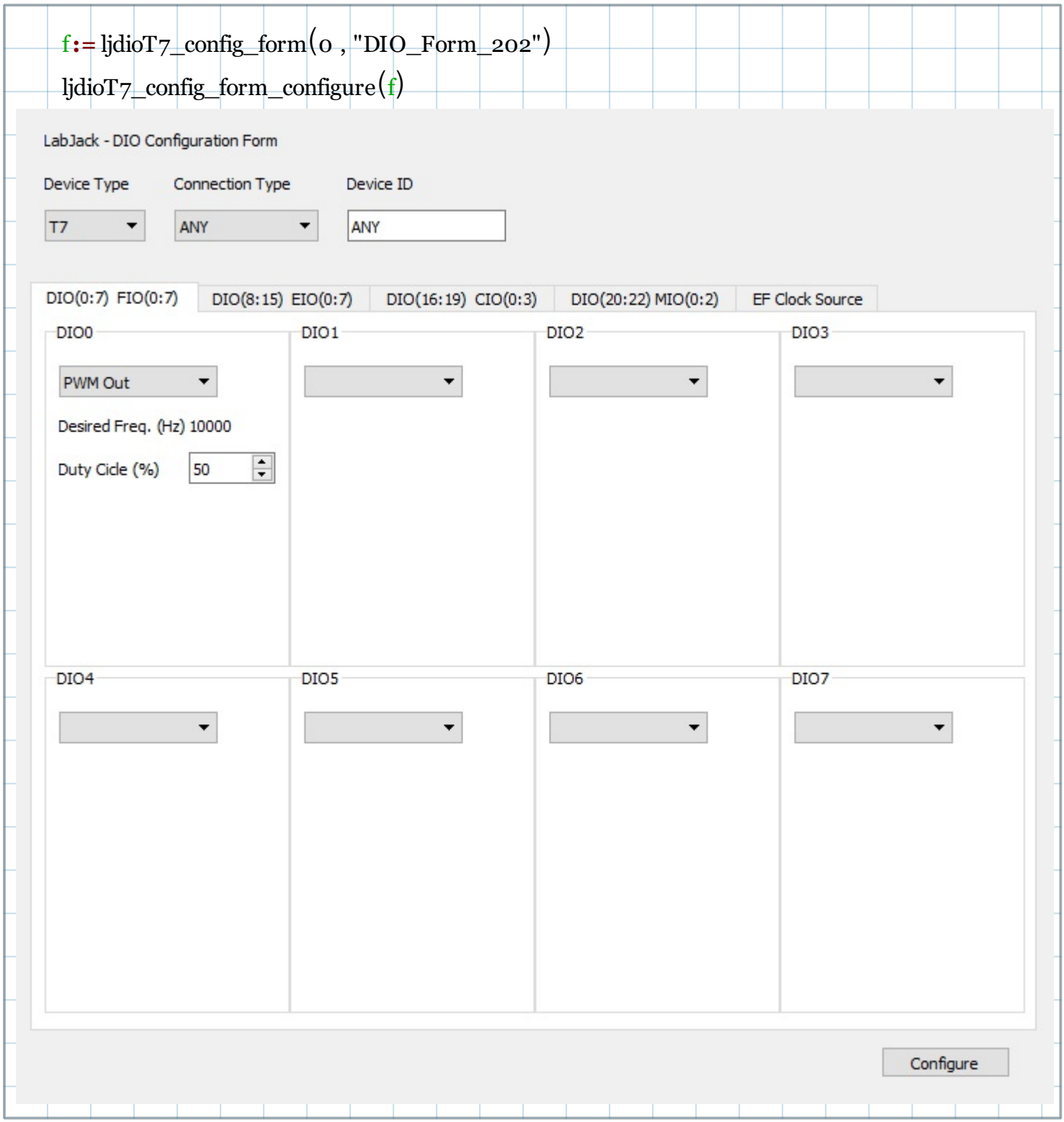
Using MatDeck GUIs for Configuration

PWM Out at FIO0(DIO0) requires the clock source, thus the clock is also configured. There are three parameters to select for the configuration: clock source, clock divisor and the roll value for the given clock. There are three different clocks supported by the T7, the most common is clock0 whose frequency is 80MHz. The clock divisor can be any power of two from 1, 2, up to 256, in this example we select a value of one. In this example, we use a divisor value equal to one. The roll value is determined according to the desired frequency of the PWM Out signal. For example, if the desired frequency is 1kHz, the roll value is

80Hz/Divisor/1kHz=80000. Before all values are written to the appropriate registers, clock0 should be disabled. PWM out at FIO0 (DIO0) is configured by setting DIO0_EF_INDEX equal to 0. A duty cycle is set by writing the appropriate value at DIO0_EF_CONFIG_A. If the desired value of the duty cycle is 50%, then DIO0_EF_CONFIG_A will be equal to half of the roll value, which is 40000. A high speed counter at DIO18 will be used to check the obtained PWM out frequency.

MatDeck provides the ljdioT7_config form which can be used for all the configurations explained above.

```
f:= ljdioT7_config_form(0 , "DIO_Form_202")
ljdioT7_config_form_configure(f)
```



The variable counter is used to export the value from Python to MatDeck:

```
counter := 0
data := 0
```

Python Code for Configuration

Python code is below.

```
#py
import time
from labjack import ljm
# Open first found LabJack
handle = ljm.openS("ANY", "ANY", "ANY")  # Any device, Any connection, Any identifier
pwmDIO = 0
aNames = ["DIO18_EF_ENABLE", "DIO%i_EF_ENABLE" % pwmDIO]
aValues = [ 1, 1]
numFrames = len(aNames)
results = ljm.eWriteNames(handle, numFrames, aNames, aValues)

# Wait 1 second.
time.sleep(1.0)

# Read from the counter.
value = ljm.eReadName(handle, "DIO18_EF_READ_A")
counter = value
print("\nCounter = %f" % (value))

# Stream Configuration
aScanListNames = ["DIO0"]  # Scan list names to stream
numAdd = len(aScanListNames)
aScanList = ljm.namesToAddresses(numAdd, aScanListNames)[0]
scanRate = 10000
# Ensure triggered stream is disabled.
ljm.eWriteName(handle, "STREAM_TRIGGER_INDEX", 0)
# Enabling internally-clocked stream.
ljm.eWriteName(handle, "STREAM_CLOCK_SOURCE", 0)
sPR = int(scanRate / 1)
scanRate = ljm.eStreamStart(handle, sPR, numAdd, aScanList, scanRate)
ret = ljm.eStreamRead(handle)
data = ret
ljm.eStreamStop(handle)

# Turn off PWM output and counter
aNames = ["DIO_EF_CLOCK0_ENABLE", "DIO%i_EF_ENABLE" % pwmDIO]
aValues = [0, 0]
numFrames = len(aNames)
results = ljm.eWriteNames(handle, numFrames, aNames, aValues)

# Close handle
ljm.close(handle)

###
// Here is the end of Python block
```

Check PWM Out Frequency

The frequency of the PWM Out signal can be checked by reading the value of the counter after one second has passed.

counter = 1004

```
ti := ynodes(x, 0, 1, 10000)
dat := data[0]
graph := join_mat_cols(ti, mat_transpose(dat))
```

Here, we plot PWM out signal that is read in Python.