Lua mode

Last Modified on 06/11/2019 11:01 am CET

Background

The X45e units (Ethernet version) can be used in more generic applications. This article focuses on the Lua mode. It should be used if a functionality is required that is not part of the build in functionalities. In this mode a Lua script controls the X45e unit (conveyor or function). It is possible to store up to 10 scripts at the unit but only one can run at a time. It is not possible to call one script from another. Password protection is available.

For deeper information please check the following manuals:

- **User documentation - Electrical System (X45e)**
- **User documentation - Parameter Setting Tool**

Prerequisites

Hardware

- **PC (Windows 7 or later)**
- **X45e unit (with power cables)**
- **Mini USB cable**

Software

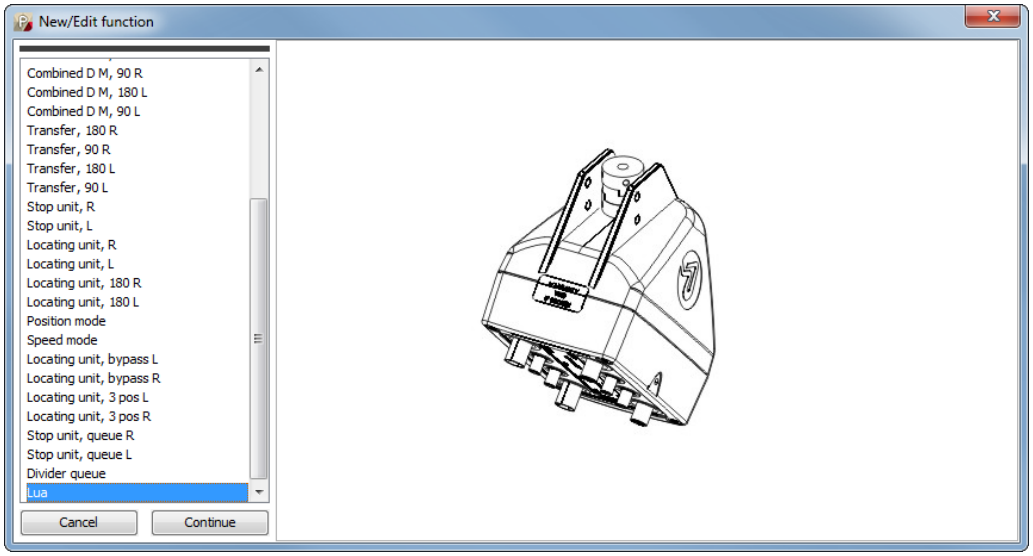
- **X45e - Parameter Setting Tool**

Software version

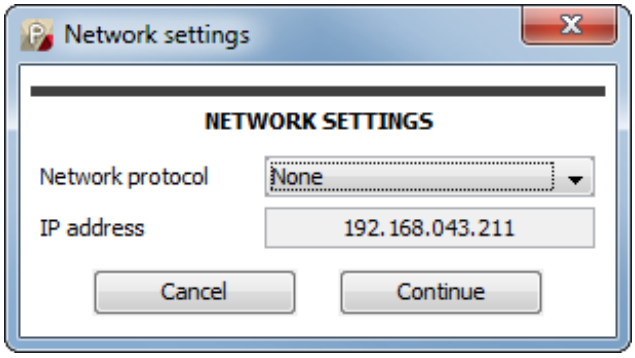
Make sure the unit has the latest software versions installed. Look at MyFlexLink under the Download Area for information about the latest release. This article was written when the FEG version 8.2.4 was the latest version. The PST version used is 3.4.1.

Wizard

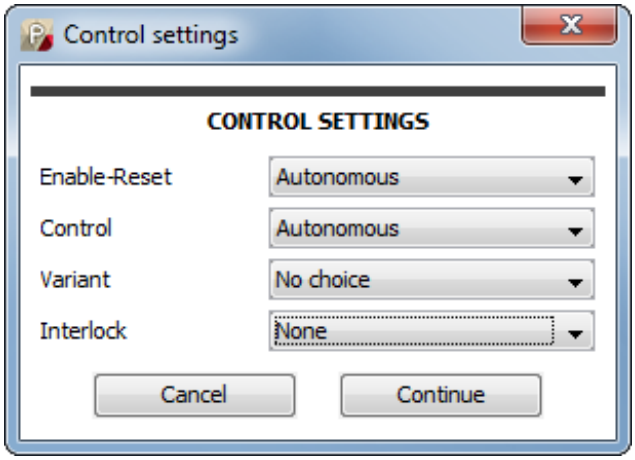
To set the proper parameter setting to the X45e unit the wizard is a good help. This is started with the command ***Change Node***.



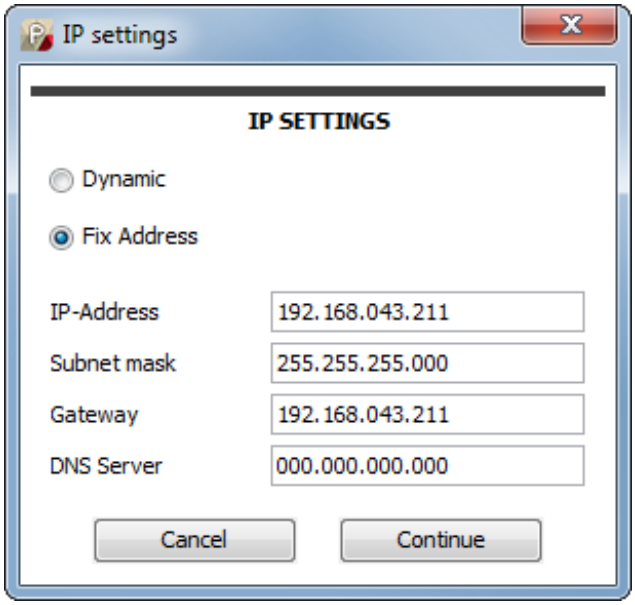
The first window in the wizard decides which standard (physical) type the unit should be set to. The ***Lua*** option should be selected in this case.



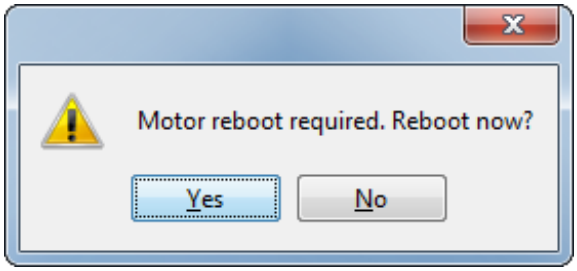
The Network settings window can only be set to ***None***. IP address is shown if already set.



Control settings window is the next popup window in the wizard. None of the choices (Enable-Reset, Control, Variant, Interlock) could be set to anything else than default values.



The last popup window in the wizard is the IP settings window. Here a fixed IP address can be set. Ethernet connection is required to use Lua editor. Otherwise only a file transfer via USB is possible.



After the wizard is completed the unit is updated with these settings (if working in Direct mode). Then a reboot of the unit is required to initialize Lua mode in the unit.

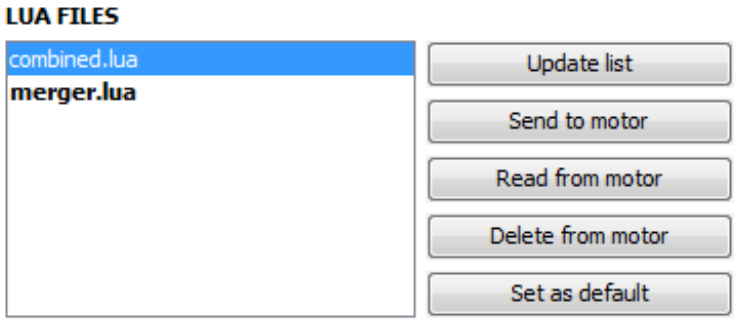
	Config	Value	Network	
Speed setpoint 1	0	27	27	rpm
Speed setpoint 2	0	360	360	rpm
Torque	0	100	100	%
Acceleration ramp	3380	26	26	ms
Deceleration ramp	3380	26	26	ms
Receive angle 1	0	70	70	°
Receive angle 2	0	140	140	°
Receive angle 3	0	210	210	°
Release angle 1	0	280	280	°
Release angle 2	0	350	350	°
Wait angle 1	0	0	0	°
Wait angle 2	0	0	0	°
Pre turn delay	0	0	0	ms
During turn delay	0	0	0	ms
Post turn delay	0	0	0	ms
Interlock delay(on)	0	0	0	ms
Interlock delay(off)	0	0	0	ms
Limit switch wait	0	0	0	ms

All of the parameters in the settings windows are updated to the unit. If some of these parameters should be changed this could be done via the Parameter Setting Tool.

There are two was available to get Lua script on the unit. Either *Tools / Activities / Lua file handling* or *Tools / Activities / Lua editor. Lua file handling* requires USB connection while *Lua editor* requires ethernet connection.

Lua file handling

Lua file handling is available with USB connection in menu *Tools / Activities / Lua file handling*. No lua script is running while this panel is active.

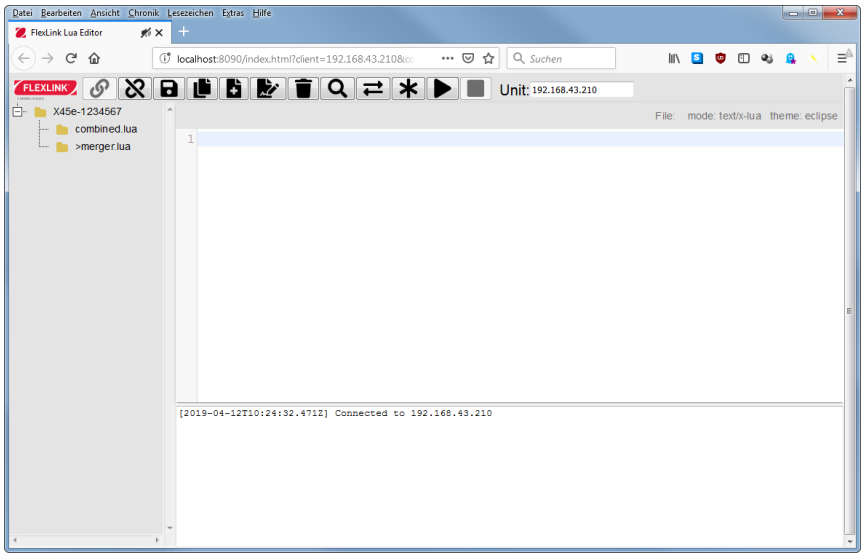


Here you see the list of Lua scripts stored at the unit. The used (default) script is written bold. You can send Lua files form computer to unit or read selected file from unit to computer. It is also

possible to delete selected file or set it as default. Password might be required.

Lua editor

Lua editor is a web-application served in PST and running in your browser. Computer and unit have to be in same Ethernet network. Lua editor can be started in menu *Tools / Activities / Lua editor.* Password might be required.



Lua editor connects automatically with unit and stops running Lua script. Here you see the list of Lua scripts stored at the unit. The used (default) script is maked with '>'. You can create new files at unit, modify existing files, delete or rename files. You can change the default scrip or run a script for testing. Editor supports auto completion (CTRL + SPACE) containing standard Lua API and X45e related API. Console printings from script are shown in logging area (lower area of editor). Editor supports auto completion (CTRL + SPACE) containing standard Lua API and X45e related API.

Lua API

Additional to standard API of Lua, X45e has an own API. It is supported in Lua editor.

General information

Lua script will be called once at end of initialization of unit. This should be used to initialize variables. Script can register "execute" function which will be called periodically.

You can get time information from status library. Function get_time_ms returns system time in milliseconds. Function get_counter of each state machine returns time in current state in milliseconds.

Library dio

This library gives control over digital inputs and outputs.

DIO functions

- **create - returns access to pin with defined name**
- **get_pin_names - returns list of available pin names**

Metatable pin

Pin is used to read/write a digital input or output.

Pin functions

- **get - returns current state of pin**
- **set - set new state to pin**

Available inputs and outputs

Name	Connector	Contact
DIGITAL_IN_1	X1	IN 1
DIGITAL_IN_2	X2	IN 1
DIGITAL_IN_3	X3	IN 1
DIGITAL_IN_4	X4	IN 1
DIGITAL_IN_5	X1	IN 2
DIGITAL_IN_6	X2	IN 2
DIGITAL_IN_7	X3	IN 2
DIGITAL_IN_8	X4	IN 2
DIGITAL_OUT_1	X1	OUT
DIGITAL_OUT_2	X2	OUT
DIGITAL_OUT_3	X3	OUT
DIGITAL_OUT_4	X4	OUT

Example code

```
pin = dio.create('DIGITAL_OUT_1')
pin:set(1)
state = pin:get()
```

Library config

This library gives access to configuration parameters.

Config functions

- **create - returns access to parameter with defined name**
- **get_parameter_names - returns list of available parameter names**

Metatable parameter

Parameter is used to read current value of configuration parameter.

Parameter functions

- **get - returns current value of parameter**

Available parameter

- **MOTOR_TYPE**
- **FUNCTION_MODE**
- **CALIBRATE**
- **NODE_ID**
- **SPEED_SET_POINT_2**
- **SPEED_SET_POINT_1**
- **TORQUE**
- **RECEIVE_ANGLE_1**
- **RECEIVE_ANGLE_2**
- **RECEIVE_ANGLE_3**
- **RELEASE_ANGLE_1**
- **RELEASE_ANGLE_2**
- **WAIT_ANGLE_1**
- **WAIT_ANGLE_2**
- **ACCELERATION_RAMP**
- **DECELERATION_RAMP**
- **ENABLE_DRIVE**
- **RESET_ERRORS**
- **DEBUG_MODE**
- **FUNCTION_PARAM_1**
- **FUNCTION_PARAM_2**
- **FUNCTION_PARAM_3**
- **FUNCTION_PARAM_4**
- **REQUEST_ANGLE**
- **PRE_TURN_DELAY**
- **DURING_TURN_DELAY**
- **POST_TURN_DELAY**
- **AUTOMATIC_RESET_ERRORS**
- **INTERLOCK_MODE**
- **INTERLOCK_DELAY_ON**
- **INTERLOCK_DELAY_OFF**
- **ENABLE_DRIVE_RESET**
- **ABUS_NETWORK_PROTOKOL**
- **FEG_PROGRAM_MAJOR_VERSION**
- **FEG_PROGRAM_MINOR_VERSION**
- **FEG_PROGRAM_PATCH_VERSION**
- **OPERATING_MOTOR_POWER**
- **OPERATING_MOTOR_CURRENT**
- **OPERATING_MOTOR_VOLTAGE**
- **OPERATING_SECONDS**
- **OPERATING_MINUTES**
- **OPERATING_HOURS**
- **OPERATING_DAYS**

- **OPERATING_YEARS**
- **OPERATING_CYCLES_HIGH**
- **OPERATING_CYCLES_LOW**
- **POSITIONER_MODE**
- **POSITIONER_RATIO**
- **FUNCTION_GROUP**
- **FUNCTION_VARIANT**
- **ECO_PROGRAM_VERSION**
- **TEMPERATURE**
- **DEBUG_ENABLE**
- **LIMIT_SWITCH_WAIT**
- **POS_REQ_ABSOLUTE_CLOSEST**
- **POS_REQ_ABSOLUTE_CCW**
- **POS_REQ_ABSOLUTE_CW**
- **POS_REQ_RELATIVE**
- **ECO_CURRENT_ANGLE_CCW**
- **ECO_CURRENT_ANGLE_CW**
- **IP_HANDLING_AUTOMATIC**
- **IP_ADDRESS**
- **IP_SUBNET_MASK**
- **IP_GATEWAY**
- **IP_DNS_SERVER**

Example code

```
parameter = config.create('FUNCTION_PARAM_1')
value = parameter:get()
```

Library status

This library gives access to rated and current status of control and motor part as well as to state machines.

Status functions

- **create_state - returns access to state with defined name**
- **get_state_names - returns list of available state names**
- **create_statemachine - returns access to state machine with defined name**
- **get_statemachine_names - returns list of available state machine names**
- **get_time_ms - returns current system time in milliseconds**

Metatable state

State is used to read rated or current state of control or motor part. Each state contains a list of values, while a value may consist of a number of bits.

State functions

- **get - returns state as list of values**

Available states

- **CURRENT_STATE**
  - **Digital in (uint8)**
    - **Digital in 1 (bit 0)**
    - **Digital in 2 (bit 1)**
    - **Digital in 3 (bit 2)**
    - **Digital in 4 (bit 3)**
    - **Digital in 5 (bit 4)**
    - **Digital in 6 (bit 5)**
    - **Digital in 7 (bit 6)**
    - **Digital in 8 (bit 7)**
  - **Status (uint8)**
    - **Function state (bit 0 - 2)**
    - **Wait (bit 3)**
    - **In position (bit 4)**
    - **Reverse direction (bit 5)**
    - **Drive enabled (bit 6)**
    - **Errors cleared (bit 7)**
  - **Position (uint8)**
    - **Release angle 1 (bit 0)**
    - **Release angle 2 (bit 1)**
    - **Receive angle 1 (bit 2)**
    - **Receive angle 2 (bit 3)**
    - **Receive angle 3 (bit 4)**
    - **Wait angle 1 (bit 5)**
    - **Wait angle 2 (bit 6)**
  - **Error (uint8)**
    - **Over current (bit 0)**
    - **High UDC (bit 1)**
    - **Low UDC (bit 2)**
    - **PCB over temperature (bit 3)**
    - **Locked rotor (bit 4)**
    - **Internal fault (bit 5)**
    - **Output overload (bit 6)**
- **RATED_MOTOR_STATE / CURRENT_MOTOR_STATE**
  - **Speed (int16)**
  - **Torque (int16)**
  - **Position (int32)**
  - **State (uint8)**
    - **Power enable (bit 0)**
    - **Power disable (bit 1)**

        - **Drive enable (bit 2)**
        - **Drive disable (bit 3)**
        - **Break enable (bit 4)**
        - **Break disable (bit 5)**
        - **Encoder enable (bit 6)**
        - **Encoder disable (bit 7)**
    - **Operational mode (uint8)**
        - **No operation (0)**
        - **Run (5)**
        - **Calibrate (6)**
        - **Speed (9)**
    - **Error (uint16)**
        - **Over current (1)**
        - **High UDC (5)**
        - **Low UDC (6)**
        - **PCB over temperature (8)**
        - **Locked rotor (10)**
        - **Internal fault (15)**
    - **Encoder angle (uint16)**

Metatable statemachine

Statemachine is used to read and write current state of functional parts of motor. Each statemachine contains current state and time since state change. It is recommended to use MOTOR_STATEMACHINE at drive-unit and FUNCTION_STATEMACHINE at function-unit.

Statemachine functions

- **get_counter - returns millisecond counter since last state change**
- **get_state - returns current state of this state machine**
- **set_state - sets new state to this state machine**

Available state machines

State machine	State name	State id
MOTOR_STATEMACHINE	STATE_RUNNING	1
	STATE_STOPPED	2
	STATE_FAULT	3
ERROR_STATEMACHINE	ERROR_STATE_CHECKING	1
	ERROR_STATE_CLEAR_LOW_UDC	2
	ERROR_STATE_WAIT	3
FUNCTION_STATEMACHINE	FUNCTION_STATE_RESCUE_CONTINUES	0
	FUNCTION_STATE_INIT	1
	FUNCTION_STATE_PRE_TURN	2
	FUNCTION_STATE_DURING_TURN	3
	FUNCTION_STATE_POST_TURN	4
	FUNCTION_STATE_CALIBRATE	5
	FUNCTION_STATE_IDLE	6
	FUNCTION_STATE_FAULT	7
	FUNCTION_STATE_DELAY	17

Example code

```
state = status. create_state ('CURRENT_STATE')
list = state:get()
error = list[4]
functionStateMachine = status.create_statemachine('FUNCTION_STATEMACHINE')
functionState = functionStateMachine:get_state()
```

Library control

This library allows to set speed, direction and angle to motor controller.

Control functions

- **create - returns access to feature with defined name**
- **get_feature_names - returns list of available feature names**

Metatable feature

Feature is used to set speed, direction and angle to motor controller.

Feature functions

- **set - set new value to feature**

Available features

- **SPEED - motor speed between -360 and 360 rpm, recommended speed for drive motor is -320 to -80 and 80 to 320**
- **DIRECTION - next turn direction of function motor: 0 means clockwise; 1 means counterclockwise**
- **ANGLE - next target angle of function motor between -360 and 360 degree**

Example code

```
feature = control.create('SPEED')
feature:set(150)
```

Library console

This library allows to print information to console of Lua editor.

Console functions

- **print - prints the string**

Example code

```
state = 1
con.print(“state: ”..state)
```

Library utility

This library provides helpful utilities.

Utility functions

- **sleep - sleeps for x milliseconds**
- **angle_for_position - calculates angle in degree from motor position**
- **direction_closest_distance - returns turn direction to move closest distance between to angles**
- **in_position - returns if two angles are close enough to say position is reached**

Example code

```
-- sleep 100 ms --
utility.sleep(100)

targetAngle = 90
motorPosition = 12345
motorAngle = utility. angle_for_position(motorPosition)
inPosition = utility.in_position(motorAngle, targetAngle)
direction = utility.direction_closest_distance(motorAngle, targetAngle)
```