RabbitMQ

Last Modified on 28/08/2019 10:38 am CEST

Background

The X45e units (Ethernet version) support RabbitMQ. RabbitMQ is a communication which uses a server as message broker instead of a direct connection. It can easily be handled by computer. RabbitMQ is running parallel to normal behavior of X45e unit. It is also possible to control X45e unit via RabbitMQ (neither Lua nor a network protocol active). A time server (PTP) is required for synchronization.

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**
- **Ethernet cables**

Software

- **X45e - Parameter Setting Tool**
- **RabbitMQ server**

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 8.2.9 was the latest version. The PST version used is 3.4.3.

Configuration

After configuring the unit using the wizard you have to open advanced settings to configure RabbitMQ settings.

- **RabbitMQ mode - Mode of RabbitMQ handling (0=off, 1=listen only, 2=control (neither Lua nor a network protocol active)**
- **RabbitMQ server name - Name or IP-address of RabbitMQ server**
- **RabbitMQ server port - Port at RabbitMQ server**

- **RabbitMQ exchange - Exchange at RabbitMQ server**
- **RabbitMQ device name - Name of device in routing key of RabbitMQ messages**
- **RabbitMQ runtime send freq. - Send frequency of runtime messages in seconds**
- **RabbitMQ power debounce - Debounce for sending messages at changed power in percent (drive unit only)**

Structure of routing keys

All routing keys starting with "x45e...". Where means that drive motor has 'd' and function motor has 'f'. is MAC-address of unit while is the device name from configuration. Routing keys of messages to unit have additional "recv.". Routing keys are completed by ..

Examples

x45ef.A0:28:33:B8:00:01.function.din.4
x45ef.A0:28:33:B8:00:01.function.recv.position_set.0

Send messages

Digital input

Routing key: din.
Data:
{
"value": 1, (is sensor active)
"ts": 1527758948000 (time stamp)
}

Motor state

Routing key: state_get.0
Data:
{
"state": 5, (3 bit state *)
"wait": 0, (is in waiting state)
"in_position": 1, (is in position)
"reverse_dir": 0, (is motor running backward)
"enbl": 1, (is motor enabled)
"err_clrd": 0, (are errors cleared)
"ts": 1527758948000 (time stamp)
}

* States:
0 - Rescue
1 - Init (F), running (D)

2 - Pre turn (F), stopped (D)
3 - During turn (F), fault (D)
4 - Post turn (F)
5 - Calibrate (F)
6 - Idle (F)
7 - Fault (F)

Motor position (function only)

Routing key: position_get.0
Data:
{
"rel_1": 0, (is at release position 1)
"rel_2": 0, (is at release position 2)
"rec_1": 0, (is at receive position 1)
"rec_2": 1, (is at receive position 2)
"rec_3": 0, (is at receive position 3)
"wait_1": 0, (is at wait position 1)
"wait_2": 0, (is at wait position 2)
"ts": 1527758948000 (time stamp)
}

Motor error

Routing key: error.0
Data:
{
"id": [2, 100], (IDs of errors *)
"desc": ["high_UDC", "no_timesync"], (clear names of errors *)
"ts": 1527758948000 (time stamp)
}

* IDs:
0 - OK
1 - over current
2 - high UDC
3 - low UDC
4 - PCB over temperature
5 - locked rotor
6 - internal fault
7 - output overload
100 - no time sync

Motor power (drive only)

Routing key: power.0
Data:

```
{
"value": 1.02, (motor power of drive motor in W)
"ts": 1527758948000 (time stamp)
}
```

Motor runtime

Routing key: runtime.0
Data:

```
{
"year": 0, (years of runtime)
"day": 23, (days of runtime)
"hour": 14, (hours of runtime)
"min": 43, (minutes of runtime)
"sec": 21, (seconds of runtime)
"ts": 1527758948000 (time stamp)
}
```

Motor cycles (function only)

Routing key: cycles.0
Data:

```
{
"value": 102, (cycles of function motor)
"ts": 1527758948000 (time stamp)
}
```

Receive messages

Digital output

Routing key: dout.
Data:

```
{
"value": 1, (set output active)
}
```

State

Routing key: state_set.0
Data:

```
{
```

```
"reverse_dir": 0, (run motor backwards)
"enbl": 1, (enable motor)
"clr_err": 0, (clear errors)
}
```

Position

Routing key: position_set.0
Data:

```
{
"rel_1": 0, (go to release position 1)
"rel_2": 0, (go to release position 2)
"rec_1": 0, (go to receive position 1)
"rec_2": 1, (go to receive position 2)
"rec_3": 0, (go to receive position 3)
"wait_1": 0, (go to wait position 1)
"wait_2": 0, (go to wait position 2)
}
```

Cmd

Routing key: cmd.0
Data:

```
{
"cmd": ""
}
```

Known commands:
"refresh" - send all status information