Master-slave connections between DOMIQ/Base modules

Update 1.8.9 introduced functionality that allows to create master-slave connections between **Base** modules. This concept was designed for installations where there are at least two **DOMIQ/Base** modules and it is required to control all modules with common user interface.

1. Master-slave connection characteristics

1. Master has access to state of all devices and variables available in slave.
2. Master receives notifications about changes of state in slave.
3. Master has access to functions from the **Logic** tab in slave.
4. Master can send any command to slave.
5. A single **Base** module can operate as master for several **Base** modules.
6. **Base** module can work as master and as slave for the other **Base** module at the same time. However, notifications about changes of state are passed only one level up in the hierarchy. **Example**: let's assume that there are three **Base** modules with the following names: *Base1*, *Base2* and *Base3*. *Base2* works as master for *Base1* and as slave relating to *Base3*. In such configuration, the *Base3* only receives information about changes of state of variables and devices connected directly to the *Base2*.
7. Master will automatically synchronize state of slave in case of restart of or when reconnected to LAN after a period longer than 30 seconds.
8. Similarily, master will automatically synchronize state of slave in case of restart of slave or when slave is reconnected to LAN after a period longer than 30 seconds.
9. In case of disconnection from LAN for periods shorter than 30 secons, master will synchronize state after the first change of state in the slave module.
10. Do not create events in a master module, that are triggered by changes of state in slave with value of 0. Such events will be false triggered due to reset of state during reboot of a slave module.

Characteristics presented above allows to create one common user interface (visualization and/or **Remote** menu) for all **Bases** working in the same installation. Moreover a master module can send commands to slave as a reaction to an event, timer, as a result of a function in the **Logic**, etc.

In general, master-slave connectons allows to control slave in the way as if everything was done within a single **Base** module.

2. Configuration of a connection

Connections are defined in the **Links** tab. Connection have to be created on both ends: in a master and in a slave. The procedure is as follows:

Master

1. Add a connection to the slave module by clicking on the **Add slave** button.
2. In the **Name** enter a connection name. Name can contain letters from a to z (capital letters as well), digits and the underscore character. **Name must be exact the same on both ends of the connection.**
3. In the **Address** field enter the IP address of the slave.
4. The remaining fields are optional:
    1. In the **Description** field you can enter description for this connection.
    2. In the **Port** field enter port number, which will be used to establish connection. Modify only when necessary.
    3. In the **Password** field you can enter a password to encrypt the communication. Due to the fact that **Base** modules work in the same LAN, encryption can be omitted. Encryption slightly slows down the communication between **Base** modules.
5. Save the **Links** tab.

Slave

Configuration of the slave module is similar.

1. Add a connection to the master by clicking on the **Add master** button.
2. In the **Address** field enter the IP address of the master module.
3. Repeat the remaining steps as for the master module configuration.

3. How to use

When connection is ready, the master **Base** will automatically fetch state of the slave. All identifiers of the slave module will be visible in the state of master module with the following prefix: `MNT.<configuration_name>`. Let's assume that the created connection has the name: *office_production* (this name is used in all examples), then whole state of the slave module will be visible with the following prefix: `MNT.office_production.<regular_identifier>`. When state of a slave is available in a master, then slave can be controlled, just like all the other devices and variables that are available locally in the master module. Study the examples:

1. **Dimmer in the Remote** – Let's assume that you want to control a dimmer connected to a slave module. The identifier of the dimmer is: `LCN.output.0.10.1`. In order to control it using master, in the **Remote** tab, please add a **Dimmer** element. In the **Channel** field enter: `MNT.office_production.LCN.output.0.10.1`.
2. **Light on visualization** – In this example let's assume that you want to control the same output as in the previous example. Add a **Light** element to visualization and in the **Channel** field type: `MNT.office_production.LCN.output.0.10.1`

3. **Command as action in an event** – Let's assume that a command is sent to a slave module as the reaction to an event in a master module. The command is switching of a relay in the LCN (relay no. 2 in a LCN module with the address 10). After setting the conditions of triggering of the event, in the **Actions** section click on the **Add command...** button. In the **Name** field enter: `C.MNT.office_production.LCN.relay.0.10.2` and in the **Value** field enter: `toggle`. Similarly, you can configure action, that could be executed by a timer.
4. **Triggering an event** – Change of state in a slave can trigger an event in a master module. Let's assume that the observed change of state is arming of the alarm zone no. 1. The configuration of an event in the master module should be as follows:
   Channel: `E.MNT.office_production.IDS.armed.1`
   Data: `1`
   In the **Actions** section you can define any commands to be executed when the event is triggered.
5. **Calling function from the Logic tab** – Master can call a function from the **Logic** in the slave module. A function can be called as an effect of each of the above described cases. We will present a function call as a result of an event.
   After setting the conditions of triggering of the event, in the **Actions** section click on the **Add command...** button. In the **Name** field enter: `C.MNT.office_production.LOGIC` and in the **Value** field enter the name of the function you want to invoke and optionally pass arguments to it. For example: `myFunc(argument1, argument2).`