TASKS FOR USER STUDY „MAINTAINING HIERARCHICAL CONTROL APPLICATIONS WITH 4DIAC IDE"

The moderator reads the tasks aloud.

Preparations

Moderator guides the subject through the import of the provided IEC 61499 project.
`File / Open Projects from File system / Directory … / Finish`

Moderator explains the setting to the subject:
Maintenance work is required in an existing plant. The motor of one conveyor belt is broken and needs to be replaced. The installed model is no longer available and a newer model of the motor has to be installed. Therefore, changes to the control software are required.

1. Orienting in an unknown application

1. Get an overview of the application.
    - For the middle capping station, find the application part that controls the motor of the conveyor belt.
    - Can you find any other control code for a motor in the project?
2. Please look at the contents of the subapplication for the middle capping station.
   In this subapplication: Which FB is connected to the event INITO of `Index`?

2. Creating/Removing hierarchies

1. Now, navigate to the left capping station. Create a subapplication for the motor control of the conveyor belt from the FBs `MotorRunning`, `ConveyorMotorCtrl`, `E_RTimeOut`. Assign a descriptive instance name for the subapplication.
2. You notice that the FB `MotorOn` should also be a part of the subapplication. Add it. Confirm that all inputs and outputs are still correctly connected, both internally and externally.
3. FB `MotorRunning` is an untyped subapplication that contains only a single FB. Verify that, and then remove this unnecessary grouping.

3. Working with types

1. You want to keep the old control code for further reference. For this reason, save the ConveyorMotor subapplication in your project library. The new file location has to be in Type Library / new folder named Motor.
2. Your colleague informs you that she has recently created control software for the same motor. She promises to share the subapplication with you. (You have received the file

`MotorM1499.SUB` via email.) Save also this subapplication in your project library, also in the folder Motor.

3. Please replace now your own subapplication for the `ConveyorMotor` with the one that you received from your colleague. Please retain the instance name, the data connections, and the event connections.

4. Editing

We currently have an instance of type `MotorM1499` and need to perform changes locally.

1. Detype the subapplication.
2. Edit the interface:
   - For the concrete use in your Application, parameters are missing at the interface of this instance. We will create the corresponding pins and connect them to the network.
   - Add ten data input pins named Param1 – Param10 to the interface.
     The required data types are 5x `INT`, 3x `BOOL`, 2x `CTRL` (Structured Type)).
   - Find out which variables are contained in the data type `CTRL`.
3. Insert FBs of the following types: `rpm`, `slip`, `TempSensor`
4. We will now create the required connections inside and outside the subapplication:

Connecting the interface of the subapplication ConveyorMotor.

Destination (ConveyorMotor)		**Source**	
Destination: Pin name	**Pin Type**	**Source: FB [Type]**	**Source: Pin name**
INIT	Event	Interface of parent subapplication	INIT
Param1	Integer (rpm)	Rpm	RD
Param2	Integer (slip)	Slip	RD
Param3	Integer (voltage)	-	Constant: 400
Param4	Integer (temperature)	TempSensor	RD
Param5	Integer (max temp)	-	Constant: 120
Param6	BOOL	-	Constant: TRUE

Tabelle 1: Required configurations for the pins of the subapplication ConveyorMotor.

Connect the events as follows:
- Pin INIT of the interface is connected with rpm.
- Pin INITO of FB rpm is connected with INIT of slip.
- Pin INITO of FB slip is connected with INIT of TempSensor.
- Edit the existing connection between INIT and ConveyorMotor: Connect INITO of TempSensor with INIT of ConveyorMotor.

Now you may apply a layout to the subapplication.