HOWTO use
CODESYS® on Tx/Cx Controller

Raspberry Pi is a trademark of the Raspberry Pi Foundation.
More information under www.raspberrypi.org

it´s all about perfection

STRIVE IN PERFECTION
IN WHATEVER YOU
DO
TAKE THE BEST THAT
EXISTS AND MAKE IT
BETTER
WHEN IT DOES NOT
EXIST. DESIGN IT.

Sir Henry Royce

CONTENT

In this application note you will find ...

PREREQUISITES

Raspberry Pi is a trademark of the Raspberry Pi Foundation.
More information under www.raspberrypi.org

PREREQUISITES

We assume that the reader is familiar how to use WINDOWS® operating system, how to configure a LINUX® Ethernet interface, how to use a remote desktop program or SSH console to configure LINUX®. Also we assume that the reader is able to install and open the CODESYS® IDE on a PC.

Furthermore we assume, that the reader is able to create a correct CODESYS® program. In special the reader is familiar how to create and write a STRUCTURED TEXT program in CODESYS®. If not, please consult the internet or book a education workshop. RESI is in no way responsible, if you or your customer cannot use the given advice here, because of lack of education in your or their staff!

With the purchase of a IoT Controller from RESI, you have not purchased the right of free education or free consulting from RESI!

We want to mention explicit, that CODESYS® has changed it's licensing method. Please refer to their homepage for more information how this affects your projects.

RESI delivers IoT controllers with the ability to run CODESYS® on it, but RESI is not liable for any functional problems, software errors, law suits or other issues which results out of using CODESYS® on our devices in your project or machinery!

IMPORTANT SAFETY NOTES
Important hint:

Before you start with the installation and the initial setup of the device, you have to read this document and the attached installation guide and the actual manual for the device very carefully. You have to follow all the herein given information very accurate!

- Only authorized and qualified personnel are allowed to install and setup the device!
- The connection of the device must be done in de-energized state!
- Do not perform any electrical work while the device is connected to power!
- Disable and secure the system against any automatic restart or power on procedure!
- The device must be operated with the defined voltage level!
- Supply voltage jitters must not exceed the technical specifications and tolerances given in the technical manuals for the product. If you do not obey this issue, the proper performance of the device cannot be guaranteed. This can lead to fail functions of the device and in worst case to a complete breakdown of the device!
- You have to obey the current EMC regulations for wiring!
- All signal, control and supply voltage cables must be wired in a way, that no inductive or capacitive interference or any other severe electrical noise disturbance may interfere with the device. Wrong wiring can lead to a malfunction of the device!
- For signal or sensor cables you have to use shielded cables, to avoid damages through induction!
- You have to obey and to apply the current safety regulations given by the ÖVE, VDE, the countries, their control authorities, the TÜV or the local energy supply company!
- Obey country-specific laws and standards!
- The device must be used for the intended purpose of the manufacturer!
- No warranties or liabilities will be accepted for defects and damages resulting from improper or incorrect usage of the device!
- Subsequent damages, which results from faults of this device, are excluded from warranty and liability!
- Only the technical data, wiring diagrams and operation instructions, which are part to the product shipment are valid!
- The information on our homepage, in our datasheets, in our manuals, in our catalogues or published by our partners can deviate from the product documentation and is not necessarily always actual, due to constant improvement of our products for technical progress!
- In case of modification of our devices made by the user, all warranty and liability claims are lost!
- The installation has to fulfill the technical conditions and specifications (e.g. operating temperatures, power supply, ...) given in the devices documentation!
- Operating our device close to equipment, which do not comply with EMC directives, can influence the functionality of our device, leading to malfunction or in worst case to a breakdown of our device!
- Our devices must not be used for monitoring applications, which solely serve the purpose of protecting persons against hazards or injury, or as an emergency stop switch for systems or machinery, or for any other similar safety-relevant purposes!
- Dimensions of the enclosures or enclosures accessories may show slight tolerances on the specifications provided in these instructions!
- Modifications of this documentation is not allowed!
- In case of a complaint, only complete devices returned in original packing will be accepted!

Install CODESYS®
on RESI-T4/C4 controller

Raspberry Pi is a trademark of the Raspberry Pi Foundation.
More information under www.raspberrypi.org

Install CODESYS® on RESI-T4/C4 controller

For this tutorial we use CODESYS V3.5 SP19 Patch 6.

IP SETTINGS FOR THE C4/T4 CONTROLLER

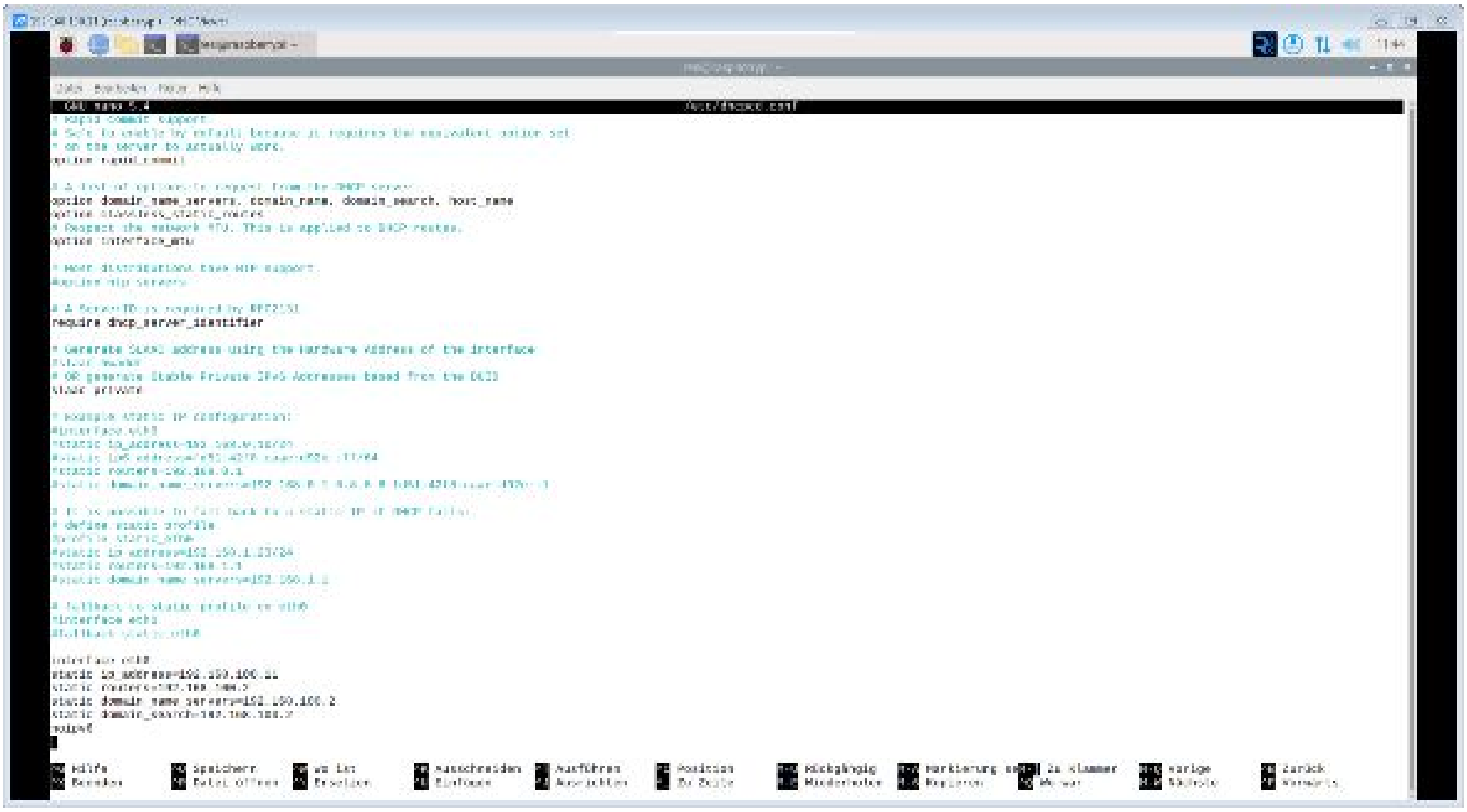
Use like we do VNC Viewer for Raspberry Pi to connect to the LINUX desktop on our preinstalled LINUX:

Open a shell and enter the command

```
sudo nano /etc/dhcpcd.conf
```

Install CODESYS® on RESI-T4/C4 controller

Scroll down to the static IP settings and change them to your needs. You will find more information about correct IP settings for Raspberry Pi, because the are so many possibilities in LINUX...



```
# Rapid commit support.
# Safe to enable by default because it requires the equivalent option set
# on the server to actually work.
option rapid_commit

# A list of options to request from the DHCP server.
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# Most distributions have NTP support.
#option ntp_servers

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate SLAAC address using the Hardware Address of the interface
#slaac hwaddr
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

interface eth0
static ip_address=192.168.100.11
static routers=192.168.100.2
static domain_name_servers=192.168.100.2
static domain_search=192.168.100.2
noipv6
```

After you have changed your settings, reboot your controller with

```
sudo reboot
```

Be aware: If you enter a false IP setting your controller will not be reachable anymore. The only way is to create a new SD-CARD with a plain Raspian LINUX or you use our standard image from our homepage to create a new SD-CARD.

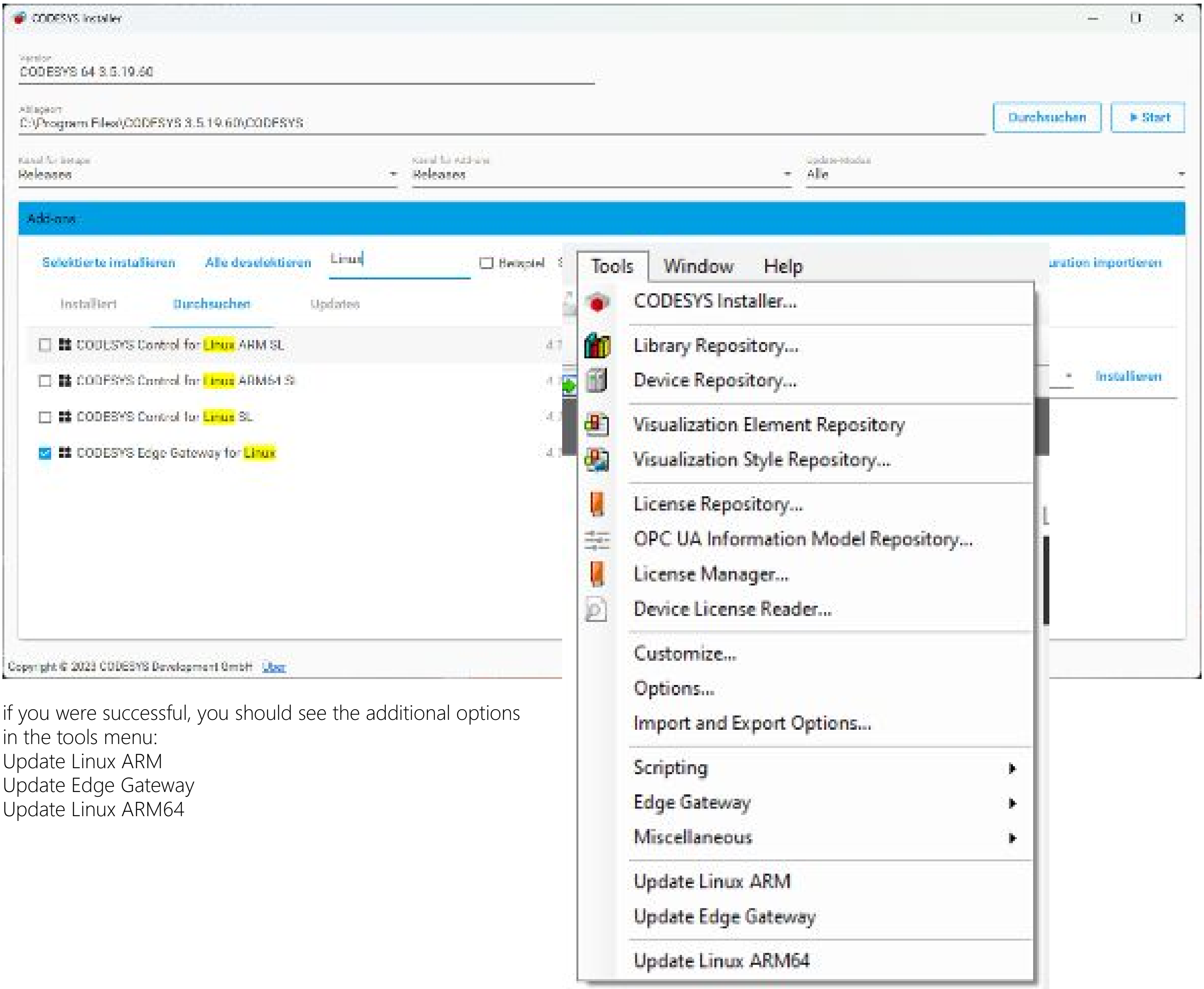
Check your new IP with the VNCViewer. If this runs properly, everything is fine.

Install CODESYS® on RESI-T4/C4 controller

INSTALL CODESYS ARM runtime

Before you can download the runtime for our controller, you have to update the CODESYS IDE to offer this possibility.
Check your Menu tool. Here you have to have the menu entries
Update Linux ARM
Update Linux ARM64
and
Update Edge Gateway

If not open the CODESYS Installer and install
CODESYS Control for Linux ARM SL for a 32 bit LINUX system (Our Image is 32 bit) or
CODESYS Control for Linux ARM64 SL for a 64 bit LINUX or both of them.
and CODESYS Edge Gateway for Linux
How you can do that consult CODESYS homepage...



Version: CODESYS 64 3.5.19.60

Ablageort: C:\Program Files\CODESYS 3.5.19.60\CODESYS

Kanal für Setups: Releases

Kanal für Add-ons: Releases

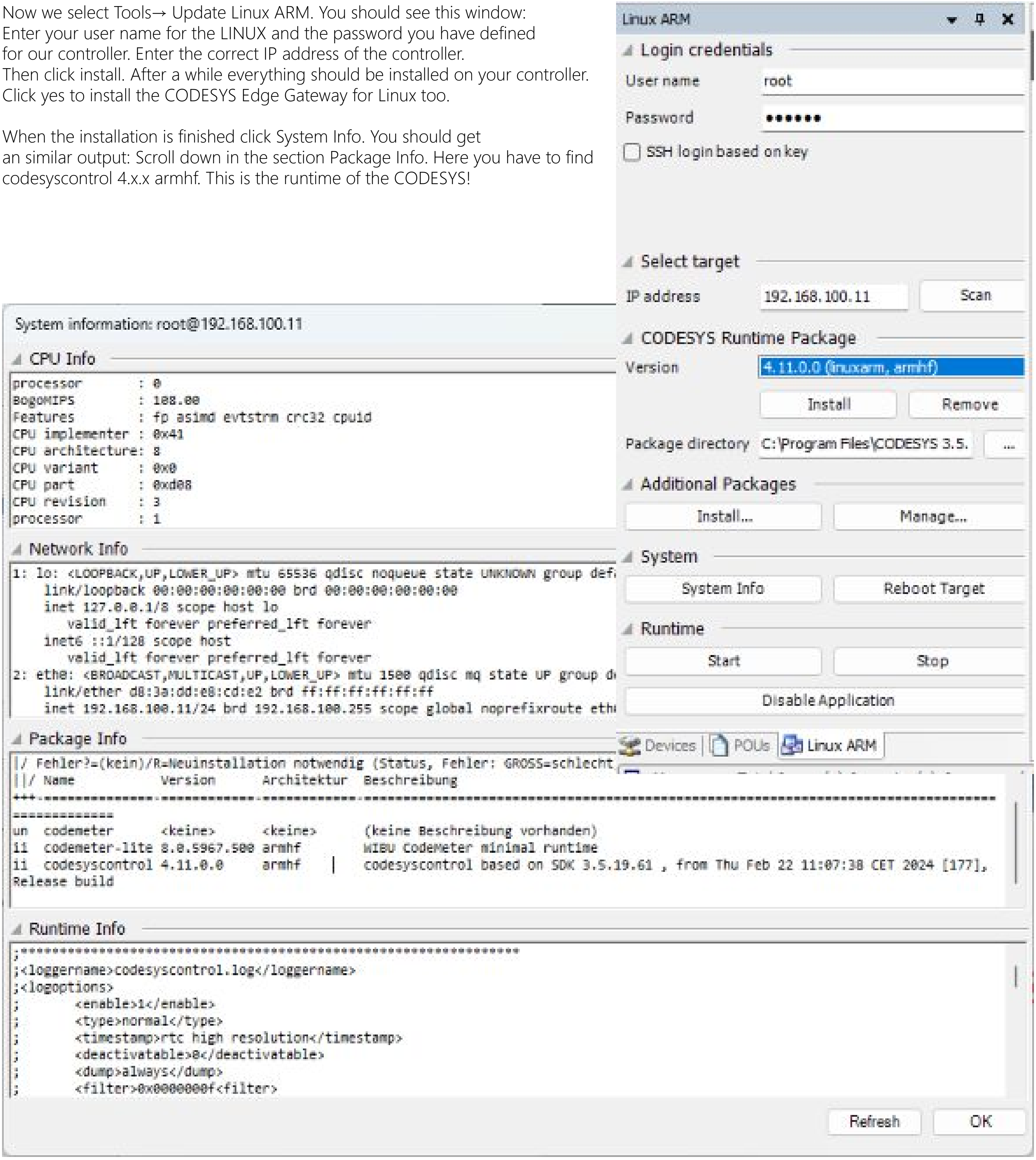
Update-Modus: Alle

if you were successful, you should see the additional options in the tools menu:
Update Linux ARM
Update Edge Gateway
Update Linux ARM64

Install CODESYS® on RESI-T4/C4 controller

Now we select Tools→ Update Linux ARM. You should see this window:
Enter your user name for the LINUX and the password you have defined for our controller. Enter the correct IP address of the controller.
Then click install. After a while everything should be installed on your controller.
Click yes to install the CODESYS Edge Gateway for Linux too.

When the installation is finished click System Info. You should get an similar output: Scroll down in the section Package Info. Here you have to find codesyscontrol 4.x.x armhf. This is the runtime of the CODESYS!



Login credentials

User name: root

Password: ••••••

☐ SSH login based on key

Select target

IP address: 192.168.100.11

CODESYS Runtime Package

Version: 4.11.0.0 (linuxarm, armhf)

Package directory: C:\Program Files\CODESYS 3.5.

System information: root@192.168.100.11

CPU Info

```
processor       : 0
BogoMIPS        : 108.00
Features        : fp asimd evtstrm crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd08
CPU revision    : 3
processor       : 1
```

Network Info

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defa
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group de
    link/ether d8:3a:dd:e8:cd:e2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.11/24 brd 192.168.100.255 scope global noprefixroute eth
```

Package Info

```
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht
||/ Name           Version        Architektur  Beschreibung
+++-==============-==============-============-==========================================================
=============
un  codemeter      <keine>        <keine>      (keine Beschreibung vorhanden)
ii  codemeter-lite 8.0.5967.500 armhf          WIBU CodeMeter minimal runtime
ii  codesyscontrol 4.11.0.0       armhf        codesyscontrol based on SDK 3.5.19.61 , from Thu Feb 22 11:07:38 CET 2024 [177],
Release build
```

Runtime Info

```
;*******************************************************************
;<loggername>codesyscontrol.log</loggername>
;<logoptions>
;        <enable>1</enable>
;        <type>normal</type>
;        <timestamp>rtc high resolution</timestamp>
;        <deactivatable>0</deactivatable>
;        <dump>always</dump>
;        <filter>0x0000000f<filter>
```

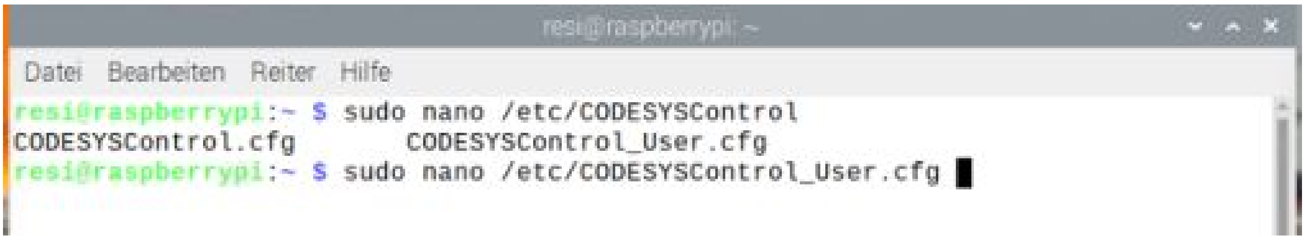
Raspberry Pi is a trademark of the Raspberry Pi Foundation.
More information under www.raspberrypi.org

Install CODESYS® on RESI-T4/C4 controller

Defining the correct serial interfaces for CODESYS

We have to tell CODESYS to use the correct serial interfaces if you open a serial device in CODESYS. Our controllers use dev/ttyACM0 to dev/ttyACM4 as new interfaces in LINUX. The amount of interfaces depend on the product:
COM1 of CODESYS will be dev/ttyACM0,
COM2 of CODESYS will be dev/ttyACM1,
COM3 of CODESYS will be dev/ttyACM2,
COM4 of CODESYS will be dev/ttyACM3

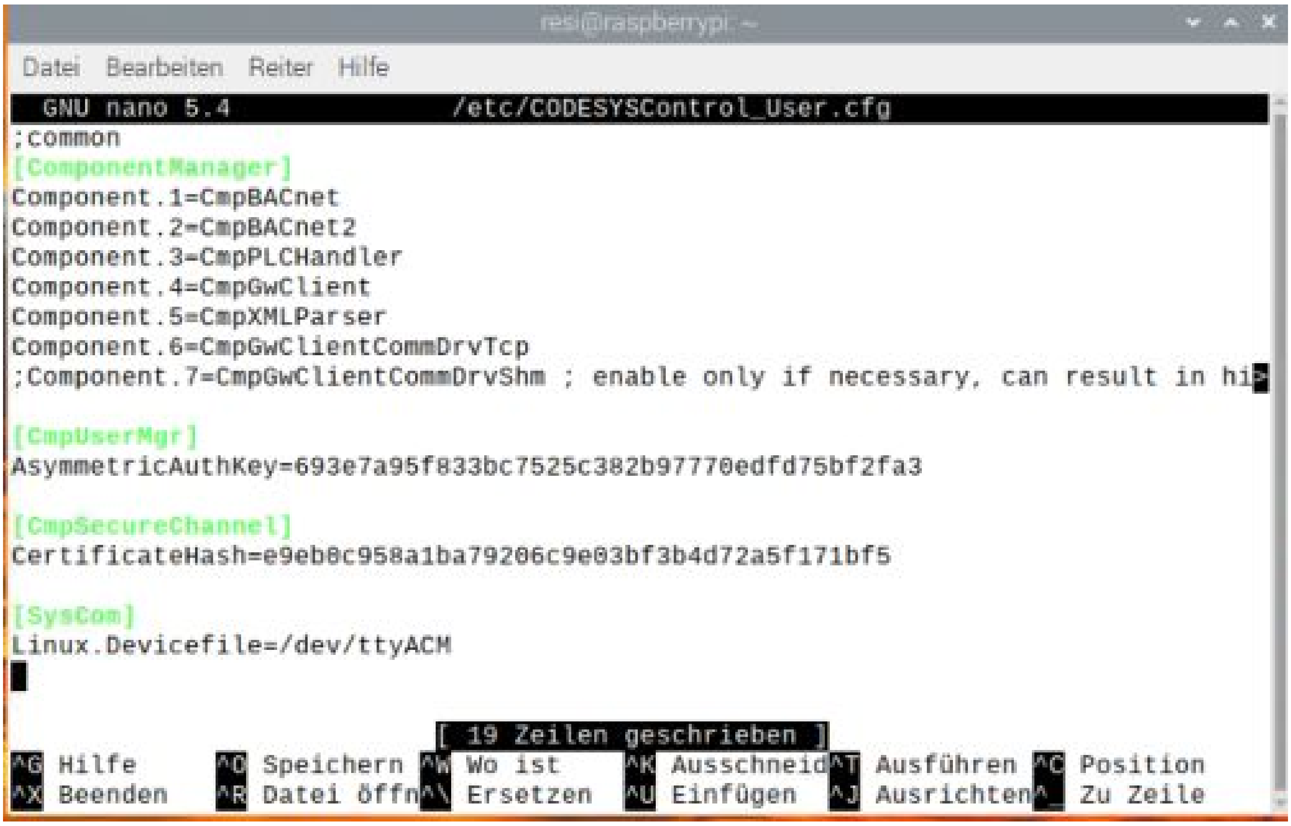
In the next step we open the VNCviewer again and we open a shell
Enter `sudo nano /etc/CODESYSControl_User.cfg`



```
resi@raspberrypi:~ $ sudo nano /etc/CODESYSControl
CODESYSControl.cfg       CODESYSControl_User.cfg
resi@raspberrypi:~ $ sudo nano /etc/CODESYSControl_User.cfg
```

Add the lines at the end of the file, save it and reboot the controller with `sudo reboot`

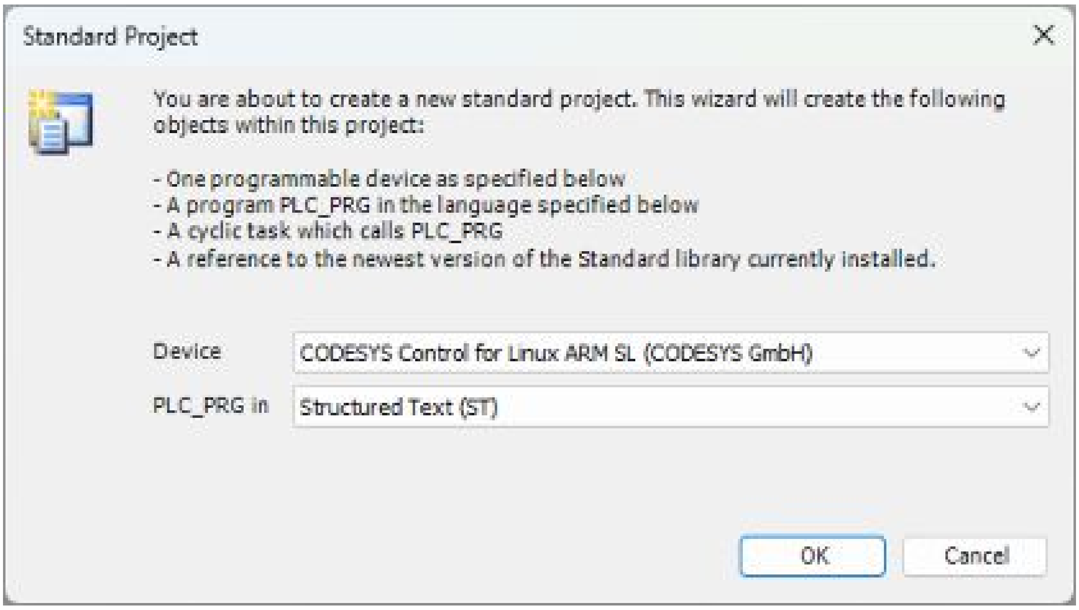
```
[SysCom]
Linux.Devicefile=/dev/ttyACM
```



```
  GNU nano 5.4                  /etc/CODESYSControl_User.cfg
;common
[ComponentManager]
Component.1=CmpBACnet
Component.2=CmpBACnet2
Component.3=CmpPLCHandler
Component.4=CmpGwClient
Component.5=CmpXMLParser
Component.6=CmpGwClientCommDrvTcp
;Component.7=CmpGwClientCommDrvShm ; enable only if necessary, can result in hi>

[CmpUserMgr]
AsymmetricAuthKey=693e7a95f833bc7525c382b97770edfd75bf2fa3

[CmpSecureChannel]
CertificateHash=e9eb0c958a1ba79206c9e03bf3b4d72a5f171bf5

[SysCom]
Linux.Devicefile=/dev/ttyACM


                              [ 19 Zeilen geschrieben ]
^G Hilfe      ^O Speichern  ^W Wo ist     ^K Ausschneid^T Ausführen  ^C Position
^X Beenden    ^R Datei öffn^\ Ersetzen    ^U Einfügen  ^J Ausrichten^_ Zu Zeile
```

Connect to IoT controller with CODESYS

Back in CODESYS we create a new standard project. Select CODESYS Control for Linux ARM SL and structured text (ST)



Standard Project

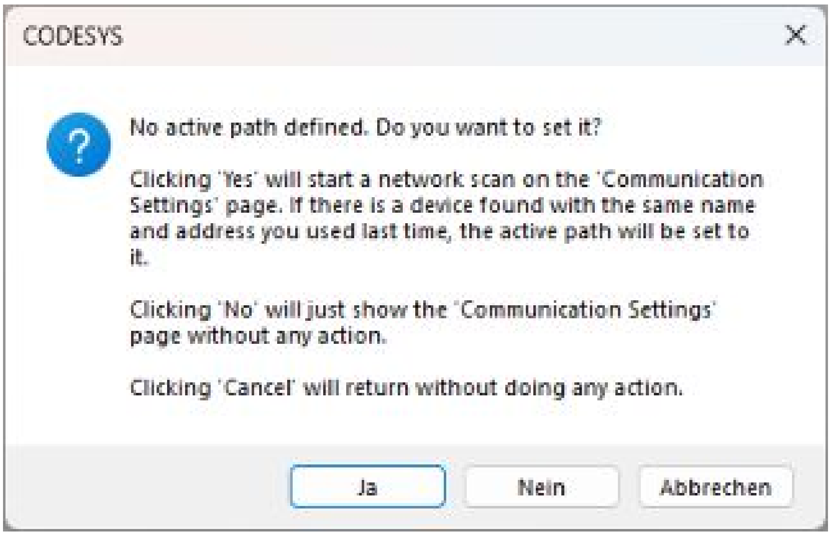
You are about to create a new standard project. This wizard will create the following objects within this project:

- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG
- A reference to the newest version of the Standard library currently installed.

Device: CODESYS Control for Linux ARM SL (CODESYS GmbH)

PLC_PRG in: Structured Text (ST)

After creating the empty program, select Online → Login.



CODESYS

No active path defined. Do you want to set it?

Clicking 'Yes' will start a network scan on the 'Communication Settings' page. If there is a device found with the same name and address you used last time, the active path will be set to it.

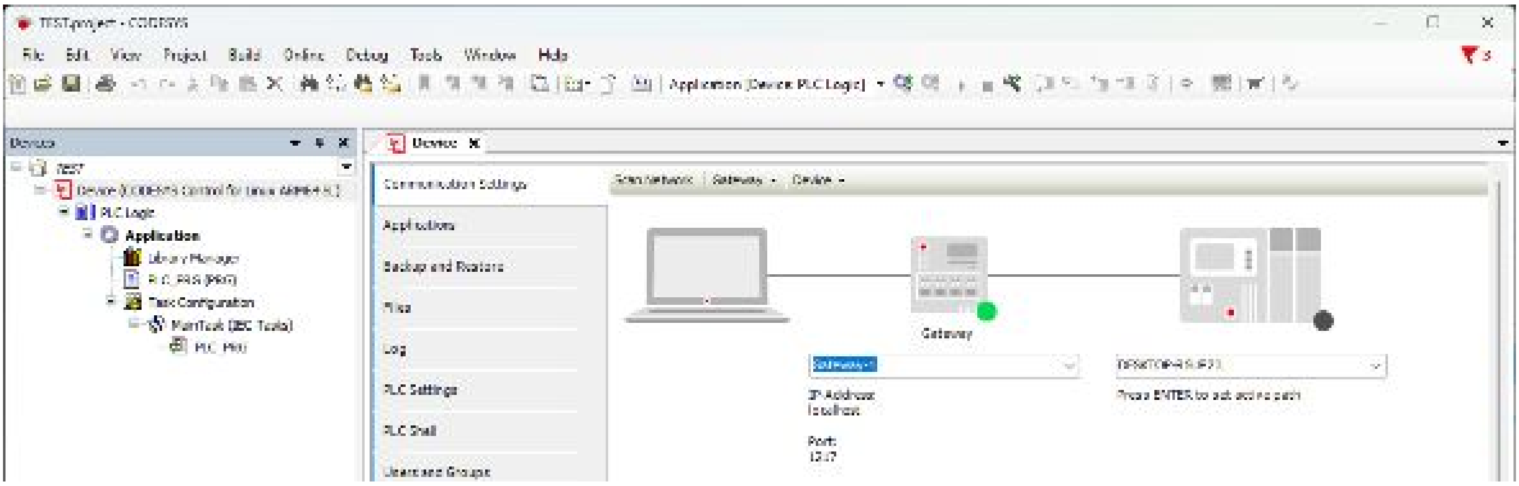
Clicking 'No' will just show the 'Communication Settings' page without any action.

Clicking 'Cancel' will return without doing any action.

Select NO in this dialog

Install CODESYS® on RESI-T4/C4 controller

You should see the following Communication Settings. Notice, that the gateway has a greed LED, which meas the connection works, but our device has a gray LED. Enter the IP address of our device (In our case 192.168.100.11:11740) beneath the device and hit ENTER. Use the socket 11740 after your IP address to successfully connect to your new device.



Gateway-1

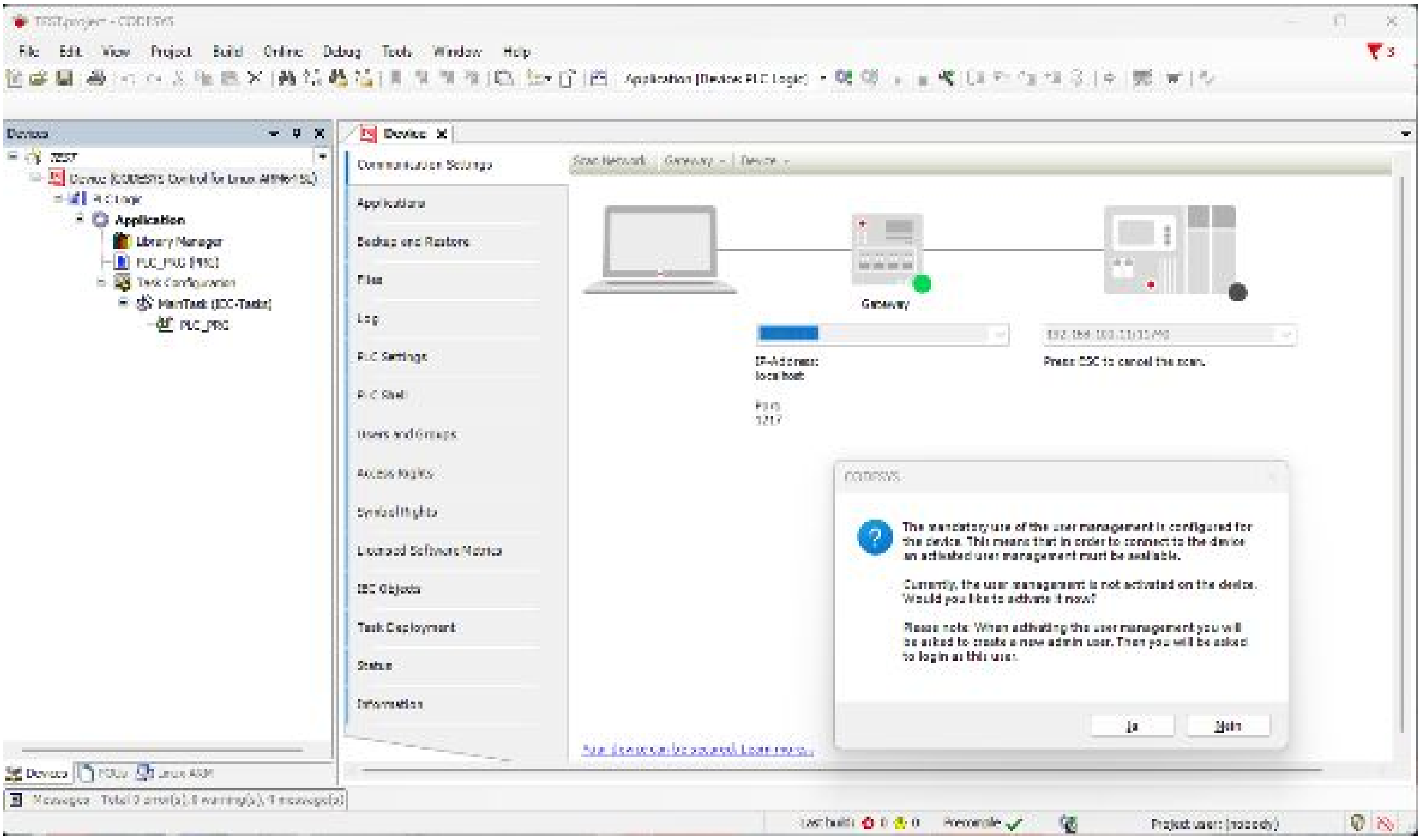
IP-Address:
localhost

Port:
1217

DESKTOP-R9UF71

Press ENTER to set active path

You will see the following screen. Answer the question for the user management with YES.



IP-Address:
localhost

Port:
1217

192.168.100.11:11740

Press ESC to cancel the scan.

The mandatory use of the user management is configured for the device. This means that in order to connect to the device an activated user management must be available.

Currently, the user management is not activated on the device. Would you like to activate it now?

Please note: When activating the user management you will be asked to create a new admin user. Then you will be asked to login as this user.

Messages - Total 0 error(s), 0 warning(s), 1 message(s)

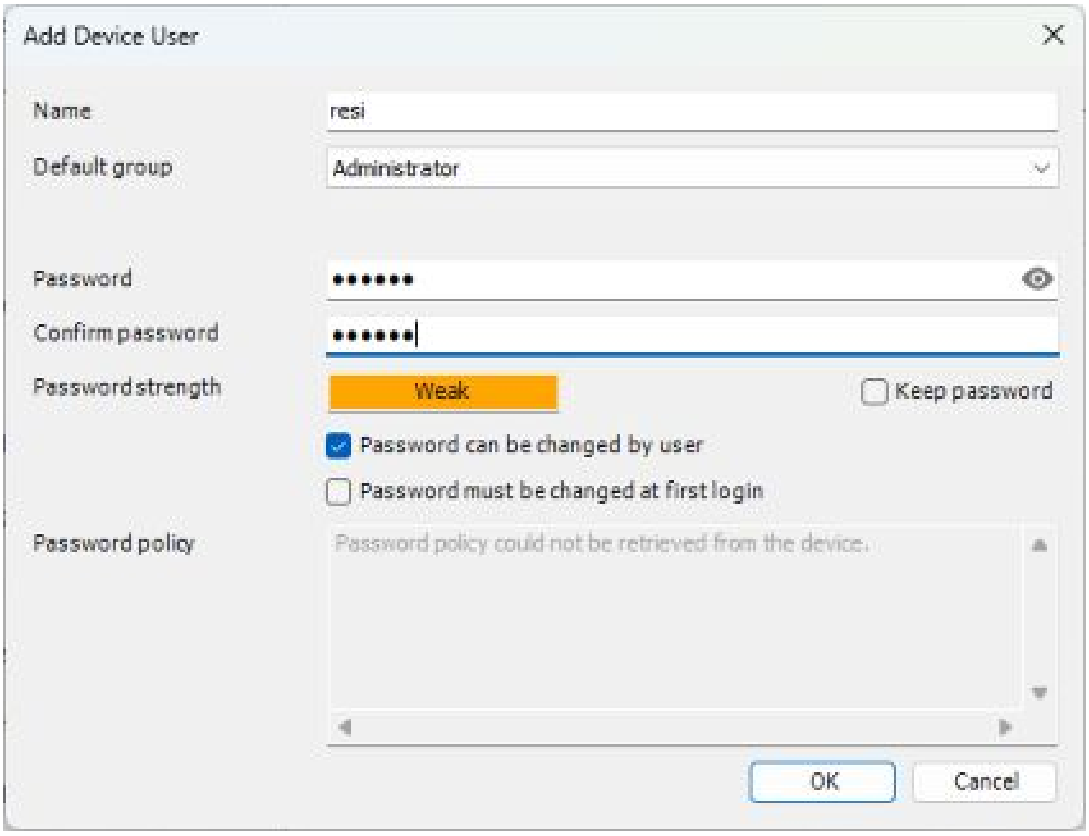
Last build: 0 0

Project user: (nobody)

Raspberry Pi is a trademark of the Raspberry Pi Foundation.
More information under www.raspberrypi.org

Install CODESYS® on RESI-T4/C4 controller

In the dialog enter a name and a password. BUT: If you forget this user name and the password you cannot access the device anymore. You really have to create a new SD-CARD with a empty CODESYS. Everything on your old system is lost. so
KEEP THIS INFORMATION SECURE ON A SAVE PLACE


Add Device User

Name: resi

Default group: Administrator

Password: ••••••

Confirm password: ••••••

Password strength: Weak

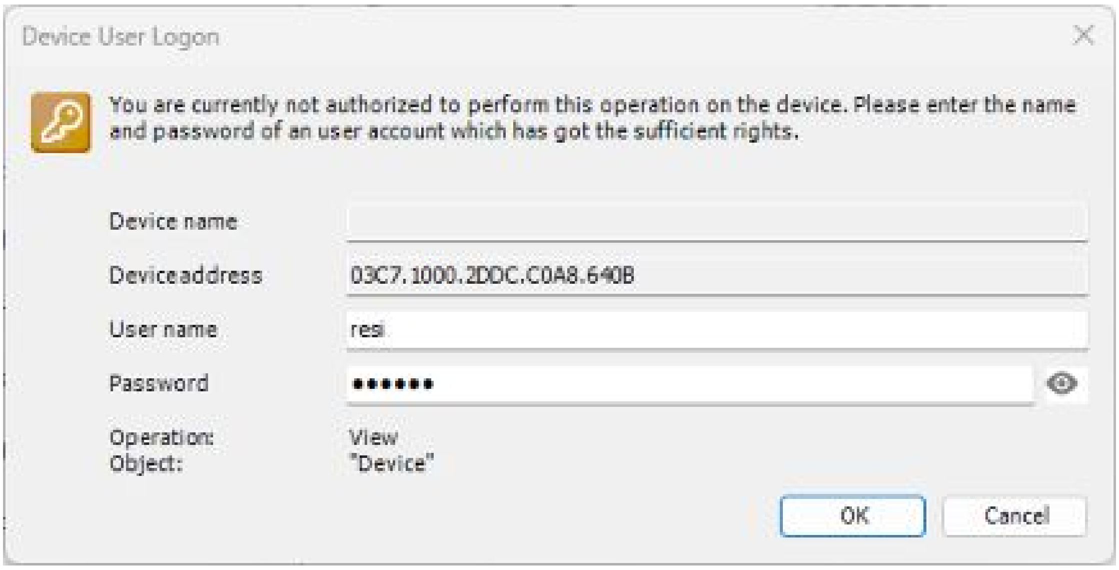
□ Keep password

☑ Password can be changed by user

□ Password must be changed at first login

Password policy: Password policy could not be retrieved from the device.

Enter now the new name and password.


Device User Logon

You are currently not authorized to perform this operation on the device. Please enter the name and password of an user account which has got the sufficient rights.

Device name:

Deviceaddress: 03C7.1000.2DDC.C0A8.640B

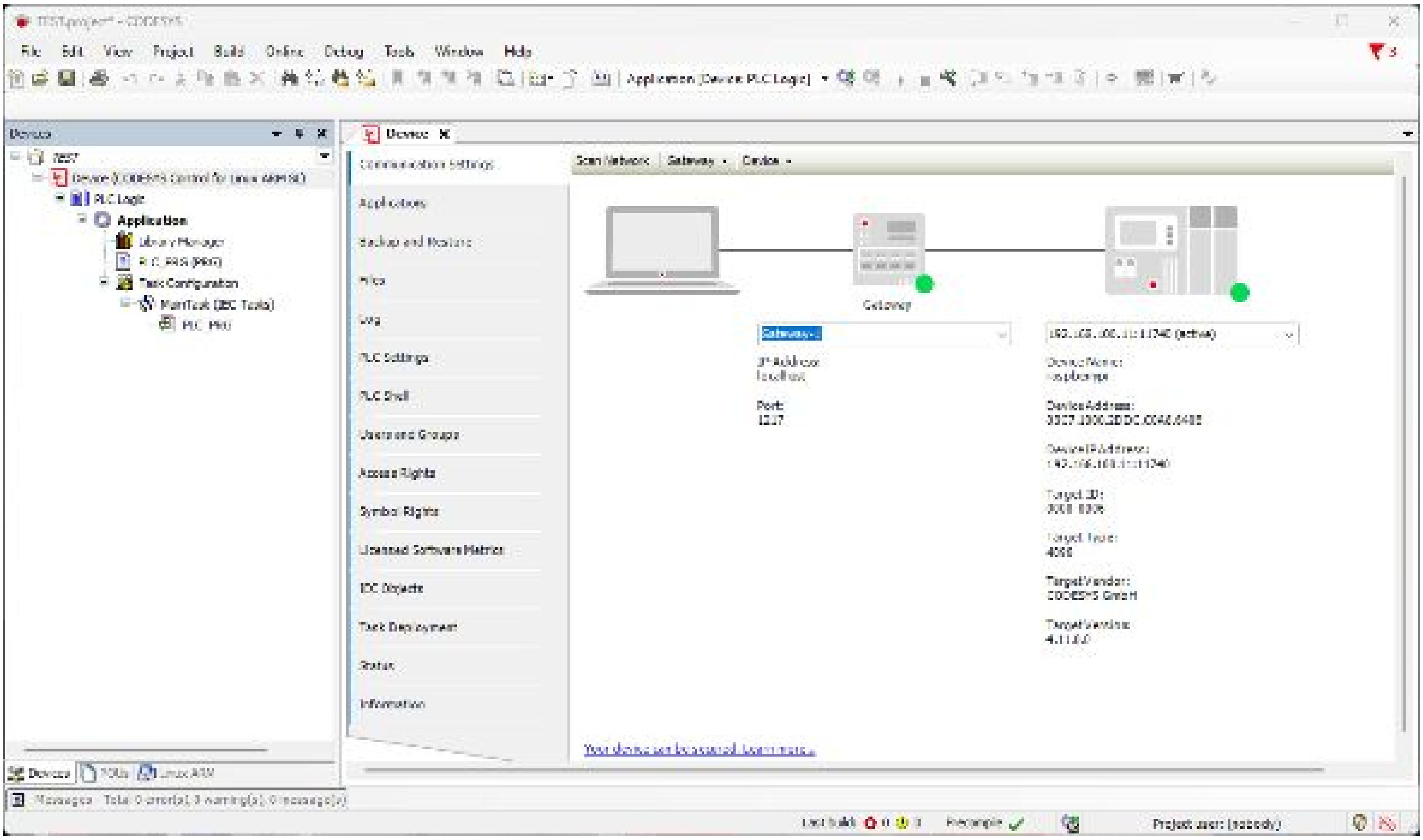
User name: resi

Password: ••••••

Operation: View
Object: "Device"

Install CODESYS® on RESI-T4/C4 controller

If everything works well, you should see two green LEDs to indicate that your device was found.



Gateway-1

IP-Address:
localhost

Port:
1217

192.168.100.11:11740 (active)

Device Name:
raspberrypi

Device Address:
0307.1000.2DDC.C0A8.6405

Device IP-Address:
192.168.100.11:11740

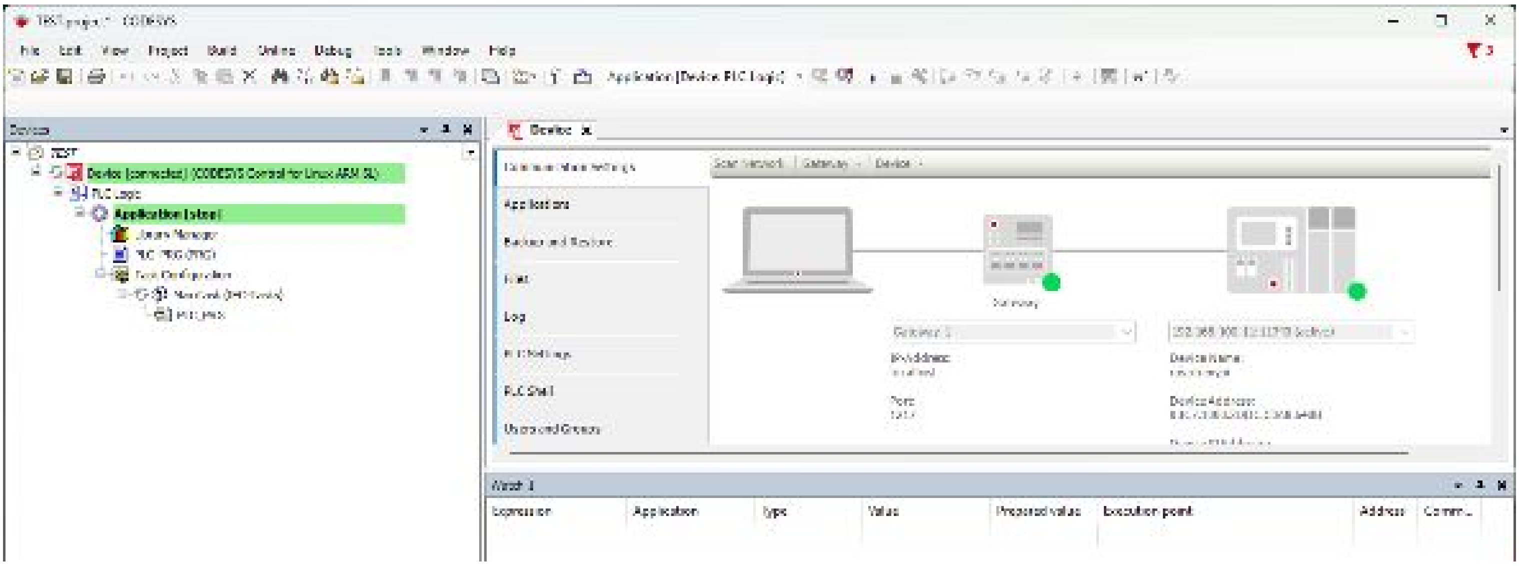
Target ID:
0000 1006

Target Type:
4096

Target Vendor:
CODESYS GmbH

Target Version:
4.11.2.0

Choose Online→ Login. Click Yes for creation of the application. Now the application is downloaded, but to start it, you have to right-click in the left tree to application [stop]. Select Start to start your empty program.

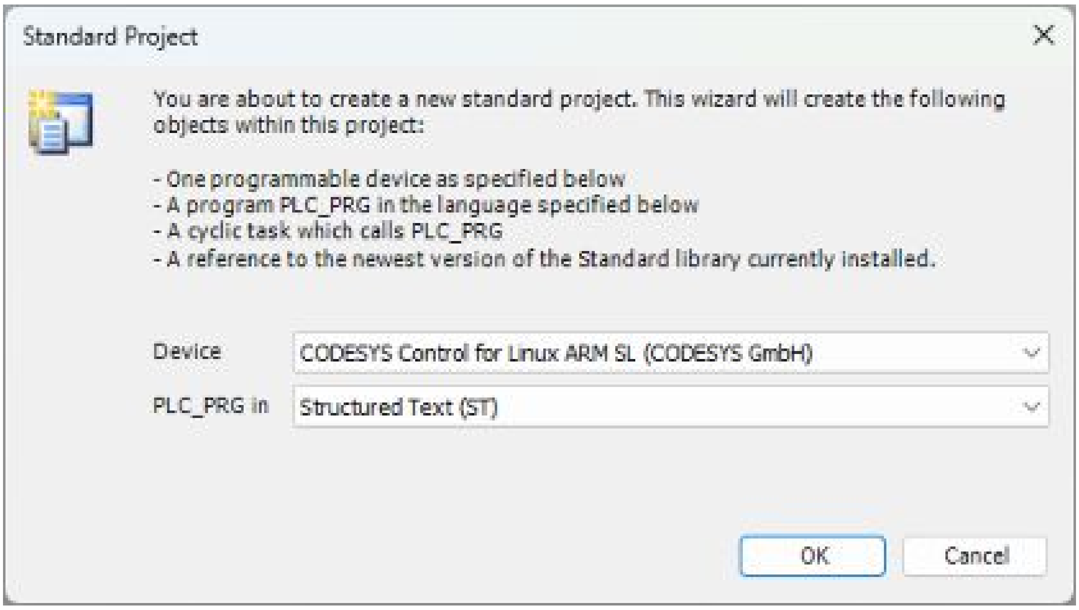


Now to write code, we have first to logout from the device. Choose Online → Logout!

CODESYS®
ASCII driver for RESI-T4 controller

Connect to IoT controller with CODESYS

Back in CODESYS we create a new standard project. Select CODESYS Control for Linux ARM SL and structured text (ST)



Standard Project

You are about to create a new standard project. This wizard will create the following objects within this project:

- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG
- A reference to the newest version of the Standard library currently installed.

Device: CODESYS Control for Linux ARM SL (CODESYS GmbH)

PLC_PRG in: Structured Text (ST)

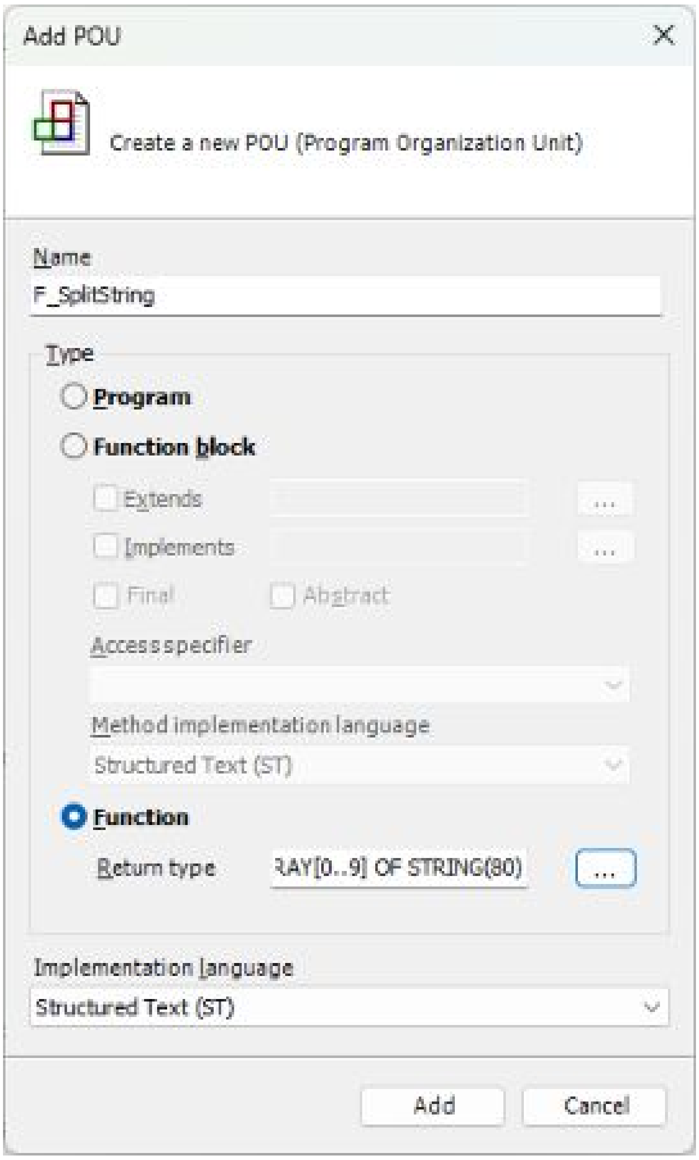
Then we add a Global variable list object named RESI and we enter the following listing into it.

Listing of Global variable list RESI:

```
{attribute 'qualified_only'}
VAR_GLOBAL
        // TRUE: ASCII Communcation to RESI Controller is ok, FALSE: No ASCII communication to RESI Controller
        RESI_IsOnline:BOOL;
        // Counter of communcation errors since start of task
        RESI_Errors:UDINT;
        // Command which produced last communcation error
        RESI_ErrorsLastCmd:STRING(80);

        // Current time of RESI Controller e.g. RESI-T4-A or RESI-C4-A-12DI12DO,...
        RESI_Type:STRING(40);
        // Current software version  of RESI Controller e.g. 1.1.0
        RESI_Version:STRING(40);

        // Current status of DIP switch 0-255, 0x00-0xFF Bit 0:DIP Switch 1, 1:DIP 2,...
        RESI_DIP_Switch:UINT;

        // LED1:GREEN
        // LED2:WHITE
        // LED3:RED
        // LED4:YELLOW
        // Current Mode for the 4 LEDs.
        // Modes:
        // OFF: LED will be OFF
        // ON: LED will be ON
        // INV: Last LED state was inverted
        // PULSE: LED is in PULSE state
        // BLINK: LED is in BLINK state
        // FLASH: LED is in FLASH state
        RESI_LEDx_Mode:ARRAY [1..4] OF STRING(20);
        // Current state of LED: TRUE: LED is ON, FALSE: LED is OFF
        RESI_LEDx_State:ARRAY [1..4] OF BOOL;
        // New Mode for LED:
        // OFF: Switch LED to OFF
        // ON: Switch LED to ON
        // BLINKVERYSLOW: LED blinks with 3s rhythm
        // BLINKSLOW: LED blinks with 1s rhythm
        // BLINKFAST: LED blinks with 0.1s rhythm
        // PULSEVERYSLOW: LED is ON for 3s and then OFF forever
        // PULSESLOW: LED is ON for 1s and then OFF forever
        // PULSEFAST: LED is ON for 0.1s and then OFF forever
        // FLASHVERYSLOW: LED is ON for 0.6s and then OFF for 5.4s, cycle will be repeated forever
        // FLASHSLOW: LED is ON for 0.3s and then OFF for 1.7s, cycle will be repeated forever
        // FLASHFAST: LED is ON for 0.03s and then OFF for 0.17s, cycle will be repeated forever
        RESI_LEDx_NewMode:ARRAY [1..4] OF STRING(20);
        // Internal used - do not overwrite
        RESI_LEDx_ActMode:ARRAY [1..4] OF STRING(20);

        // Acutual Date+Time of internal clock...
        // Format: YMD,<Year:24-99>,<Month:1-2>,<Day:1-31>,
        // HMS:<Hour:0-23>,<Minute:0.59>,<Seconds:0-59>,<DayOfWeek:MON,TUE,WED,THU,FRI,SAT,SUN>,
        // DOK,<1=Date is OK,0=Date is NOK>,TOK,<1=Time is OK,0=Time is NOK>
        // e.g. YMD,24,2,29,HMS:14:56:34,THU
        RESI_RTC_CurrentTime:STRING(40);

        // New Date+Time of internal clock...
        // Format: YMD,<Year:24-99>,<Month:1-2>,<Day:1-31>,
        // HMS:<Hour:0-23>,<Minute:0.59>,<Seconds:0-59>,<DayOfWeek:MON,TUE,WED,THU,FRI,SAT,SUN>
        // e.g. YMD,24,2,29,HMS:14:56:34,THU
        RESI_RTC_NewTime:STRING(40);
        // Internal used - do not overwrite
        RESI_RTC_ActTime:STRING(40);
END_VAR
```

Now we add a POU for a function named F_SplitString with the return type
`ARRAY[0..9] OF STRING(80)`

Enter the Listing into this function



Add POU

Create a new POU (Program Organization Unit)

Name
F_SplitString

Type

○ Program

○ Function block

□ Extends

□ Implements

□ Final □ Abstract

Access specifier

Method implementation language
Structured Text (ST)

● Function

Return type RAY[0..9] OF STRING(80)

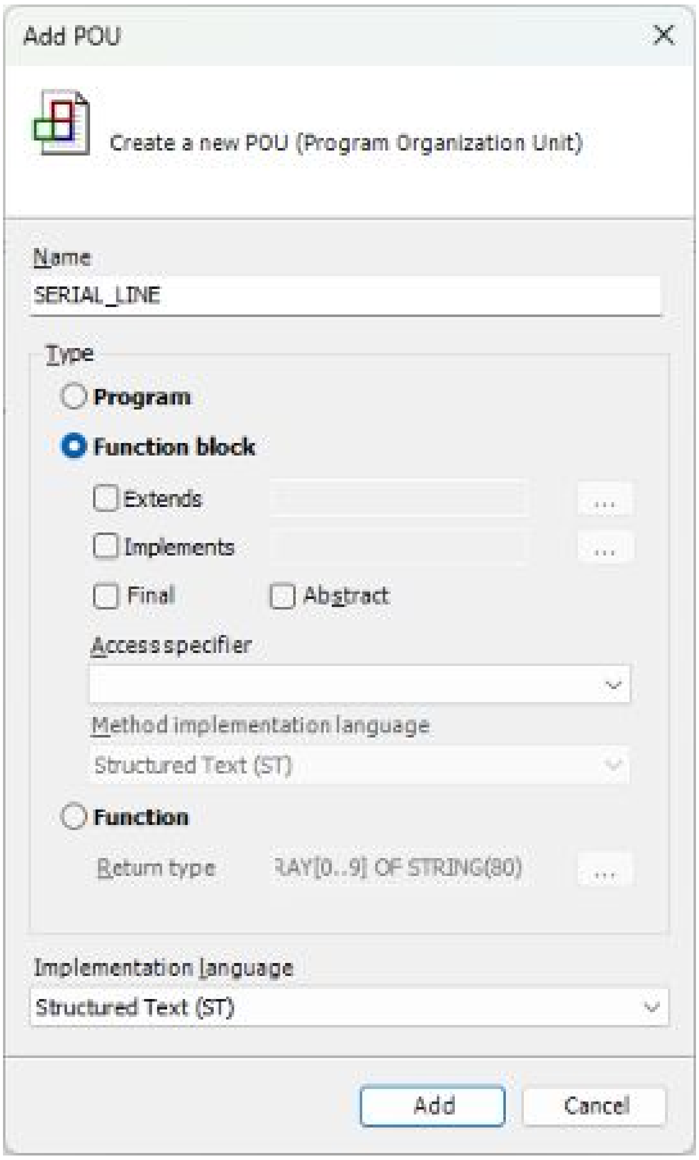
Implementation language
Structured Text (ST)

Listing of function F_SplitString:

```
FUNCTION F_SplitString : ARRAY[0..9] OF STRING(80)
VAR_INPUT
   sInput : STRING(80);
   sSplitChar : STRING(1);
END_VAR
VAR_OUTPUT
   iParts : INT;
END_VAR
VAR
   sInputCopy : STRING(80);
   sSplitValue : STRING(80);
   iSplitLength : INT := 0;
   i : INT;
END_VAR

sInputCopy := sInput;
iParts:=0;
FOR i := 0 TO 9 DO
   IF FIND(sInputCopy, sSplitChar) > 0 THEN
     sSplitValue := LEFT(sInputCopy, FIND(sInputCopy, sSplitChar) - 1);
     iSplitLength := LEN(sSplitValue) + 1;
          iParts:=iParts+1;
   ELSE
     sSplitValue := sInputCopy;
     iSplitLength := LEN(sSplitValue);
          iParts:=iParts+1;
   END_IF

   sInputCopy := DELETE(sInputCopy, iSplitLength, 1);

   F_SplitString[i] := sSplitValue;

   IF LEN(sInputCopy) = 0 THEN
     EXIT;
   END_IF
END_FOR
```

Now we add a POU for a function block named SERIAL_LINE

Enter the Listing into this function



Add POU

Create a new POU (Program Organization Unit)

Name
SERIAL_LINE

Type

○ Program

● Function block

□ Extends ...

□ Implements ...

□ Final □ Abstract

Access specifier

Method implementation language
Structured Text (ST)

○ Function

Return type RAY[0..9] OF STRING(80) ...

Implementation language
Structured Text (ST)

Listing of function block SERIAL_LINE:

```
FUNCTION_BLOCK SERIAL_LINE
VAR_INPUT
        udiPort: UDINT;
        udiBaudrate: UDINT;
        sbStopBits: COM.STOPBIT:= COM.STOPBIT.ONESTOPBIT;
        paParity: COM.PARITY:= COM.PARITY.EVEN;
        iCmdIndex: INT;
        sWriteCmd: STRING(80);
        sWriteFullCmd: STRING(80);
        udiByteSize: UDINT;
END_VAR
VAR_OUTPUT
        sReadPart: STRING(80);
        sReadTmp: STRING(80);
        sReadCmd: STRING(80);
        errError: COM.ERROR;
        xClosed: BOOL:= TRUE;
END_VAR
VAR
        iState: INT;
        tTimer: TON;
        comOpen: COM.Open; (* Instance of the function block for opening a port *)
        hCom: CAA.HANDLE; (* handle of the port*)
        aParameter: ARRAY [1..7] OF COM.PARAMETER;
        comWrite: COM.Write;      (* Instance of the Write function  block *)
        bWriteBuffer: ARRAY [1..80] OF BYTE; (*Used to write data to the serial port*)
        szWrite: CAA.SIZE;
        comRead: COM.Read; (* Instance of the Read function block *)
        bReadBuffer: ARRAY [1..80] OF BYTE; (*Used to read data from the serial port*)
        szRead: CAA.SIZE;
        comClose: COM.Close; (* Instance of the function block for closing a port *)
        index: UDINT;
        TmpStr: STRING(80);
        SplitParts:INT;
        Parts: ARRAY [0..9] OF STRING(80);
END_VAR

CASE iState OF

        0:      // The parameter for the COM Ports are set
                aParameter[1].udiParameterId:= COM.CAA_Parameter_Constants.udiPort;
                aParameter[1].udiValue:= udiPort;
                aParameter[2].udiParameterId:= COM.CAA_Parameter_Constants.udiBaudrate;
                aParameter[2].udiValue:= udiBaudrate;
                aParameter[3].udiParameterId:= COM.CAA_Parameter_Constants.udiStopBits;
                aParameter[3].udiValue:= INT_TO_UDINT(sbStopBits);
                aParameter[4].udiParameterId:= COM.CAA_Parameter_Constants.udiParity;
                aParameter[4].udiValue:= INT_TO_UDINT(paParity);
                aParameter[5].udiParameterId:= COM.CAA_Parameter_Constants.udiTimeout;
                aParameter[5].udiValue:= 0;
                aParameter[6].udiParameterId:= COM.CAA_Parameter_Constants.udiBinary;
                aParameter[6].udiValue:= 0;
                aParameter[7].udiParameterId:= COM.CAA_Parameter_Constants.udiByteSize;
                aParameter[7].udiValue:= udiByteSize;
                comOpen(xExecute:= FALSE);
                RESI.RESI_IsOnline:=FALSE;
                RESI.RESI_Errors:=0;
                RESI.RESI_ErrorsLastCmd:='';

                iCmdIndex:=-1;
                iState:= 1;

        1:      //First the COM Port is opened
                comOpen(xExecute:= TRUE, usiListLength:= SIZEOF(aParameter) / SIZEOF(COM.PARAMETER), pParameterList:= ADR(aParameter),
hCom=> hCom);
                xClosed:= FALSE;
                IF comOpen.xDone THEN
                        comOpen(xExecute:= FALSE);
                        //depending on the mode the state for reading or writing is set
                        iState:= 100;
                ELSIF comOpen.xError THEN
                        errError:= comOpen.eError;
                        comOpen(xExecute:= FALSE);
                        iState:= 32767;
                END_IF
```

Listing of function block SERIAL_LINE:

```
100: // ASCII Commands for RESI-T4...
        iState:=1000;
    iCmdIndex:=iCmdIndex+1;
        IF iCmdIndex>9 THEN
                iCmdIndex:=0;
        END_IF
        CASE iCmdIndex OF
                0: sWriteCmd:='VERSION';
                1: sWriteCmd:='TYPE';
                2: sWriteCmd:='GDIP';
                3: sWriteCmd:='GLED1';
                4: sWriteCmd:='GLED2';
                5: sWriteCmd:='GLED3';
                6: sWriteCmd:='GLED4';
                7: sWriteCmd:='GRTC';
                8:
                        sWriteCmd:='';
                        IF RESI.RESI_LEDx_NewMode[1]<>RESI.RESI_LEDx_ActMode[1] THEN
                                RESI.RESI_LEDx_ActMode[1]:=RESI.RESI_LEDx_NewMode[1];
                                TmpStr:=RESI.RESI_LEDx_NewMode[1];
                                sWriteCmd:='SL1';
                        ELSIF RESI.RESI_LEDx_NewMode[2]<>RESI.RESI_LEDx_ActMode[2] THEN
                                RESI.RESI_LEDx_ActMode[2]:=RESI.RESI_LEDx_NewMode[2];
                                TmpStr:=RESI.RESI_LEDx_NewMode[2];
                                sWriteCmd:='SL2';
                        ELSIF RESI.RESI_LEDx_NewMode[3]<>RESI.RESI_LEDx_ActMode[3] THEN
                                RESI.RESI_LEDx_ActMode[3]:=RESI.RESI_LEDx_NewMode[3];
                                TmpStr:=RESI.RESI_LEDx_NewMode[3];
                                sWriteCmd:='SL3';
                        ELSIF RESI.RESI_LEDx_NewMode[4]<>RESI.RESI_LEDx_ActMode[4] THEN
                                RESI.RESI_LEDx_ActMode[4]:=RESI.RESI_LEDx_NewMode[4];
                                TmpStr:=RESI.RESI_LEDx_NewMode[4];
                                sWriteCmd:='SL4';
                        END_IF
                        IF sWriteCmd<>'' THEN
                                IF TmpStr='ON' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'ON');
                                ELSIF TmpStr='OFF' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'OFF');
                                ELSIF TmpStr='BLINKVERYSLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'BLINK:3000');
                                ELSIF TmpStr='BLINKSLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'BLINK:1000');
                                ELSIF TmpStr='BLINKFAST' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'BLINK:100');
                                ELSIF TmpStr='PULSEVERYSLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'PULSE:3000');
                                ELSIF TmpStr='PULSESLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'PULSE:1000');
                                ELSIF TmpStr='PULSEFAST' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'PULSE:100');
                                ELSIF TmpStr='FLASHVERYLSLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'FLASH:600,5400');
                                ELSIF TmpStr='FLASHSLOW' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'FLASH:300,1700');
                                ELSIF TmpStr='FLASHFAST' THEN
                                        sWriteCmd:=CONCAT(sWriteCmd,'FLASH:30,170');
                                ELSE
                                        iState:=100;
                                END_IF
                        ELSE
                                iState:=100;
                        END_IF
                9:
                        IF RESI.RESI_RTC_NewTime<>RESI.RESI_RTC_ActTime THEN
                          RESI.RESI_RTC_ActTime:=RESI.RESI_RTC_NewTime;
                          sWriteCmd:=CONCAT('SRTC:',RESI.RESI_RTC_NewTime);
                        ELSE
                                iState:=100;
                        END_IF
        END_CASE
```

Listing of function block SERIAL_LINE:

```
        1000: // the writing process is started
                IF NOT comWrite.xExecute THEN
                        sWriteFullCmd:=CONCAT('#255,',sWriteCmd);
                        sWriteFullCmd:=CONCAT(sWriteFullCmd,'$R');
                        szWrite:= INT_TO_UDINT(LEN(sWriteFullCmd));
                        MEM.MemMove(ADR(sWriteFullCmd), ADR(bWriteBuffer), ANY_TO_UINT(szWrite));
                END_IF
                sReadTmp:='';
                sReadCmd:='';
                comWrite(xExecute:= TRUE, hCom:= hCom, pBuffer:= ADR(bWriteBuffer), szSize:= szWrite);
                IF comWrite.xDone THEN
                        // the flag is set to false, that in the next cyle this process is started again, by setting it to true
                        comWrite(xExecute:= FALSE);
                        iState:= 1010;
                ELSIF comWrite.xError THEN
                        errError:= comWrite.eError;
                        comWrite(xExecute:= FALSE);
                        iState:= 32767;
                END_IF

        1010: //start timer for timeout for read answer
                tTimer(IN:= FALSE);
                tTimer();
                tTimer(IN:= TRUE, PT:= T#1000MS);
                tTimer();
                iState:= 1020;

        1020: //The reading process is started
                // Timeout with answer ...
                tTimer();
                IF tTimer.Q OR tTimer.ET>=T#1000MS THEN
                        tTimer(IN:= FALSE);
                        comRead(xExecute:= FALSE);
                        RESI.RESI_IsOnline:=FALSE;
                        RESI.RESI_Errors:=RESI.RESI_Errors+1;
                        RESI.RESI_ErrorsLastCmd:=sWriteCmd;
                        iState:= 100;
                ELSE
                        comRead(xExecute:= TRUE, hCom:= hCom, pBuffer:= ADR(bReadBuffer), szBuffer:= SIZEOF(bReadBuffer));
                        IF comRead.xDone THEN
                                szRead:= comRead.szSize;
                                //check if the Port has send something
                                IF szRead > 0 THEN
                                        //the text from the read buffer is saved in the sReadText variable
                                        MEM.MemMove(ADR(bReadBuffer), ADR(sReadPart), ANY_TO_UINT(szRead));
                                        MEM.MemFill(ADR(sReadPart) + ANY_TO_UINT(szRead), 1, 0);
                                        sReadTmp:=CONCAT(sReadTmp,sReadPart);
                                END_IF
                                // the flag has to be set to false, that in the next cylce, the read task will start again, by
setting it to true
                                comRead(xExecute:= FALSE);
                                IF LEFT(sReadTmp,5)='#255,' AND RIGHT(sReadTmp,1)='$R' THEN
                                        // Antwort ist vollständig...
                                        sReadCmd:=MID(SReadTmp,LEN(sReadTmp)-6,6);
                                        tTimer(IN:= FALSE);
                                        iState:=2000;
                                END_IF
                        ELSIF comRead.xError THEN
                                tTimer(IN:= FALSE);
                                errError:= comRead.eError;
                                comRead(xExecute:= FALSE);
                                iState:= 32767;
                        END_IF
                END_IF
```

Listing of function block SERIAL_LINE:

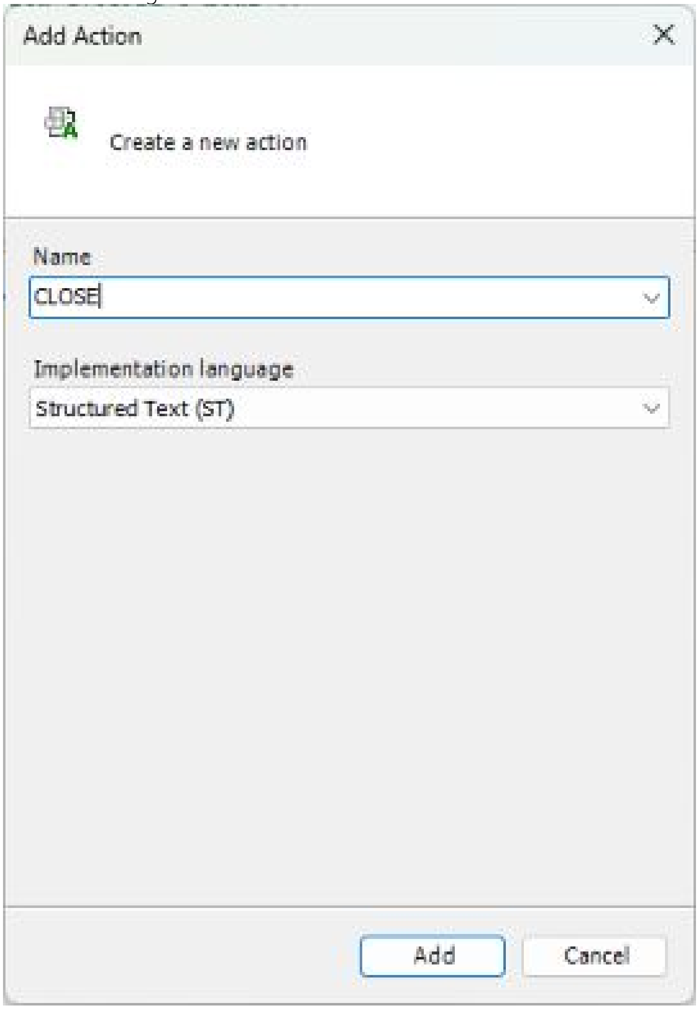
```
        2000: // Answer is here..
                RESI.RESI_IsOnline:=TRUE;
                CASE iCmdIndex OF
                        0:
                                IF LEFT(sReadCmd,8)='VERSION:' THEN
                                        RESI.RESI_Version:=MID(sReadCmd,LEN(sReadCmd)-8,9);
                                END_IF
                        1:
                                IF LEFT(sReadCmd,5)='TYPE:' THEN
                                        RESI.RESI_Type:=MID(sReadCmd,LEN(sReadCmd)-5,6);
                                END_IF
                        2:
                                IF LEFT(sReadCmd,5)='GDIP:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-5,6);
                                        Parts := F_SplitString(sInput := TmpStr, sSplitChar := ',', iParts => SplitParts);
                                        IF SplitParts=2 THEN
                                                RESI.RESI_DIP_Switch:=STRING_TO_WORD(Parts[0]);
                                        END_IF
                                END_IF
                        3:
                                IF LEFT(sReadCmd,6)='GLED1:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-6,7);
                                        Parts := F_SplitString(sInput := TmpStr, sSplitChar := ',', iParts => SplitParts);
                                        IF SplitParts=3 THEN
                                                RESI.RESI_LEDx_Mode[1]:=Parts[0];
                                                RESI.RESI_LEDx_State[1]:=TO_BOOL(STRING_TO_WORD(Parts[1]));
                                        END_IF
                                END_IF
                        4:
                                IF LEFT(sReadCmd,6)='GLED2:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-6,7);
                                        Parts := F_SplitString(sInput := TmpStr, sSplitChar := ',', iParts => SplitParts);
                                        IF SplitParts=3 THEN
                                                RESI.RESI_LEDx_Mode[2]:=Parts[0];
                                                RESI.RESI_LEDx_State[2]:=TO_BOOL(STRING_TO_WORD(Parts[1]));
                                        END_IF
                                END_IF
                        5:
                                IF LEFT(sReadCmd,6)='GLED3:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-6,7);
                                        Parts := F_SplitString(sInput := TmpStr, sSplitChar := ',', iParts => SplitParts);
                                        IF SplitParts=3 THEN
                                                RESI.RESI_LEDx_Mode[3]:=Parts[0];
                                                RESI.RESI_LEDx_State[3]:=TO_BOOL(STRING_TO_WORD(Parts[1]));
                                        END_IF
                                END_IF
                        6:
                                IF LEFT(sReadCmd,6)='GLED4:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-6,7);
                                        Parts := F_SplitString(sInput := TmpStr, sSplitChar := ',', iParts => SplitParts);
                                        IF SplitParts=3 THEN
                                                RESI.RESI_LEDx_Mode[4]:=Parts[0];
                                                RESI.RESI_LEDx_State[4]:=TO_BOOL(STRING_TO_WORD(Parts[1]));
                                        END_IF
                                END_IF
                        7:
                                IF LEFT(sReadCmd,5)='GRTC:' THEN
                                        TmpStr:=MID(sReadCmd,LEN(sReadCmd)-5,6);
                                        RESI.RESI_RTC_CurrentTime:=TmpStr;
                                END_IF

                                ;
                END_CASE
                iState:=100;
```

Listing of function block SERIAL_LINE:

```
        9999: //Closing the ports
                IF hCom <> 0 THEN
                        comClose(xExecute:= TRUE, hCom:= hCom);
                        IF comClose.xDone THEN
                                comClose(xExecute:= FALSE);
                                Fb_Init(FALSE, FALSE);
                        ELSIF comClose.xError THEN
                                errError:= comClose.eError;
                                comClose(xExecute:= FALSE);
                                iState:= 32767;
                        END_IF
                ELSE
                        Fb_Init(FALSE, FALSE);
                END_IF

        32767:
                RESI.RESI_IsOnline:=FALSE;
                ;

END_CASE
```

Now we add an action to the function block named SERIAL_LINE named CLOSE
Right-Click on the FB SERIAL_LINE in the Device tree and select Add Object → Action

Enter the Listing into this function


Add Action

Create a new action

Name: CLOSE

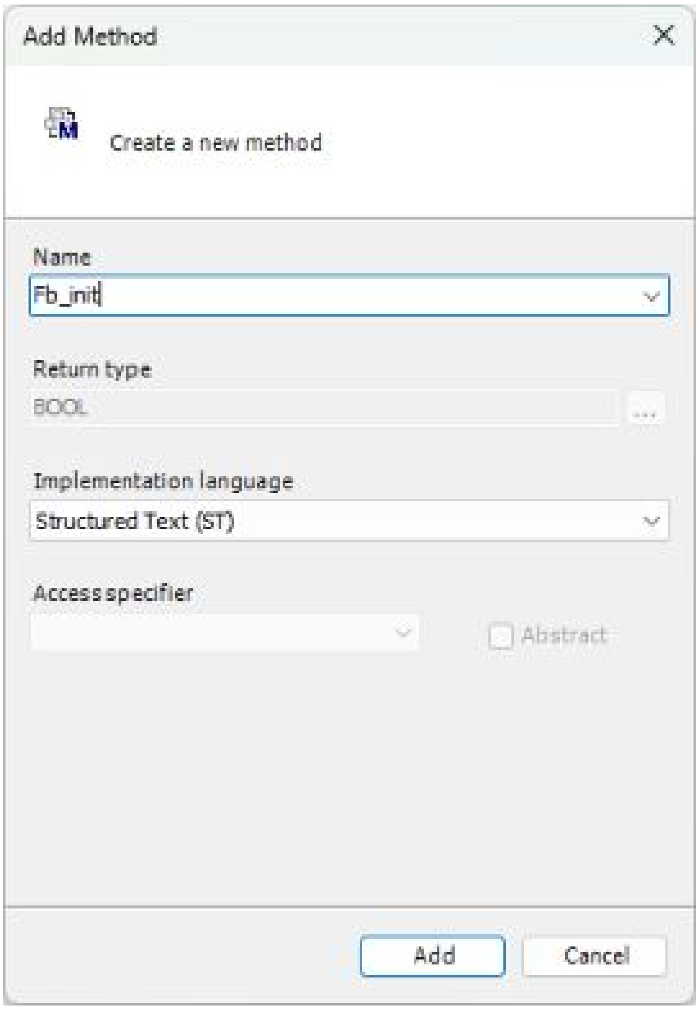
Implementation language: Structured Text (ST)

Listing of function block action SERIAL_LINE.CLOSE:

```
iState:= 9999;
THIS^();
```

Now we add a method to the function block named SERIAL_LINE named FB_init
Right-Click on the FB SERIAL_LINE in the Device tree and select Add Object → Method

Enter the Listing into this function



Name: Fb_init

Return type: BOOL

Implementation language: Structured Text (ST)

Access specifier:

□ Abstract

Listing of function block method SERIAL_LINE.FB_init:

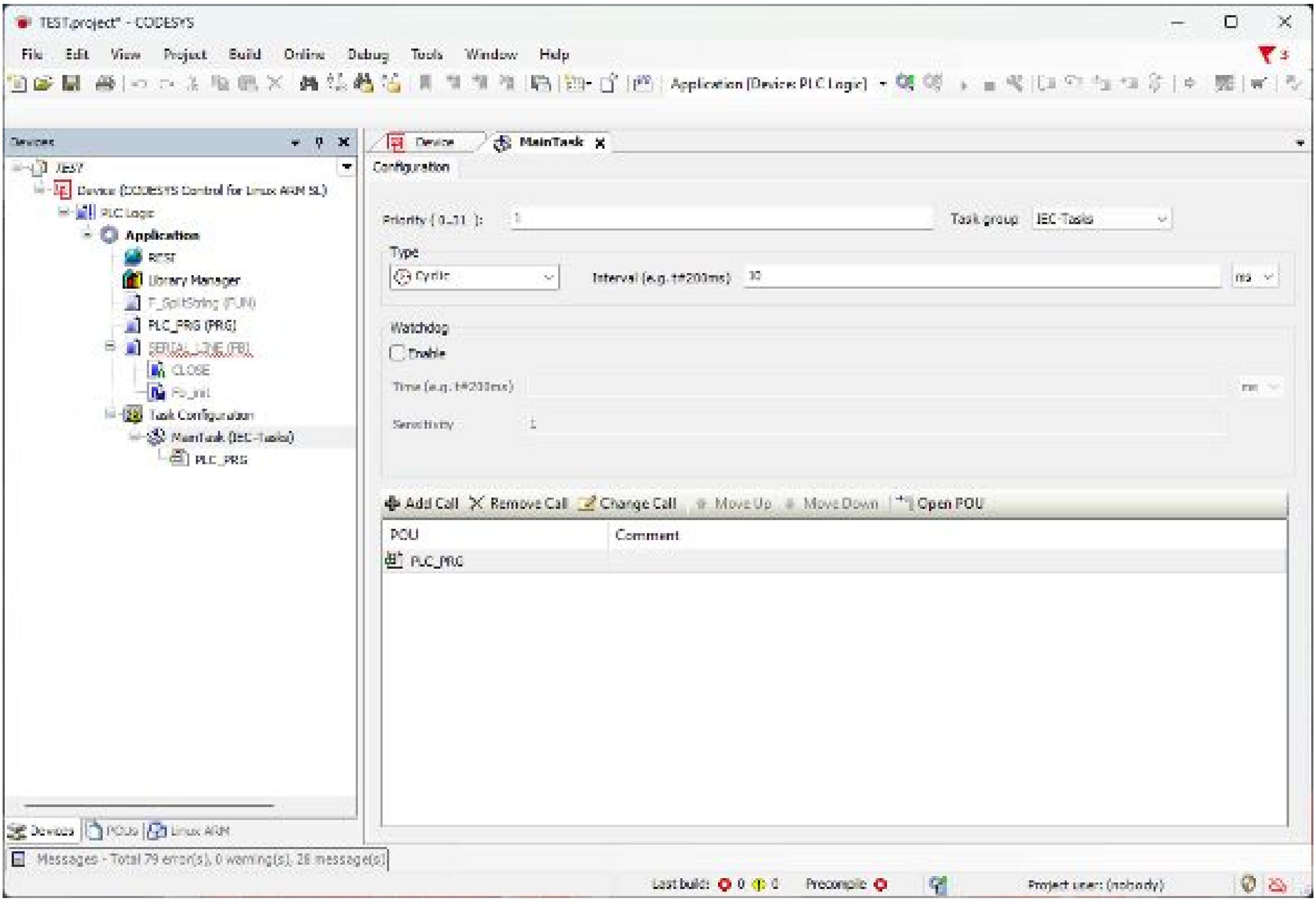
```
METHOD Fb_init : BOOL
VAR_INPUT
        bInitRetains : BOOL;
        bInCopyCode : BOOL;
END_VAR
```

Last we have to write the main program. Open the PROGAM block PLC_PRG and enter the following listing

Listing of PLC_PRG:

```
PROGRAM PLC_PRG
VAR
        RESI_ASCII: SERIAL_LINE;
END_VAR

//set the special parameters for the port
RESI_ASCII(
        udiPort:= 1, //Port number
        udiBaudrate:= 9600, //bandwidth
        paParity:= COM.PARITY.NONE, //the parity is optional
        sbStopBits:= COM.STOPBIT.ONESTOPBIT,
        udiByteSize := 8); //the stopbits are optional );
```

Now we have programmed everything and our device tree should look like this. Change the settings of the MainTask to 10ms cyclic.
And everything is almost ready for testing...



```
TEST
  Device (CODESYS Control for Linux ARM SL)
    PLC Logic
      Application
        RESI
        Library Manager
        F_SplitString (FUN)
        PLC_PRG (PRG)
        SERIAL_LINE (FB)
          CLOSE
          FB_init
        Task Configuration
          MainTask (IEC-Tasks)
            PLC_PRG
```

Priority (0..31): 1

Task group: IEC-Tasks

Type: Cyclic

Interval (e.g. t#200ms): 10 ms

Watchdog

☐ Enable

Time (e.g. t#200ms): ms

Sensitivity: 1

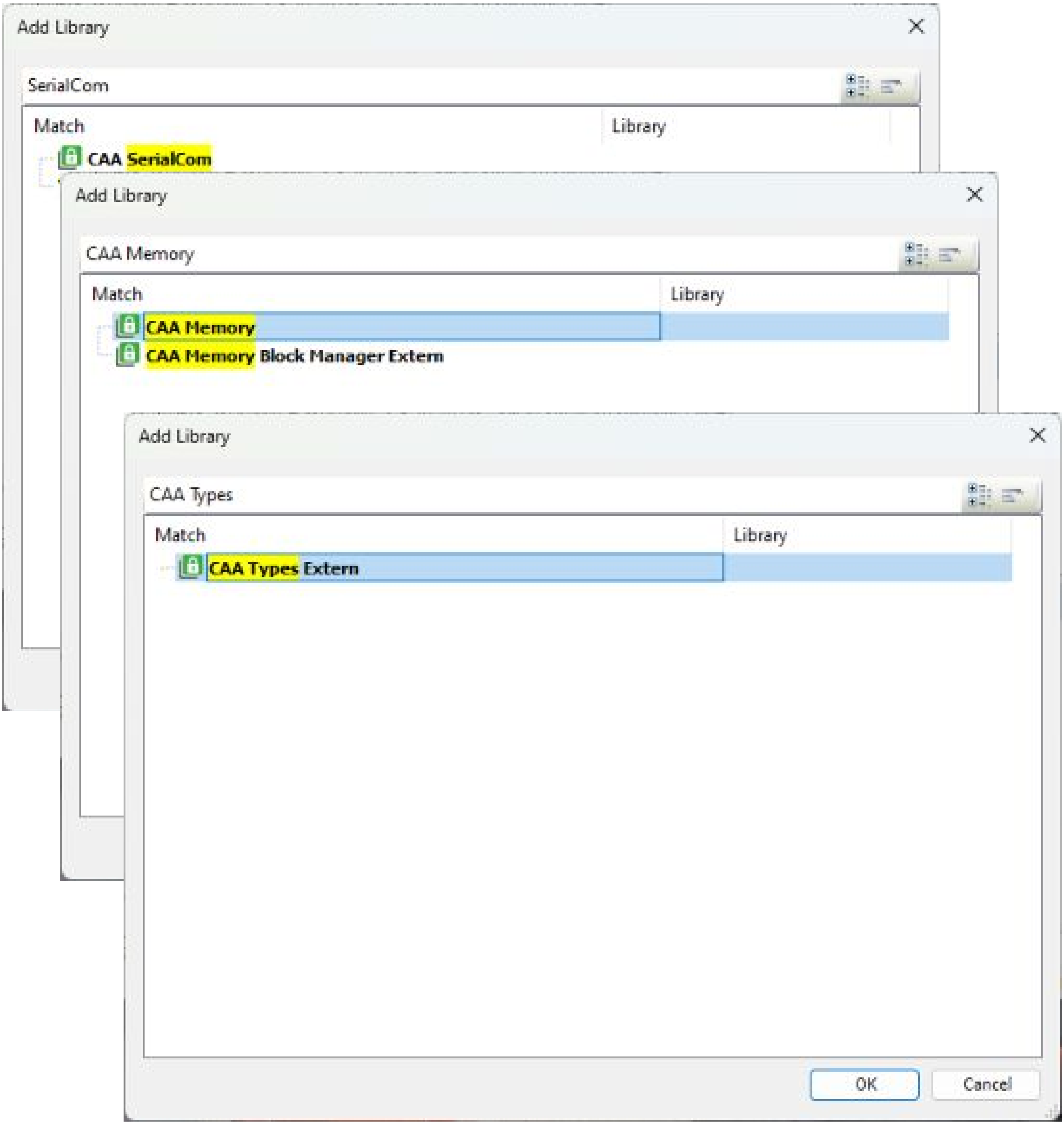
POU	Comment
PLC_PRG	

Messages - Total 79 error(s), 0 warning(s), 28 message(s)

Last build: 0 0 Precompile Project user: (nobody)

CODESYS® ASCII driver for RESI-T4 controller

The last issue is to install the right libraries. Open the library manager and select the library CAA SerialCom. Install it. Then install CAA Memory. Then install CAA Types Extern. Finished.



Add Library

SerialCom

Match	Library
CAA SerialCom	

Add Library

CAA Memory

Match	Library
CAA Memory	
CAA Memory Block Manager Extern	

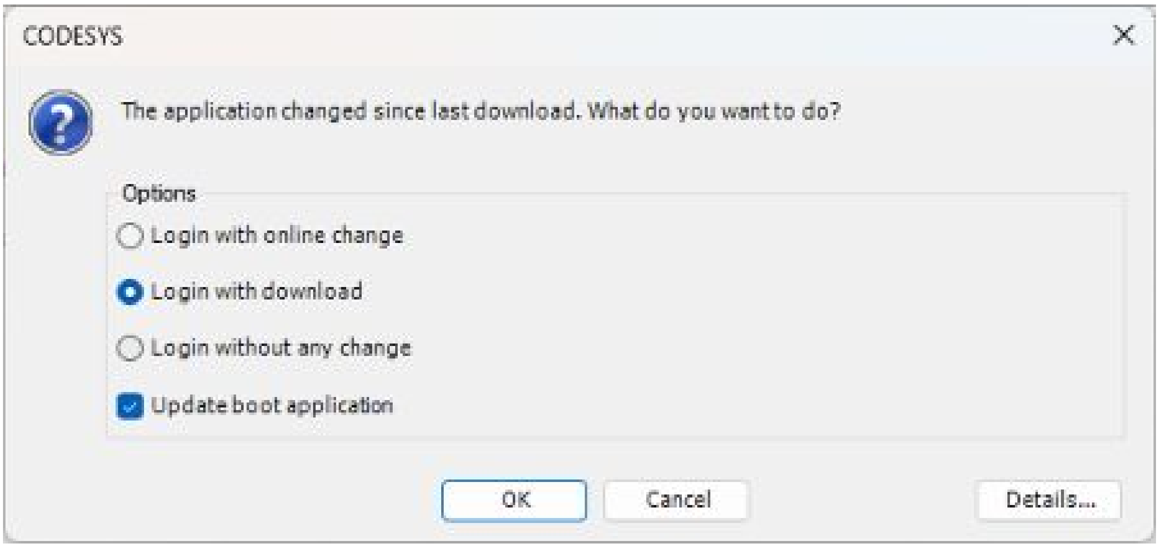
Add Library

CAA Types

Match	Library
CAA Types Extern	

CODESYS® ASCII driver for RESI-T4 controller

Click on Online → Login. Select Login with download.
After successful download click on the start button. Now the application runs in our controller.



CODESYS

The application changed since last download. What do you want to do?

Options

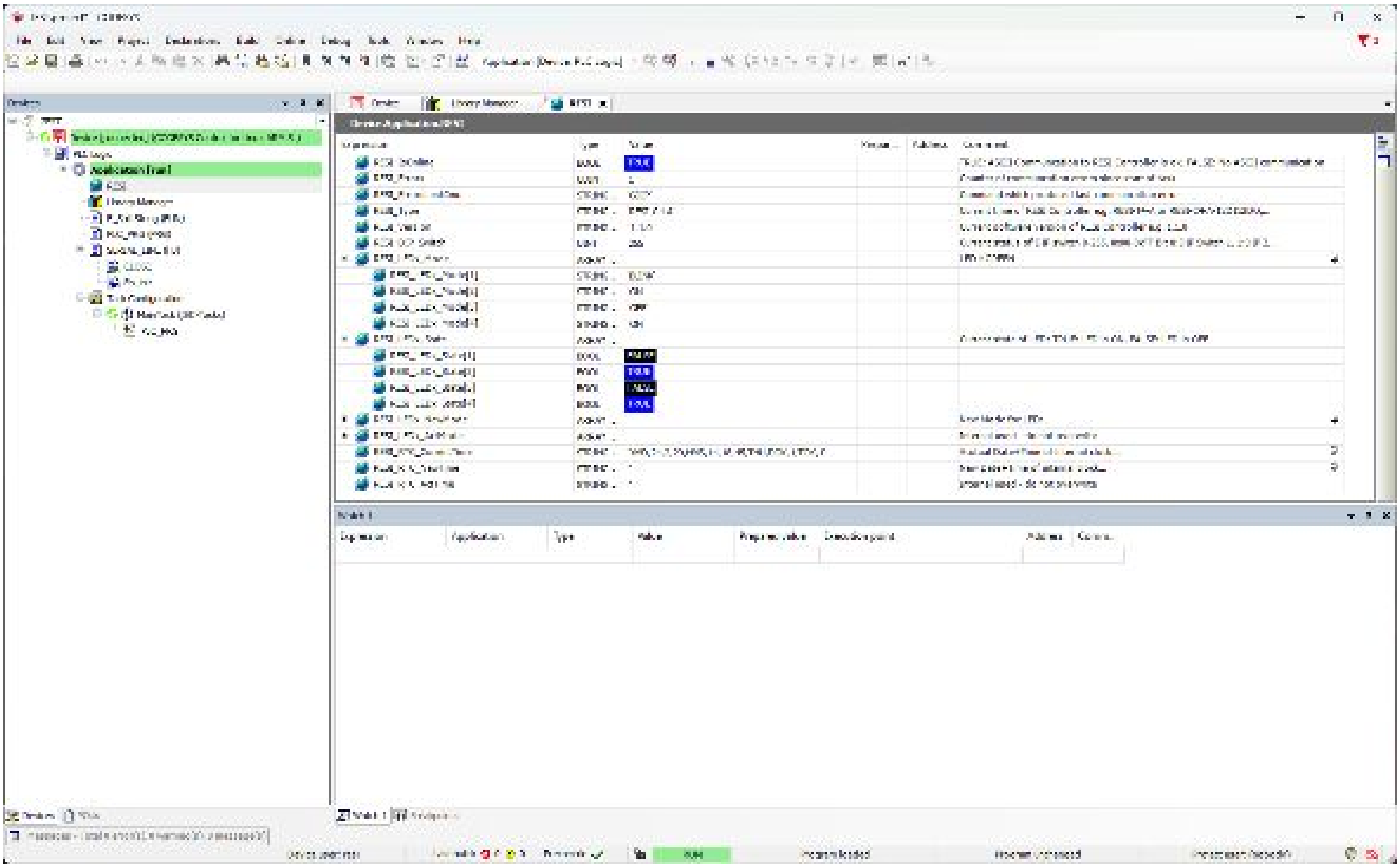
○ Login with online change

● Login with download

○ Login without any change

☑ Update boot application

If everything runs fine, your output should look like this, if you double click on the global variable list RESI



CODESYS® ASCII driver for RESI-T4 controller

Using the ASCII driver

RESI_IsOnline must be TRUE. This indicates, that the CODESYS communicates with the RESI IoT Controller.

If a communication error arises the variable RESI_Errors is incremented. This can happen sometimes. This is no big deal.
RESI_ErrorsLastCmd will show the last command that causes this error.

RESI_Type and RESI_Version are strings which show the current type and software version of our controller.

RESI_DIP_Switch gives back the current setting of the 8-pin DIP switch in the cover. Each bit stands for a different DIP Switch (Bit 0=DIP 1, 1=DIP2,..,7=DIP 8)
I
RESI_LEDx_Mode is a array with 4 elements showing the current mode of the LEDs:
OFF: LED is OFF
ON: LED is ON
INV: LED was inverted
PULSE: LED does one time pulse
BLINK: LED blinks cyclic symmetrically
FLASH: LED blinks cyclic asymmetrically
Index 1 is the GREEN, 2=WHITE, 3=RED and 4 is the YELLOW LED

To set a new mode write to RESI_LEDx_NewMode a STRING from the following list:
OFF: Switch LED to OFF
ON: Switch LED to ON
BLINKVERYSLOW: LED blinks with 3s rhythm
BLINKSLOW: LED blinks with 1s rhythm
BLINKFAST: LED blinks with 0.1s rhythm
PULSEVERYSLOW: LED is ON for 3s and then OFF forever
PULSESLOW: LED is ON for 1s and then OFF forever
PULSEFAST: LED is ON for 0.1s and then OFF forever
FLASHVERYSLOW: LED is ON for 0.6s and then OFF for 5.4s, cycle will be repeated forever
FLASHSLOW: LED is ON for 0.3s and then OFF for 1.7s, cycle will be repeated forever
FLASHFAST: LED is ON for 0.03s and then OFF for 0.17s, cycle will be repeated forever

RESI_RTC_CurrentTime return the current date and time and weekday of the integrated RTC with accu buffering.
The string format is:
Format: YMD,<Year:24-99>,<Month:1-2>,<Day:1-31>,HMS:<Hour:0-23>,<Minute:0.59>,<Seconds:0-59>,
<DayOfWeek:MON,TUE,WED,THU,FRI,SAT,SUN>,
DOK,<1=Date is OK,0=Date is NOK>,TOK,<1=Time is OK,0=Time is NOK>
e.g. YMD,24,2,29,HMS:14:56:34,THU

You can set this RTC writing a string to RESI_RTC_NewTime.
Format: YMD,<Year:24-99>,<Month:1-2>,<Day:1-31>,HMS:<Hour:0-23>,<Minute:0.59>,<Seconds:0-59>,
<DayOfWeek:MON,TUE,WED,THU,FRI,SAT,SUN>
e.g. YMD,24,2,29,HMS:14:56:34,THU

You can download this demo software from our homepage www.RESI.cc

RESI Informatik & Automation GmbH
Altenmarkt 29, A-8551 Wies, AUSTRIA
help@RESI.cc www.RESI.cc