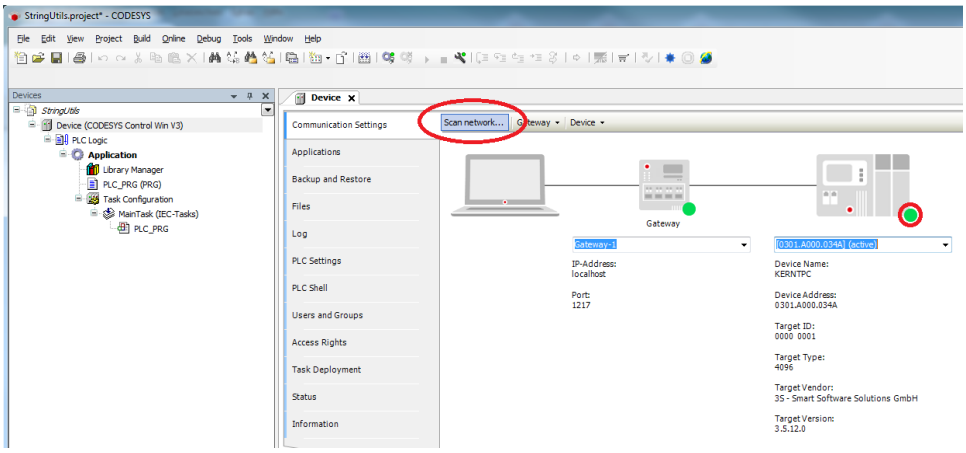
Working with Strings More Than 255 Characters

All functions are also included in the standard library. However, these are restricted to strings less than 255 characters due to historical limitations.

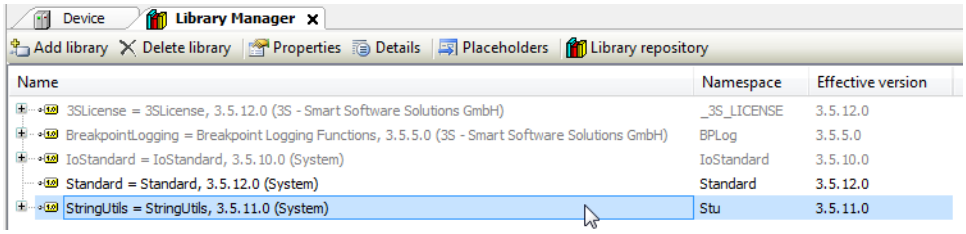
Twenty years ago, it was simply inconceivable to work with longer strings on a control system.

In order to maintain backwards compatibility, a separate *StringUtilities* library had to be created for these applications.

- Create a "Standard project" and select *CODESYS Control Win V3* as the device.
- Define the target system by means of the *Network scan*.



- Open the *Library Manager* and add the *StringUtils* library.



- Adapt the POU *PLC_PRG* as follows:

Declaration

```
VAR
    sTo             :    STRING(300) := 'Hello ';
    psTo            :    POINTER TO BYTE := ADR(sTo);
    sFrom           :    STRING(7) := 'World';
    psFrom          :    POINTER TO BYTE := ADR(sFrom);
    xResult         :    BOOL;
    xConcat         :    BOOL;
    diLen           :    DINT;
    diCopied        :    DINT;
    sCopied         :    STRING(300);
    iFind           :    INT;
    sMid            :    STRING;
    uiMid           :    UINT;


    sUpperText      :    STRING := 'This Text will be converted';
    pstUpper        :    POINTER TO BYTE := ADR(sUpperText);
    xUpper          :    BOOL;
    xLower          :    BOOL;

    sCmp1           :    STRING := 'Hello';
    sCmp2           :    STRING := 'HElo';
    xCaseCmp        :    BOOL;
    xCaseCmpEnd     :    BOOL;
    xCaseCmpStart   :    BOOL;
    iResult         :    INT;
    xCmpEnd         :    BOOL;
    xCmpStart       :    BOOL;

    sDelete         :    STRING := '2 characters will delete';
    xDelete         :    BOOL;

    sTrim           :    STRING;
    xTrim           :    BOOL;
    xTrimEnd        :    BOOL;
    xTrimStart      :    BOOL;

    sOldReplace     :    STRING(50) := 'Hello World';
    sReplace        :    STRING := ', good morning CODESYS-';
    xReplace        :     BOOL;
END_VAR
```

Implemen
tation

```
diLen := STu.StrLenA(pstData := psTo);
IF xConcat THEN
    xConcat := FALSE;
    IF NOT Stu.StrIsNullOrEmptyA(pstData := psFrom) THEN
        // Create an string with more than 255 characters
        //Concat-Function
        WHILE Stu.StrConcatA(pstFrom := psFrom,pstTo := psTo, SIZEOF(sTo) - 8) DO
        // Dummy line
        END_WHILE
        // Copy the whole string
        diCopied := Stu.StrCpyA(pBuffer := ADR(sCopied), SIZEOF(sCopied), pStr := psTo);
        sFrom := 'CODESYS';
        Stu.StrConcatA(pstFrom := psFrom,pstTo := psTo, SIZEOF(sTo));
        // Find position of the first substring
        iFind := Stu.StrFindA(pst1 := psTo, pst2 := psFrom, uiSearchStart := 1);
        // Copy just a part of the string
        Stu.StrMidA(pst := psTo, uiInputBufferSize := SIZEOF(sTo), iLength := 5, iPosition :=
iFind , pstResult := ADR(sMid), uiResultBufferSize := uiMid);
    END_IF
END_IF


If xUpper Then
    xUpper := False;
```

```
    Stu.StrToUpperA(pString := pstUpper);
END_IF

IF xLower THEN
    xLower := FALSE;
    Stu.StrToLowerA(pString := pstUpper);
END_IF

IF xCaseCmp THEN
    xCaseCmp := FALSE;
    // caseinsensitive comparison !!
    iResult := Stu.StrCaseCmpA(pByte1 := ADR(sCmp1), pByte2 := ADR(sCmp2));
END_IF

IF xCaseCmpEnd THEN
    xCaseCmpEnd := FALSE;
    // sCmp1 := 'Hello'; sCmp2 := 'llO';
    //=> Retdurns "0" = for equal
    iResult := Stu.StrCaseCmpEndA(pString := ADR(sCmp1), pSuffix := ADR(sCmp2));
END_IF

IF xCaseCmpStart THEN
    xCaseCmpStart := FALSE;
    // sCmp1 := 'Hello'; sCmp2 := 'HELL';
    //=> Retdurns "0" = for equal
    iResult := Stu.StrCaseCmpStartA(pString := ADR(sCmp1), pPrefix := ADR(sCmp2));
END_IF

IF xCmpEnd THEN
    xCmpEnd := FALSE;
    // sCmp1 := 'Hello'; sCmp2 := 'llO';
    //=> Retdurns "-1" = for not equal
    iResult := Stu.StrCmpEndA(pString := ADR(sCmp1), pSuffix := ADR(sCmp2));
END_IF

IF xCmpStart THEN
    xCmpStart := FALSE;
    // sCmp1 := 'Hello'; sCmp2 := 'HELL';
    //=> Retdurns "-1" = for not equal
    iResult := Stu.StrCmpStartA(pString := ADR(sCmp1), pPrefix := ADR(sCmp2));
END_IF

IF xDelete THEN
    xDelete := FALSE;
    Stu.StrDeleteA(pby := ADR(sDelete), iLength := 2, iPosition := 4);
END_IF

IF xTrim THEN
    xTrim := FALSE;
    sTrim := '  The spaces   will removed   ';
    Stu.StrTrimA(pString := ADR(sTrim)); // Space character on the left and the right side will
removed
END_IF

IF xTrimEnd THEN
    xTrimEnd := FALSE;
    sTrim := '  The spaces   will removed   ';
    Stu.StrTrimEndA(pString := ADR(sTrim)); // Only the space character on the right side will
removed
END_IF

IF xTrimStart THEN
    xTrimStart := FALSE;
    sTrim := '  The spaces   will removed   ';
    Stu.StrTrimStartA(pString := ADR(sTrim)); // Only the space character on the left side will
removed
END_IF

IF xReplace THEN
    xReplace := FALSE;
    Stu.StrReplaceA(
```

```
    pstInput:= ADR(sOldReplace),
    uiInputBufferSize:= SIZEOF(sOldReplace),
    pstReplaceWith:= ADR(sReplace),
    iLengthInput:= DINT_TO_INT(Stu.StrLenA(ADR(sOldReplace))),
    iLengthToReplace:= 1, // 1 = only the space character will replaced; 2 = " W" will replaced
    iLengthToReplaceWith:= DINT_TO_INT(Stu.StrLenA(ADR(sReplace))),
    iPosition:= 6);
END_IF
```

- Start the project and, for example, set a breakpoint in line 11 of the POU *PLC_PRG*. Then step through the single function samples.