Error Checking with "POU for implicit checks"

Sometimes it happens that the application on the controller crashes mysteriously with a message such as "Access violation". Double-clicking the "Source position" in the PLC log jumps to the code position, showing crash positions that do not appear to have any connection to the problem (for example, a simple TON box or a library). Another indication for these types of errors is when the code position changes when the application is changed.

The common cause is writing beyond the bounds of an array.

This can be detected, for example, by using "POU for implicit checks", and here particularly with the function "CheckBounds".

For reasons of performance, "POU for implicit checks" should be removed from the project after debugging. It is not enough to rename it.

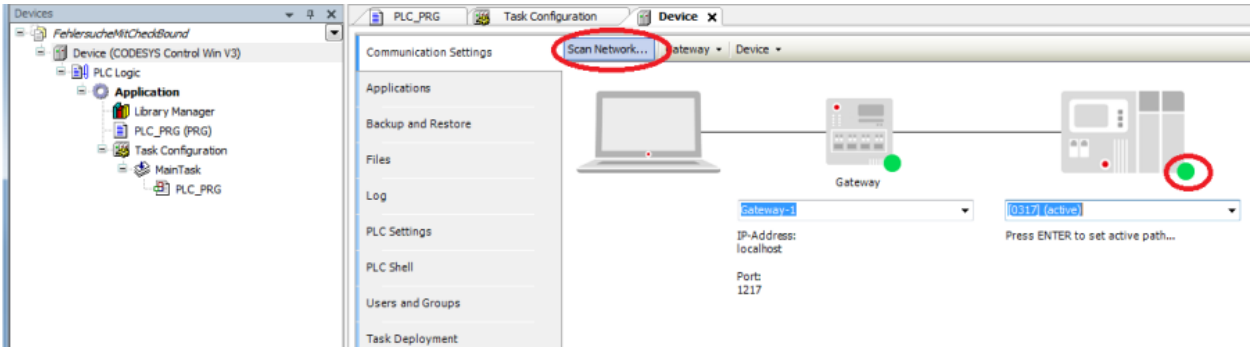
The implicit checks are performed only for code within the project.

If libraries are to be checked, then the compiler definition *checks_in_libs* must be set.

Example:

Requirement

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



- Adapt the "PLC_PRG" POU as follows:

Declaration

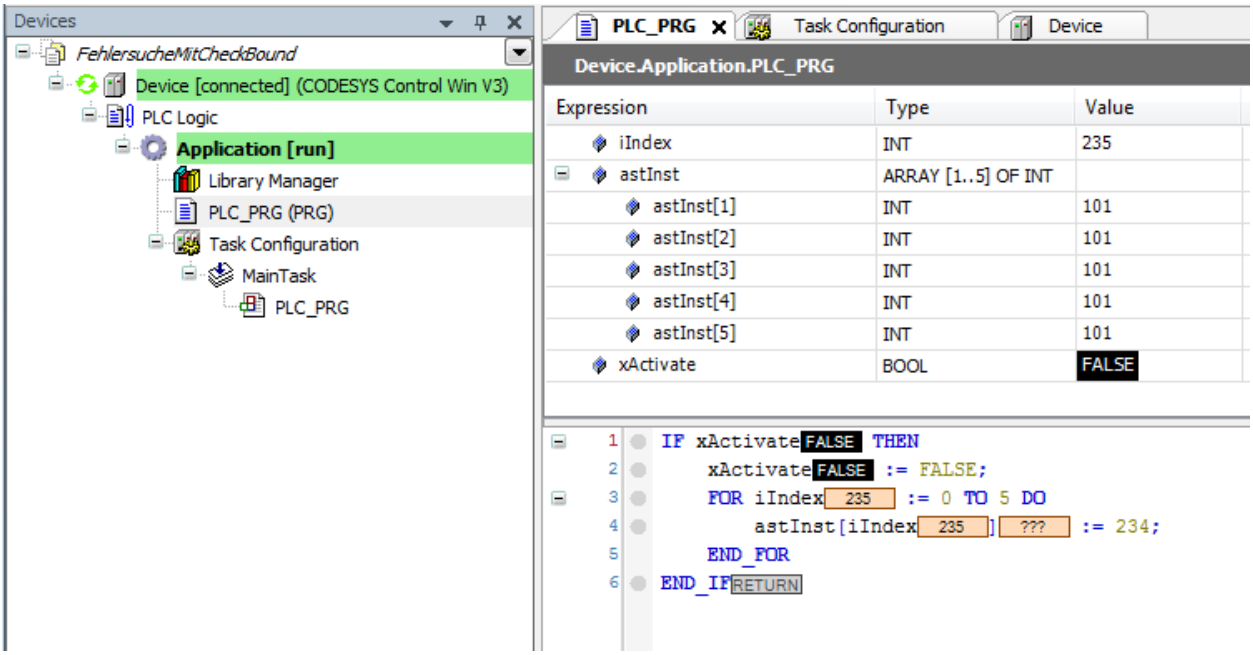
```
VAR

      iIndex : INT;
      astInst : ARRAY [1..5] OF INT := [5
      (101)];
      xActivate : BOOL;

END_VAR
```

Implementation

```
IF xActivate THEN
    xActivate := FALSE;
    FOR iIndex := 0 TO 5 DO
        astInst[iIndex] := 234;
    END_FOR
END_IF
```

Downloading and starting the project

- Download the project to the controller and start the application.
- The following image is displayed after you set the xActivate variable:



The expected result is that all elements of the array are set to the value "234".

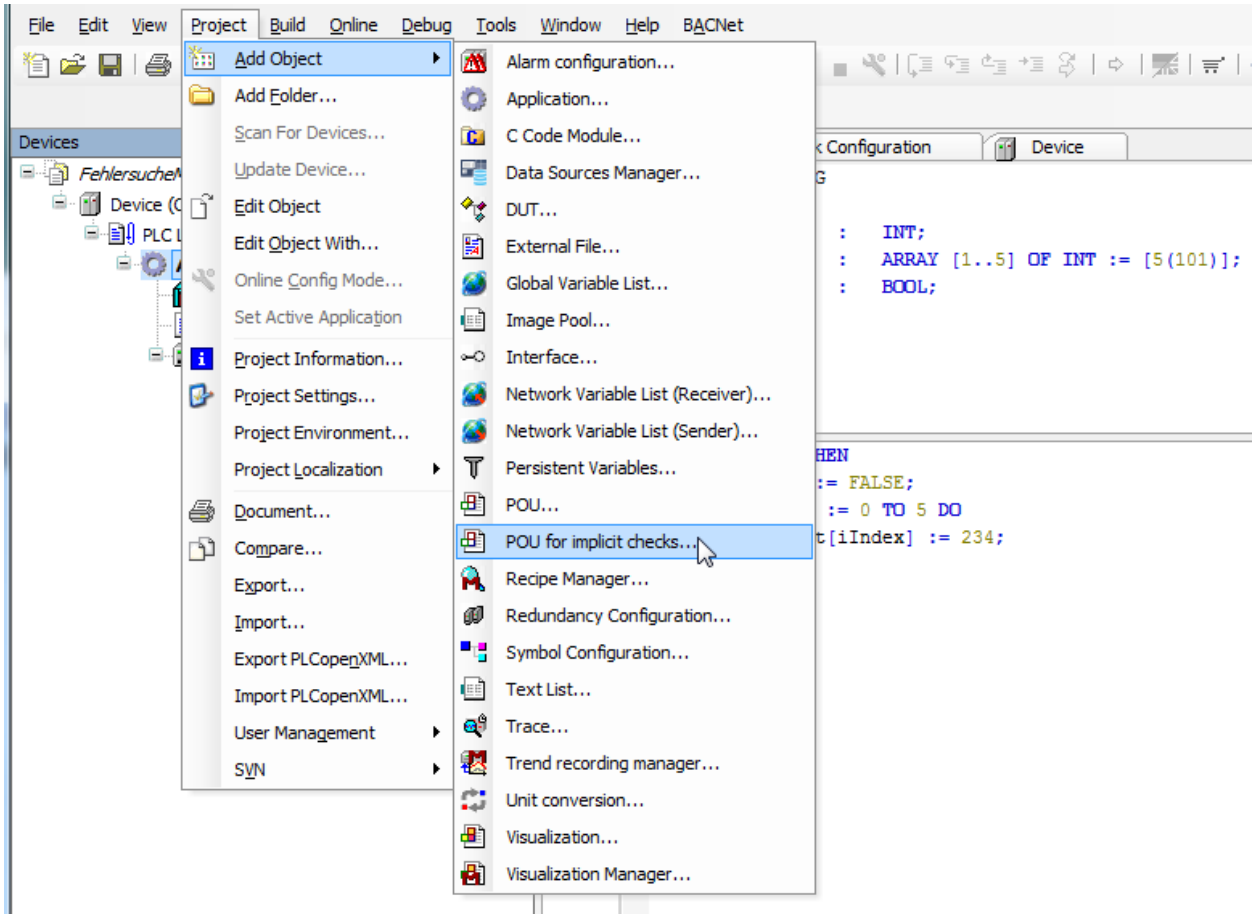
As the lower bound of the array is set to "1", the memory area of the counter variable is described incorrectly with 234 in the first executed loop (iIndex has value "0") because it is located in the memory area before the array.

In the next executed loop, it is detected that the cancellation condition of the loop is fulfilled, and as a result no element of the array will be written.

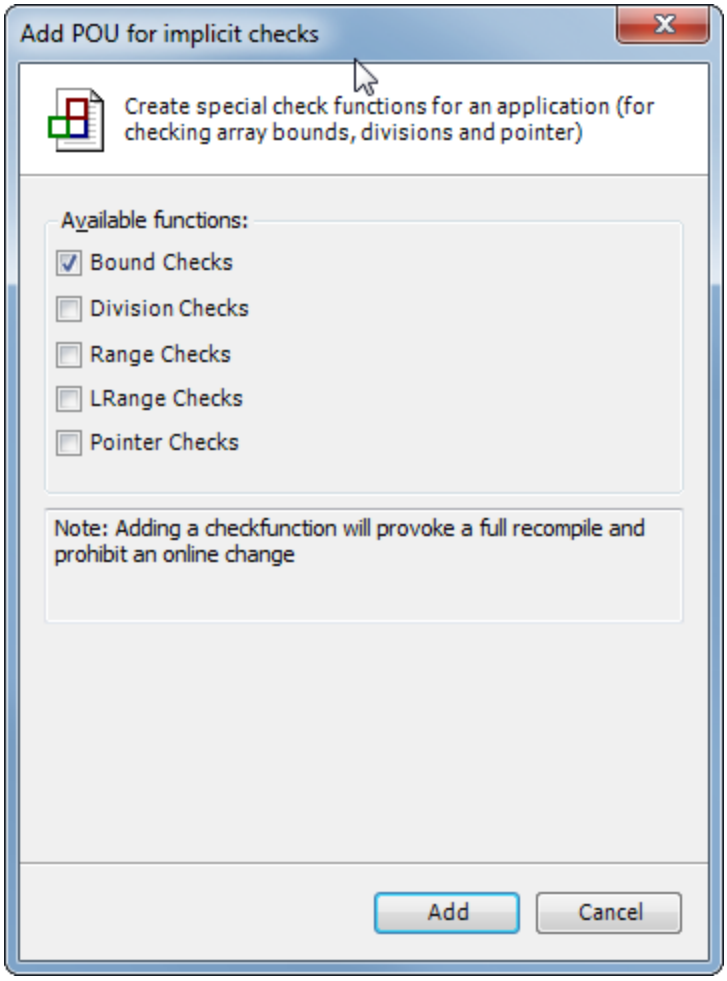
These kinds of mistakes in the memory can have unexpected results, as severe are the controller crashing.

Including the function "CheckBounds"

- Add the object "POU for implicit checks" to the project:



The following dialog opens automatically, where the type of check can be selected.
Select the option Bound checks.

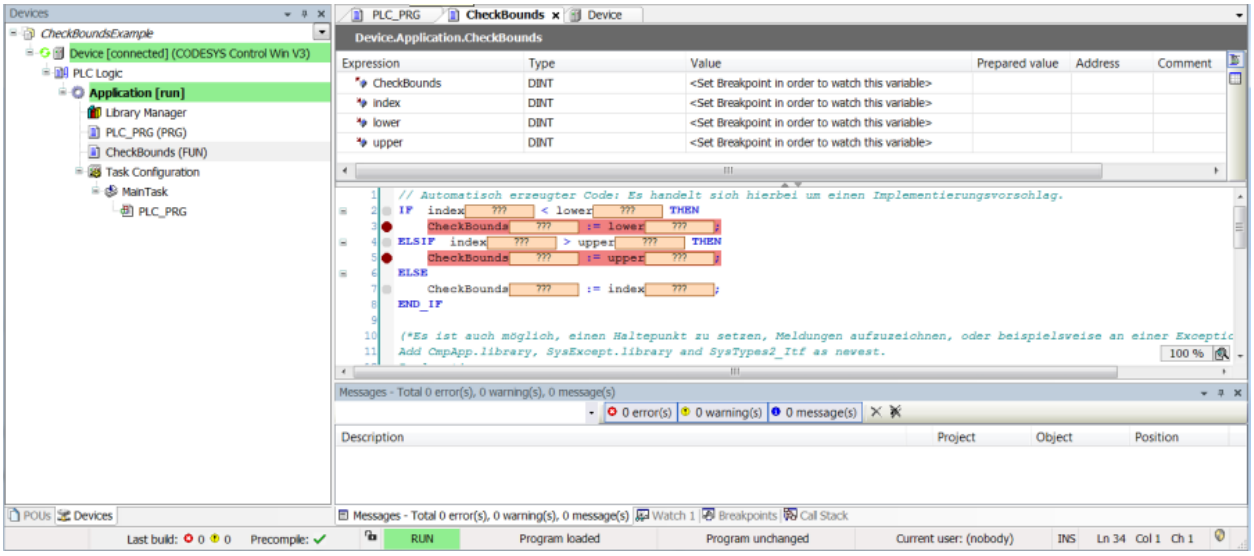


Only one option should be selected. The check may have to be repeated with another option.

Using the function "CheckBounds"

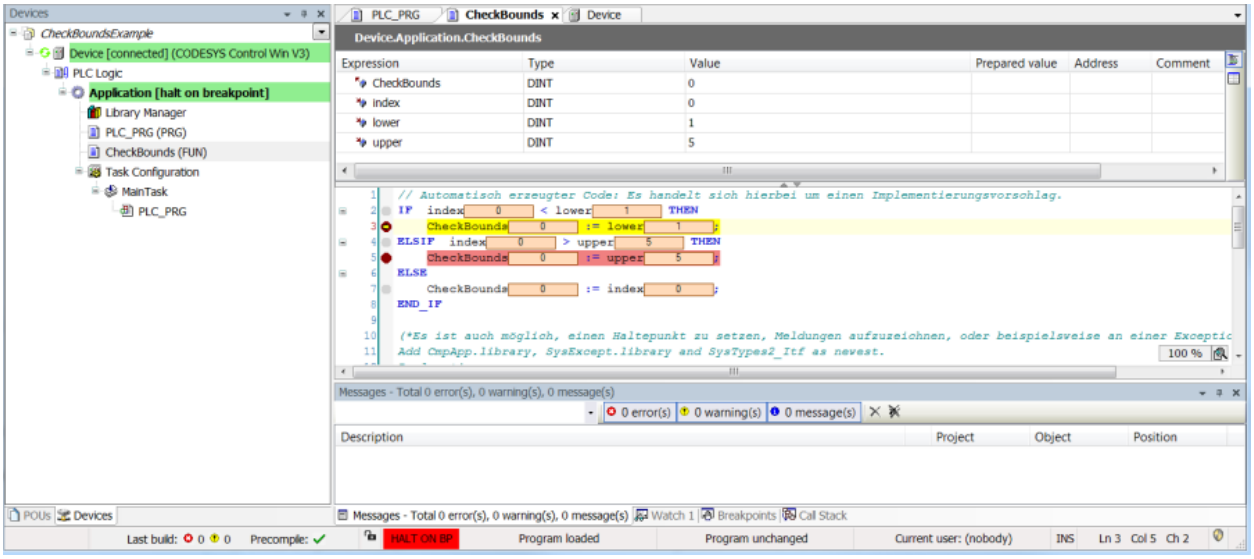
- Download the project to the controller and start the application.
- Set a breakpoint at the desired check.

If a bound violation is detected, then the project is halted at the breakpoint.

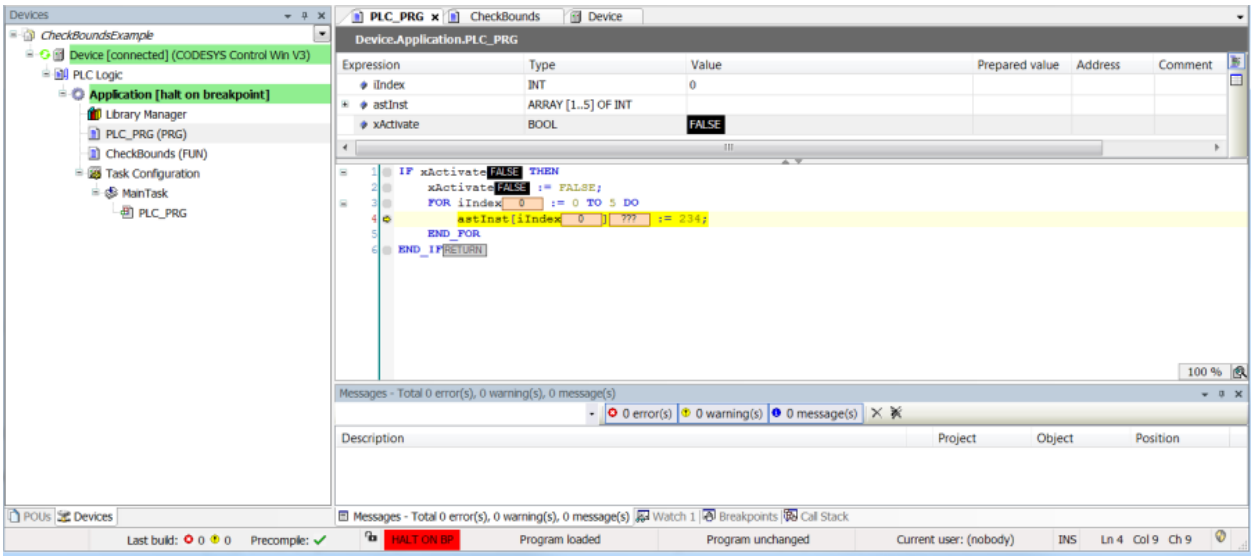


- Switch to the PLC_PRG POU and set the xActivate variable to TRUE.

The projects stops at the breakpoint regarding lower bound violation.



- Exit the CheckBounds function by stepping (press F10 two times) and the debugger jump automatically to the position where the boundary violation was detected:



- As an alternative, the call stack can also be displayed and and from there a jump made to the relevant position: