Supervisar acceso Arrays – Función CheckBounds

SoMachine V4.1 SP1.1			Autor: Cynthia Garibo Pérez
Versión	Autor	Fecha	Comentarios
V 1.0	C. Garibo	17/06/15	Creación del documento

Las variables tipo Array, se definen por su tamaño y por el tipo de datos que contienen. Por ejemplo: MyArray [0..10] of INT.

Normalmente se accede a los datos según la posición de la variable Array en la que estén contenidos (MyArray [3]), pero, ¿Qué pasa si accedemos a una posición que esté fuera de los límites de la variable (MyArray[15])?

Si accedemos a zonas de memoria no definidas, puede que causemos errores aleatorios e inesperados.

Existe una función de supervisión, CheckBounds, que evita que se acceda a posiciones que no están definidas dentro de un Array.

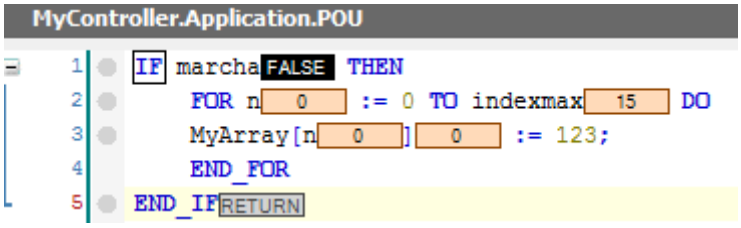
1. Importancia de la función CheckBounds

Tenemos:

- Una variable matriz MyArray [0…10] OF INT
- El siguiente programa:

```
1  IF marcha THEN
2      FOR n := 0 TO indexmax DO
3      MyArray[n] := 123;
4      END_FOR
5  END_IF
```

- Escribimos en MyArray desde la posición 0 a la 15 →DESBORDAMIENTO



Para comprobar si se está escribiendo en alguna zona de memoria tras pasar MyArray[10], se ha direccionado la matriz en %MW0 (%MB0..%MB20). De esta manera, si añadimos una variable en %MB22, podremos ver si se sigue escribiendo.

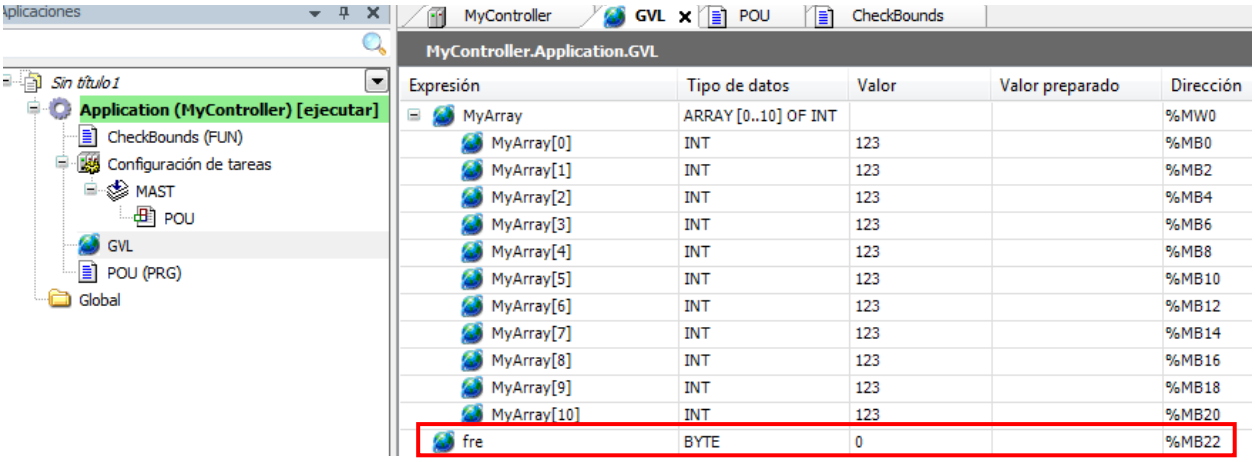
MyController.Application.GVL

Expresión	Tipo de datos	Valor	Valor preparado	Dirección
MyArray	ARRAY [0..10] OF INT			%MW0
MyArray[0]	INT	123		%MB0
MyArray[1]	INT	123		%MB2
MyArray[2]	INT	123		%MB4
MyArray[3]	INT	123		%MB6
MyArray[4]	INT	123		%MB8
MyArray[5]	INT	123		%MB10
MyArray[6]	INT	123		%MB12
MyArray[7]	INT	123		%MB14
MyArray[8]	INT	123		%MB16
MyArray[9]	INT	123		%MB18
MyArray[10]	INT	123		%MB20
fre	BYTE	123		%MB22

Por lo tanto, tras un desbordamiento de la matriz, se sigue escribiendo, pudiendo afectar a otras variables.

- Realizamos la misma operación, tras añadir el bloque de función CheckBounds

  Resultado: No escribimos fuera de los límites del Array

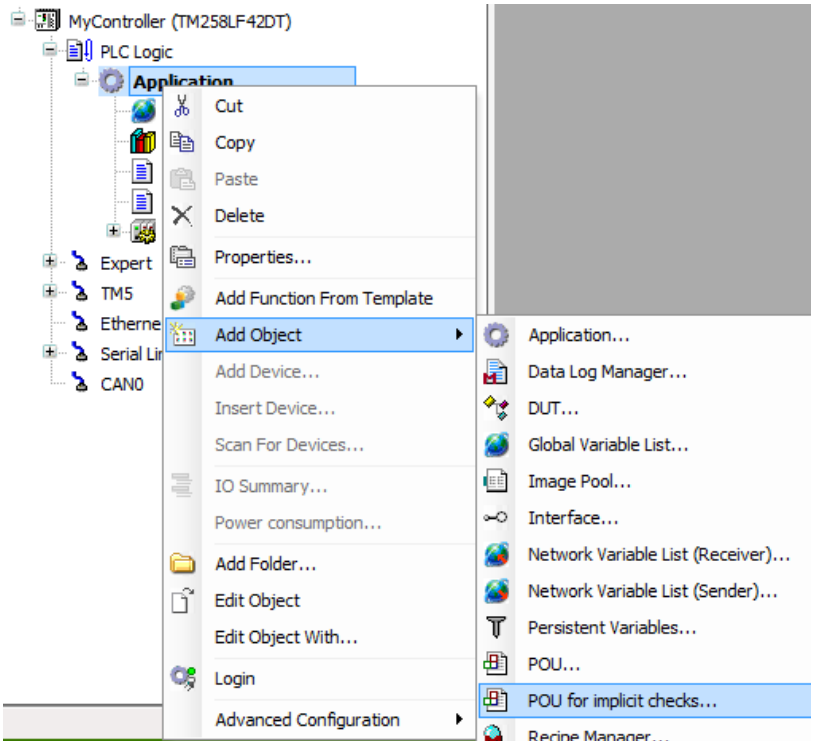


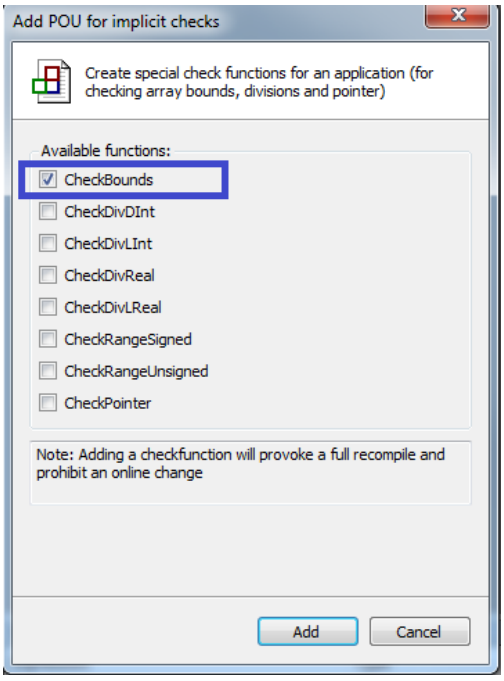
2. Añadir la función CheckBounds

Aplicación→Añadir objeto → POUs para comprobaciones implícitas → CheckBounds

Un POU con el siguiente código se añade a la aplicación:

```
IF  index < lower THEN
     CheckBounds := lower;
ELSIF  index > upper THEN
     CheckBounds := upper;
ELSE
     CheckBounds := index;
END_IF
```





3. Detectar accesos incorrectos a un Array

- Declarar 2 variables globales:

```
VAR_GLOBAL
     indexTooHighCounter : INT;
     indexTooLowCounter : INT;
END_VAR
```

- Modificar la función CheckBounds de la siguiente manera:

```
IF  index < lower THEN
  CheckBounds := lower;
  indexTooLowCounter := indexTooLowCounter + 1;
ELSIF  index > upper THEN
  CheckBounds := upper;
  indexTooHighCounter := indexTooHighCounter + 1;
ELSE
  CheckBounds := index;
END_IF
```

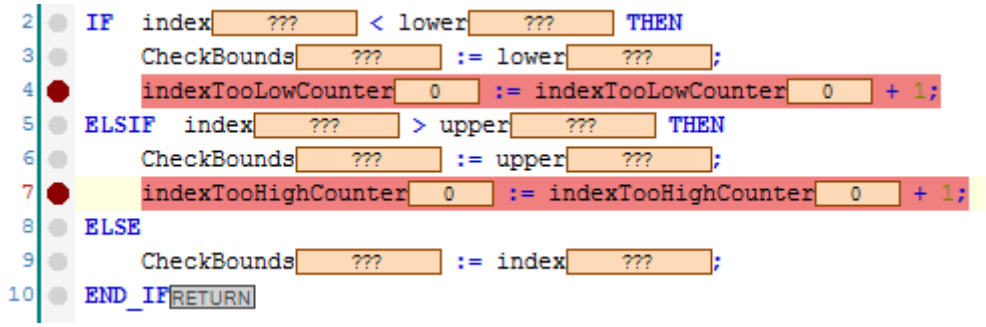
Cada vez que ocurra un desbordamiento, la variable contador incrementará su valor.

Si tras un tiempo de funcionamiento, ambos contadores permanecen a 0, significa que no se producen desbordamientos. Durante dicho tiempo en funcionamiento, todas las zonas de programa deberán ejecutarse.

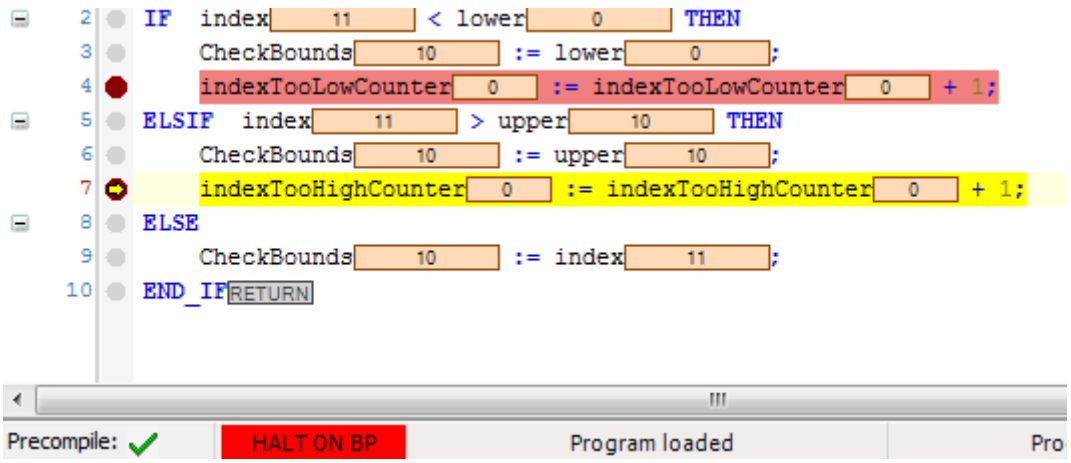
Si los contadores no están a 0, es necesario saber donde se produce un desbordamiento y corregir el programa.

4. Detectar donde se produce un desbordamiento

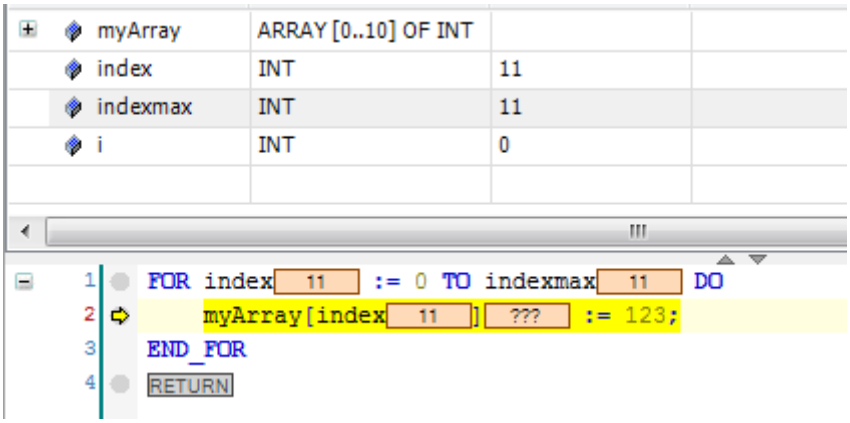
Se necesita usar puntos de interrupción.
Para añadir un punto de interrupción, seleccionamos la línea y pulsamos F9.
Tenemos que añadir dos puntos de interrupción:



Cuando la variable Indexmax pasa a 11(sale de los límites del array), el punto de interrupción de la línea 7 se activa.



Ahora pulsamos F10 dos veces y vemos donde se ha producido el desbordamiento:



Efectivamente, debido a que se ha incrementado el valor de 'index' dentro de un FOR sin controlar el límite del Array, se ha producido un desbordamiento.