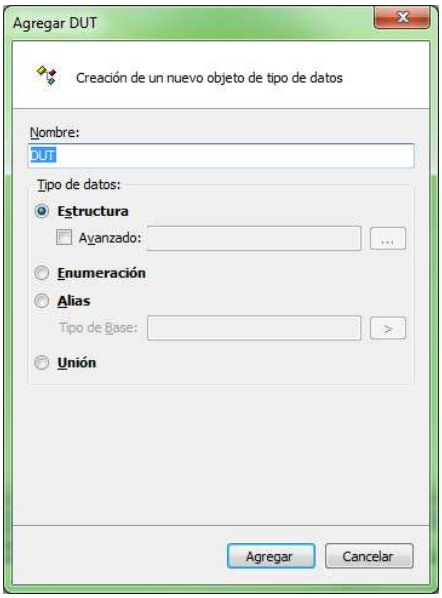
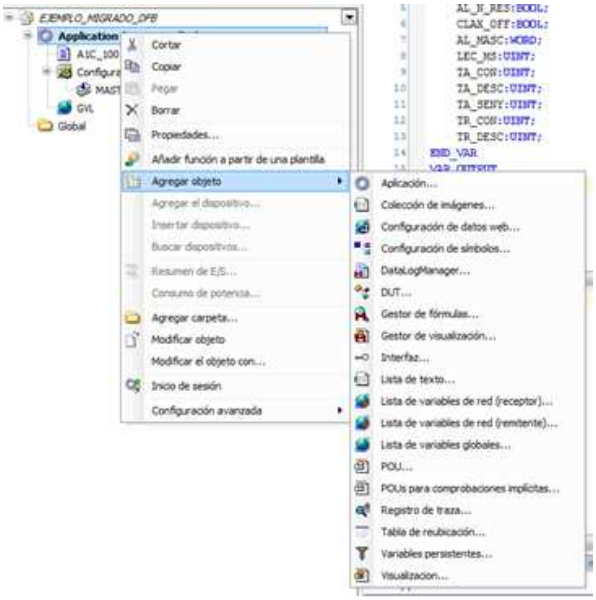
ESTRUCTURAS DE DATOS SOMACHINE

SOMACHINE 4.1 SP1.1			Autor: César Rufo Domínguez
Versión	Autor	Fecha	Comentarios
V 1.0	César Rufo	29/4/15	
V 2.0	César Rufo	5/8/15	Consideraciones sobre ocupación de memoria al ubicar en %ML

Para crear variables de tipo estructura en Somachine, a semejanza de los tipos de datos que pueden definirse en Unity, se deben seguir los siguientes pasos:



Para poder agregar un DUT se hace clic derecho sobre la raíz de la aplicación y se elige la opción agregar objeto:



Se elige el nombre del tipo de datos que se quieren crear. En nuestro ejemplo lo llamaremos T_Temp. En el fondo, se va a tratar de un array de 10 elementos tipo WORD. La única restricción es que no se permite la declaración AT para utilizar variables alocatadas en la definición del propio tipo, pero sí al usarlo en programa.

A1C_100 | T_Temp

```
1  TYPE T_Temp :
2  STRUCT
3      T_Temp:ARRAY [1..10] OF WORD;
4  END_STRUCT
5  END_TYPE
6
```

Se pueden crear igualmente objetos más complejos y anidados:

A1C_100 | T_Config

```
1  TYPE T_Config :
2  STRUCT
3      Bits:WORD;
4      Registros: T_Temp;
5      T_Most:UINT;
6  END_STRUCT
7  END_TYPE
8
```

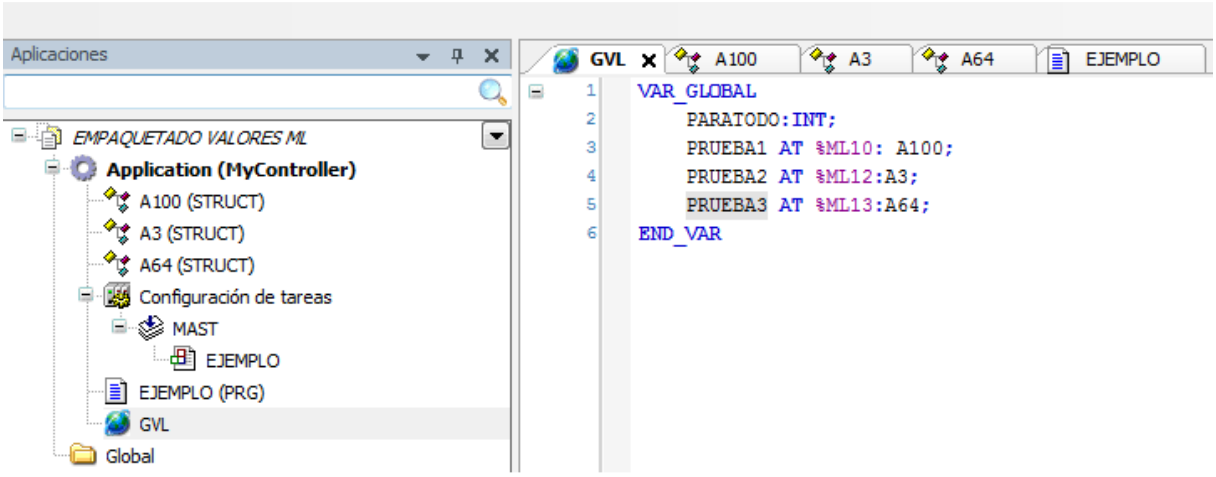
Así, por ejemplo, en una declaración de una variable en otra parte del programa usando el tipo T_Config éste ya queda perfectamente admitido:

A1C_100

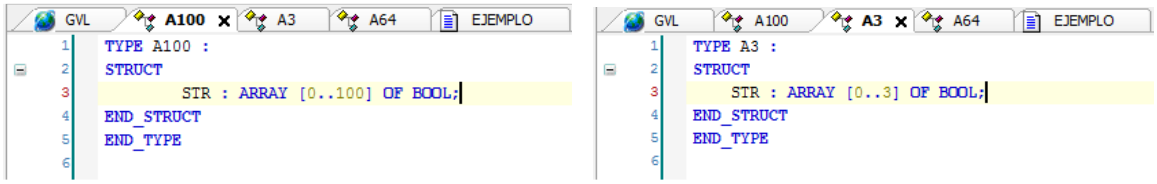
```
1  FUNCTION_BLOCK A1C_100
2  VAR_INPUT
3      ULT_EST:BOOL;
4      AL_N_REM:BOOL;
5      AL_N_RES:BOOL;
6      CLAX_OFF:BOOL;
7      AL_MASC:WORD;
8      LEC_MS:UINT;
9      TA_CON:UINT;
10     TA_DESC:UINT;
11     TA_SENY:UINT;
12     TR_CON:UINT;
13     TR_DESC:UINT;
14 END_VAR
15 VAR_OUTPUT
16     CFG: T_Config;
17 END_VAR
18 VAR
19 END_VAR
20
```

Si se necesita alocatar una variable de tipo estructura sí es posible llevarlo a cabo como una variable más, pero hay consideraciones importantes que deben tenerse en cuenta:

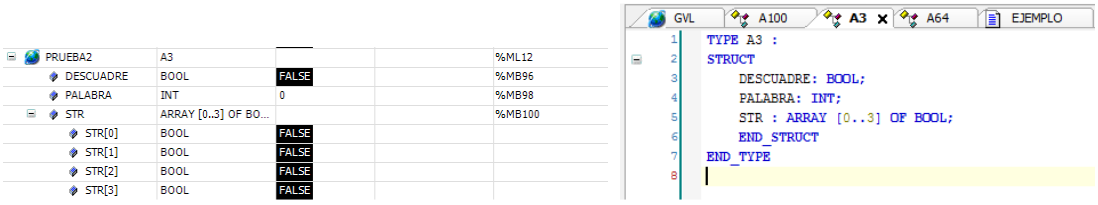
1.- Hay que declararlas en variables %ML (%ML10 ocupa el mismo lugar que %MW40).



2.- Hay que tener cuidado con los solapamientos de memoria y calcular correctamente la dimensión ocupada en %MW. Es fácil cuando los objetos son de tipo palabra o múltiplos pero para bytes o bits hay que tener presente que tras un byte impar se salta a la siguiente palabra y los bits son considerados como byte. Así, en el caso anterior por ejemplo, A3 estaría en %MW48 y %MW49 (o más exactamente en %MB96, %MB97, %MB98 y %MB99).



En el caso de colocar 101 booleanos a partir de %ML10 (%MW40), las 51 palabras resultantes entrarían en conflicto con lo ya establecido en %MW48 y %MW49. Incluso con la zona a partir de %ML13.



Sirva como ejemplo la modificación anterior de la estructura A3 para ilustrar las asignaciones reales en bytes según los datos que forman parte de la misma.