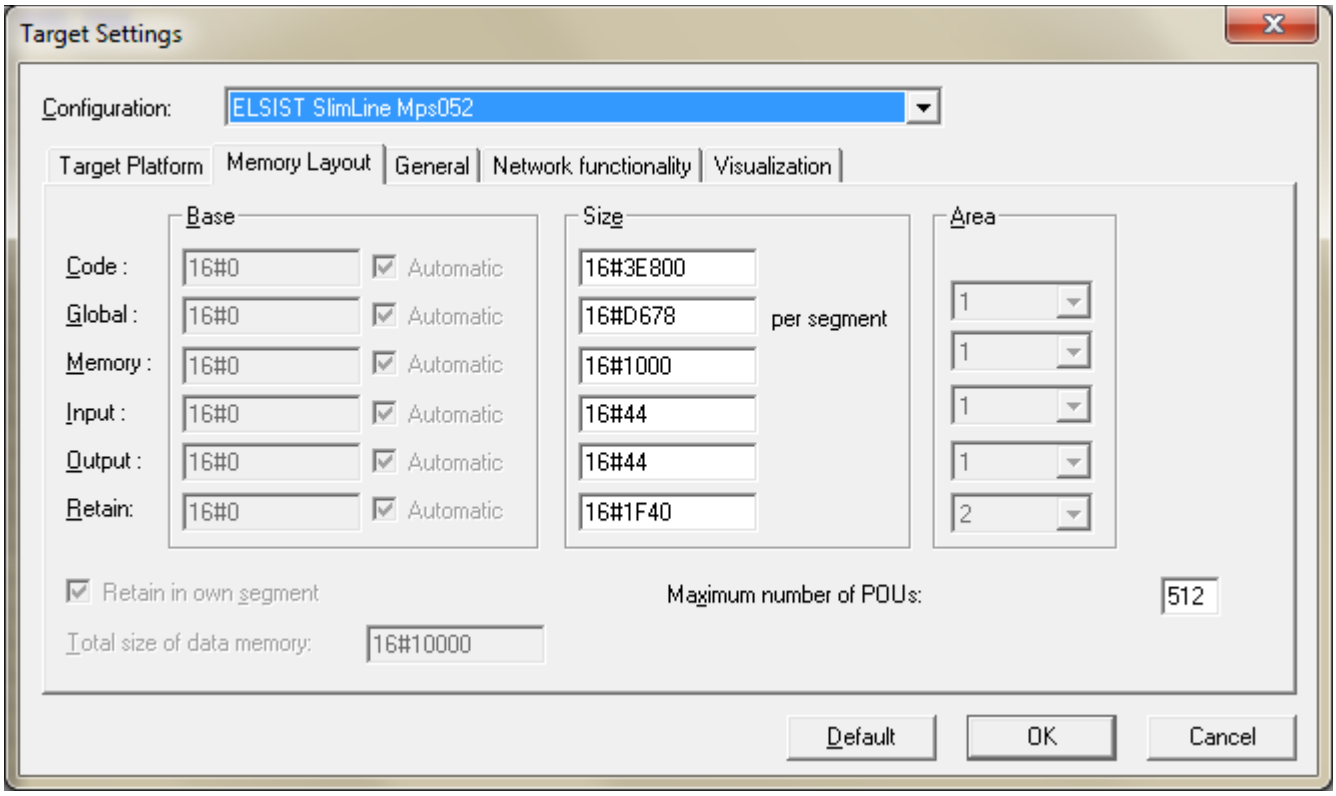
1.5 Configurazione memoria

Di default nei settaggi del target il layout di memoria è configurato nel modo:



Memoria codice (FLASH), dimensionata a 256Kb (16#3E800): In questa area si trova il programma PLC sviluppato con CODESYS.

Memoria dati (RAM) dimensionata a 64Kb (16#10000): In quest'area si trovano tutte le aree dati gestite da CODESYS. In particolare avremo:

> **Memoria variabili globali 54Kb (16#D678):** In quest'area sono allocate tutte le variabili mnemoniche definite e le variabili automatiche generate dal compilatore (Esempio ***MyVar: UDINT;***).
>
> **Memoria allocazione variabili 4Kb (16#1000):** In quest'area sono allocate tutte le variabili mnemoniche il cui indirizzo è definito dall'utente con la direttiva AT (Esempio ***MyVar AT %MB100: USINT;***).
>
> **Immagine ingressi logici 68 bytes (16#44):** In quest'area è allocata l'immagine di processo degli ingressi logici. 16 moduli + modulo CPU 4 bytes per ogni modulo.
>
> **Immagine uscite logici 68 bytes (16#44):** In quest'area è allocata l'immagine di processo delle uscite logiche. 16 moduli + modulo CPU 4 bytes per ogni modulo.
>
> **Area header POU 6Kb (12 bytes per 512 POU):** Ogni POU utilizza un header di memoria di 12 bytes.

Memoria retain (RAM) dimensionata a 8Kb (16#1F40): In quest'area sono allocate tutte le variabili mnemoniche definite dall'utente con l'attributo RETAIN. Esempio di dichiarazione:

```
VAR_GLOBAL RETAIN
   MyVar1: INT; (* Retain INT variable *)
   MyVar2: UINT; (* Retain UINT variable *)
   MyVar3: UDINT; (* Retain UDINT variable *)
END_VAR
```

1.6 Allocazione variabili in memoria

Come visto nel paragrafo precedente esiste un'area di memoria da 4Kb per l'allocazione delle variabili, l'utente nel suo programma può definire liberamente le variabili in quest'area, consiglio di eseguire l'allocazione nello spazio dedicato alle variabili globali in modo da trovare facilmente le aree occupate. Come si vede dalla tabella di allocazione è possibile allocare:

Bit	E' possibile allocare 32768 variabili BOOL. Range da MX 0.0 a MX 2047.15
8 Bits	E' possibile allocare 4096 variabili BYTE, USINT, SINT. Range da MB 0 a MB 4095
16 Bits	E' possibile allocare 2048 variabili WORD, UINT, INT. Range da MW 0 a MW 2047
32 Bits	E' possibile allocare 1024 variabili DWORD, UDINT, DINT, REAL. Range da MD 0 a MD 1023

La definizione può avvenire nel modo:

```
VAR_GLOBAL
   MyBOOL AT %MX2047.15: BOOL; (* Variabile BOOL *)
   MyUSINT AT %MB4095: USINT; (* Variabile USINT *)
   MyUINT AT %MW2047: UINT; (* Variabile UINT *)
   MyUDINT AT %MD1023: UDINT; (* Variabile UDINT *)
END_VAR
```

Da quanto detto sopra è evidente che una assegnazione del tipo:

```
VAR_GLOBAL
   MyVarLSB AT %MB1024: USINT; (* Least significant BYTE variabile *)
   MyVarMB1 AT %MB1025: USINT; (* Middle BYTE variabile *)
   MyVarMB2 AT %MB1026: USINT; (* Middle BYTE variabile *)
   MyVarMSB AT %MB1027: USINT; (* Most significant BYTE variabile *)

   MyVarLSW AT %MW512: UINT; (* Least significant WORD variabile *)
   MyVarMSW AT %MW513: UINT; (* Most significant WORD variabile *)

   MyVar AT %MD256: UDINT; (* Variabile UDINT *)
END_VAR
```

Fà riferimento ad una variabile UDINT suddividendola in due variabili WORD e quattro variabili BYTE. Ricordo che il sistema è Little-Endian. Ecco lo screnshoot del debugger.