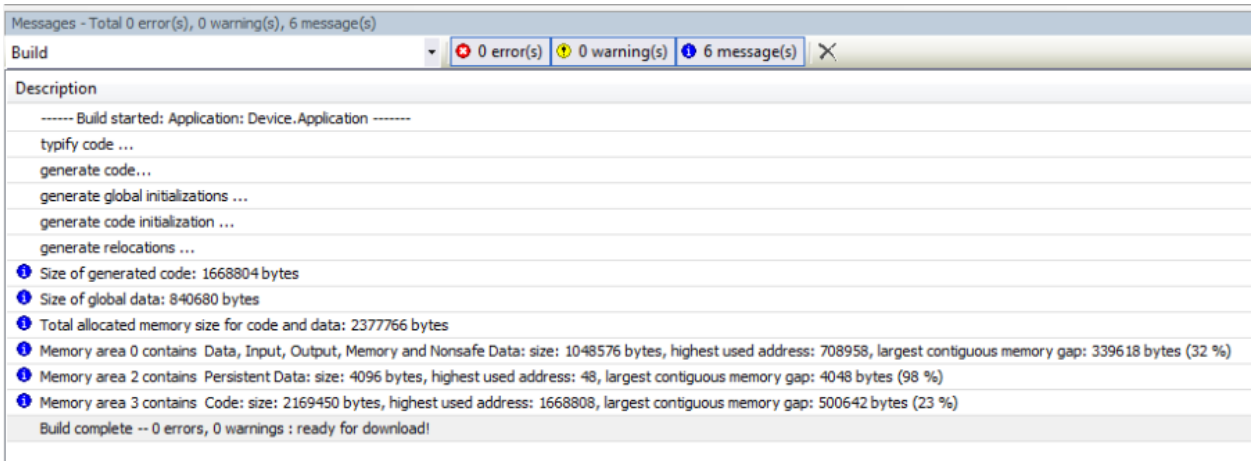
What are the meanings of the memory occupation data in the message window?

Each time code is generated, data for the memory occupation are displayed in the message window. The terminology used is explained below.



Size of generated code
The sum of all code pieces produces the size of the generated code.

Size of global data
The size of the global data indicates the entire memory occupied by the global variables. Inputs and outputs are not taken into account in this, unless inputs or outputs are mapped in the area of the global variables.

Total allocated memory size
Encompasses the entire allocated memory, including the gaps caused by incremental compilation or online change. In addition, this area also contains memory that is reserved for online change and is still unused for the time being; i.e. it contains no code and no data. The memory that is occupied corresponds to the **highest used address**.

Highest used address
This is the highest occupied address in the memory range. When compiling for the first time, the memory addresses are issued to variables in ascending order, taking into account the alignment (as a rule 8 bytes). The highest used address corresponds approximately to the memory used.

Largest contiguous memory gap
In the case of an online change, memory is used only for new variables and new code. Memory previously occupied by deleted variables and code is freed up. This can result in gaps in the memory. After many online changes the memory may become fragmented, i.e. many small gaps are created that can no longer be used.

The largest contiguous memory gap is the assured size that can still be used in any case for further data. Memory gaps are used, of course. If, for example, a global variable of the type byte is added, it is placed in the first free byte in the memory. An FB instance, a variable of the type structure or array, or the code for a POU have to be saved contiguously and therefore occupy more memory accordingly.
Therefore one can only assure the largest contiguous free memory.