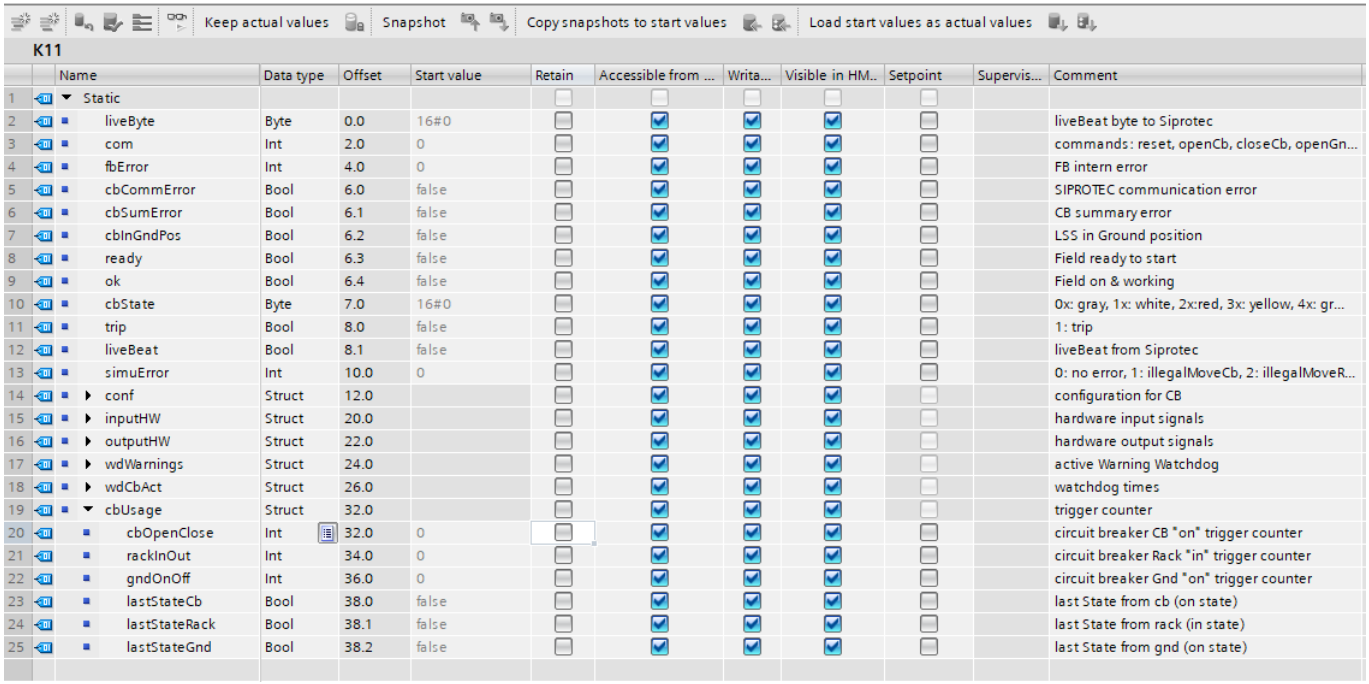
Data block (DB)

“DB” is an abbreviation for DATA_BLOCK or the German “Datenbaustein” and is used to denote a data area. A DB can contain any set of data types allowed and defined on the given PLC. A DB can have a maximum size of 64 kByte. The size of the PLC data area limits the total space occupied by all DBs. It is important to note that the PLC is not optimized for storing large amounts of data; therefore, we do not store images, music, files, or even large text files in a DB. The TIA-Portal uses the small blue barrel symbol ( ) to denote DBs. The image below shows the contents of a DB, along with some settings:



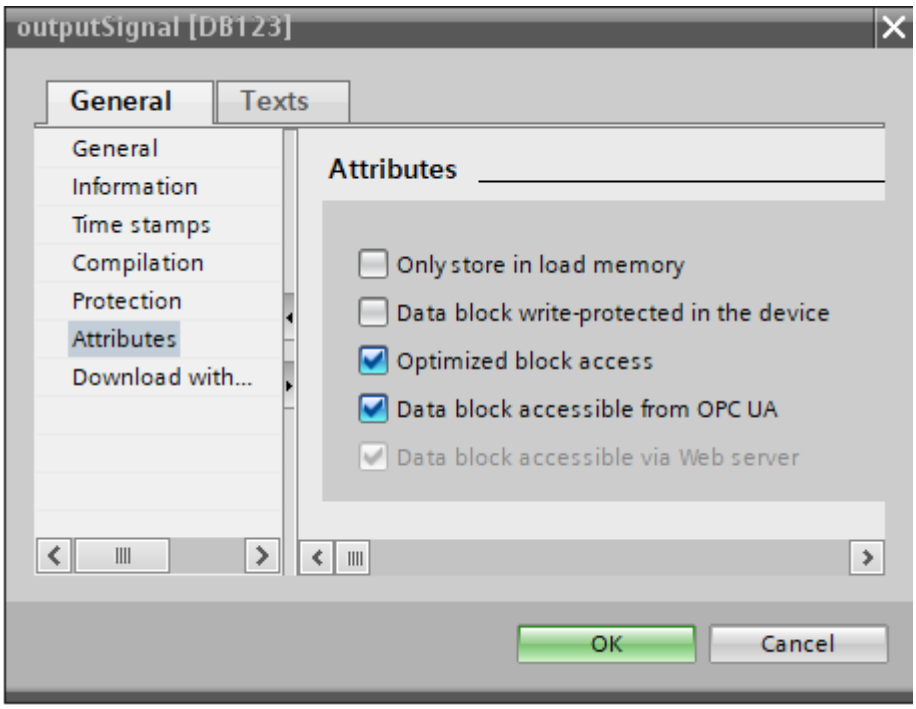
	Name	Data type	Offset	Start value	Retain	Accessible from ...	Writa...	Visible in HM...	Setpoint	Supervis...	Comment
1	Static										
2	liveByte	Byte	0.0	16#0							liveBeat byte to Siprotec
3	com	Int	2.0	0							commands: reset, openCb, closeCb, openGn...
4	fbError	Int	4.0	0							FB intern error
5	cbCommError	Bool	6.0	false							SIPROTEC communication error
6	cbSumError	Bool	6.1	false							CB summary error
7	cbInGndPos	Bool	6.2	false							LSS in Ground position
8	ready	Bool	6.3	false							Field ready to start
9	ok	Bool	6.4	false							Field on & working
10	cbState	Byte	7.0	16#0							0x: gray, 1x: white, 2x:red, 3x: yellow, 4x: gr...
11	trip	Bool	8.0	false							1: trip
12	liveBeat	Bool	8.1	false							liveBeat from Siprotec
13	simuError	Int	10.0	0							0: no error, 1: illegalMoveCb, 2: illegalMoveR...
14	conf	Struct	12.0								configuration for CB
15	inputHW	Struct	20.0								hardware input signals
16	outputHW	Struct	22.0								hardware output signals
17	wdWarnings	Struct	24.0								active Warning Watchdog
18	wdCbAct	Struct	26.0								watchdog times
19	cbUsage	Struct	32.0								trigger counter
20	cbOpenClose	Int	32.0	0							circuit breaker CB "on" trigger counter
21	rackInOut	Int	34.0	0							circuit breaker Rack "in" trigger counter
22	gndOnOff	Int	36.0	0							circuit breaker Gnd "on" trigger counter
23	lastStateCb	Bool	38.0	false							last State from cb (on state)
24	lastStateRack	Bool	38.1	false							last State from rack (in state)
25	lastStateGnd	Bool	38.2	false							last State from gnd (on state)

The columns are as follows:

Column	Desription
Name	The name of the variable within the DB. The variable names are unique, and the DB name is displayed in the upper-left corner, in this case: *K11*. The variable names are supplemented with this, e.g., *“K11”.liveByte*. This also means that the DB can be copied and renamed one-for-one. That is, if this DB is copied and renamed to, for example, “K12”, the above reference will be *“K12”.liveByte*. In the case of a structure, for example, “*cbUsage*”, the entire structure depth must be defined, for example: *“K11”.cbUsage.cbOpenClose*.
Data type	The data type. Structures and arrays must be created when defining the DB by entering, for example, type Struct in the Data type field.
Offset	The offset of the variable within the DB. This appears only for non-optimized DBs. More details: optimized DB
Start value	The starting value of the given variables, which the PLC takes on when restarting. The default value can be overwritten in the cell.
Retain	Values to be retained when restarting. It can only be set for the entire DB, so it is worth grouping the values to be stored in a DB
Accessible from HMI/OPC UA/Web API	The value is accessible from external applications. For structures and arrays, the setting can only be defined for the entire block. OPC access can be enabled/disabled in the settings, see DB Properties.

Column	Desription
Writable from HMI/OPC UA/Web API	The given value can be written from external applications.
Visible in HMI engineering	The setting disables or enables the HMI integration of the variable. In addition to disabling HMI, OPC can also be enabled, see DB Properties.
Setpoint	This allows you to initialize values in a data block (DB) online while the CPU is in RUN mode.
Comment	Description of the function of the field.

DB Properties

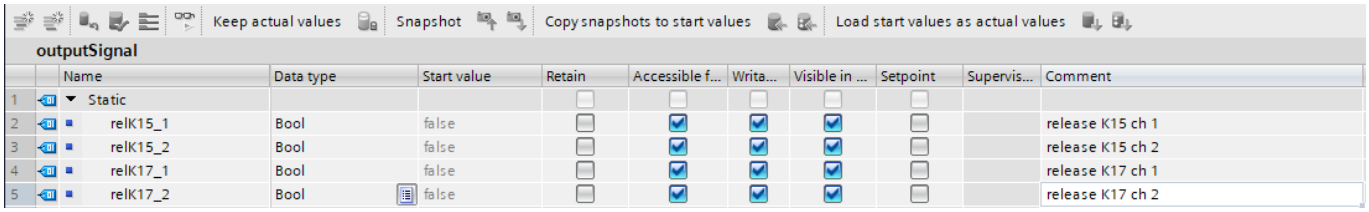


(*right-click on the DB → Properties..*)

Name the attribut	Description
Only store in load memory	This attribute is stored on the PLC's Micro Memory Card (MMC) or similar non-volatile storage, not in the CPU's working RAM, making it ideal for large, infrequently used data such as recipes or logs. It's accessed using special instructions like READ_DBL or WRIT_DBL to transfer data to/from working memory. This preserves precious working memory, but requires explicit programming to move data for active processing. The data survives power cycles but can be lost with a factory reset.
Data block write-protected in the device	Make the entire data block read-only.
Optimized block access	Optimized variable order within the DB. See below: Optimized DB.
Data block accessible from OPC UA	The data block can be accessed and published by OPC UA. See: OPC UA.

Optimized DB

Simatic groups variables in the optimized DB so they occupy as little storage space as possible. This means that it is “not visible from the outside” where a given data item is located within the storage space, i.e., in this case, the offset is not displayed in the editor window:



	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
1	Static									
2	relK15_1	Bool	false		☑	☑	☑			release K15 ch 1
3	relK15_2	Bool	false		☑	☑	☑			release K15 ch 2
4	relK17_1	Bool	false		☑	☑	☑			release K17 ch 1
5	relK17_2	Bool	false		☑	☑	☑			release K17 ch 2

On the one hand, this helps better utilize the PLC's storage space. Still, on the other hand, it makes operations that require direct addressing (communication modules - Modbus, direct addressing, etc.) impossible. In such cases, this option must be disabled in the settings (*right-click on the DB → Properties.. → Attributes → Optimized block access → OFF*)

Tags vs DB data

There are two basic methods for storing data in PLCs (in a simplified view). One involves placing variables in a global memory table alongside input and output variables, while the other uses data blocks (DBs). From my experience, storing data in DBs tends to be simpler and more straightforward for several reasons.

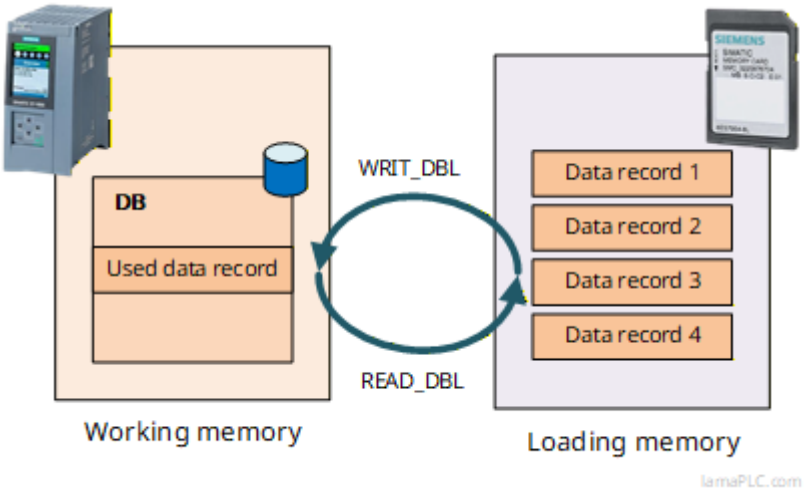
- Function-specific data can be stored in DBs. For example, the “motor1” DB contains only data for the 1st motor, but all of them (speed, load, temperature, on-off, errors, …)
- If someone wants to define a “motor2” as well, identical to “motor1” in terms of its parameters, they just need to copy the previous DB
- Cross-reference management of data immediately points to the given DB, from which we can immediately deduce their function
- If the data is already in the instant DBs assigned to the FBs, it is easy to embed them in a calling FB to use them as multi-instants.

Typically, I don't bother defining variables within the Tags; creating them directly in the databases suffices—though this is just my personal preference.

Storing DB records in the load memory

In PLCs, working memory is PLC-dependent and often very limited. We may have a lot of information that does not need to be read and written cyclically. Examples include recipe data (a list of technological components), parameter data, or database assignments that are needed only occasionally.

In these cases, one option is to store the data not in working memory but on the SD card and in load memory, and to transfer them only when needed using the “WRIT_DBL” and “READ_DBL” operations.



More information:
TIA Datatypes: S7 data types summary table

From:
http://lamaplc.com/ - **lamaPLC**

Permanent link:
http://lamaplc.com/doku.php?id=automation:db&rev=1767718781

Last update: **2026/01/06 16:59**