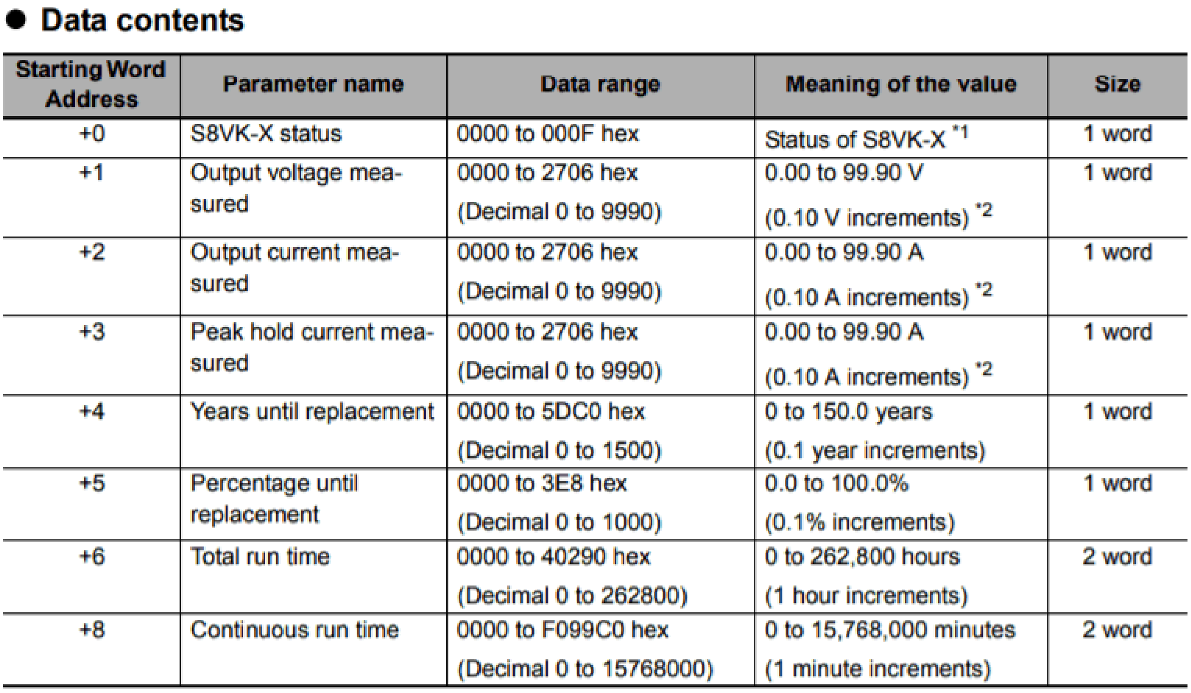
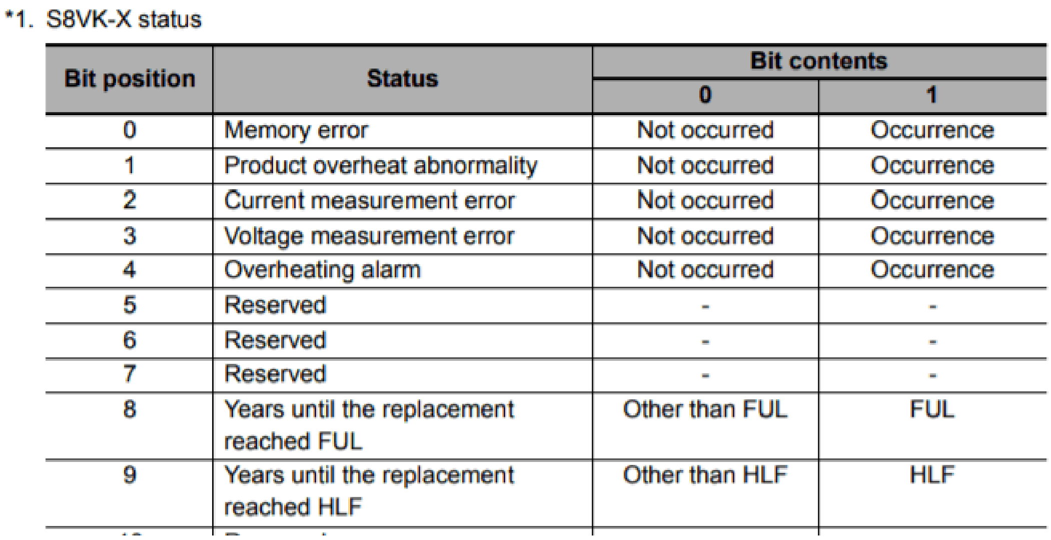
Description and example how to establish communication between S8VK-X and NX102 PLC

Tag links must be used to read those parameters (S8VK-X Comm. manual T213):



● **Data contents**

Starting Word Address	Parameter name	Data range	Meaning of the value	Size
+0	S8VK-X status	0000 to 000F hex	Status of S8VK-X [1]	1 word
+1	Output voltage measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 V (0.10 V increments) [2]	1 word
+2	Output current measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 A (0.10 A increments) [2]	1 word
+3	Peak hold current measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 A (0.10 A increments) [2]	1 word
+4	Years until replacement	0000 to 5DC0 hex (Decimal 0 to 1500)	0 to 150.0 years (0.1 year increments)	1 word
+5	Percentage until replacement	0000 to 3E8 hex (Decimal 0 to 1000)	0.0 to 100.0% (0.1% increments)	1 word
+6	Total run time	0000 to 40290 hex (Decimal 0 to 262800)	0 to 262,800 hours (1 hour increments)	2 word
+8	Continuous run time	0000 to F099C0 hex (Decimal 0 to 15768000)	0 to 15,768,000 minutes (1 minute increments)	2 word



*1. S8VK-X status

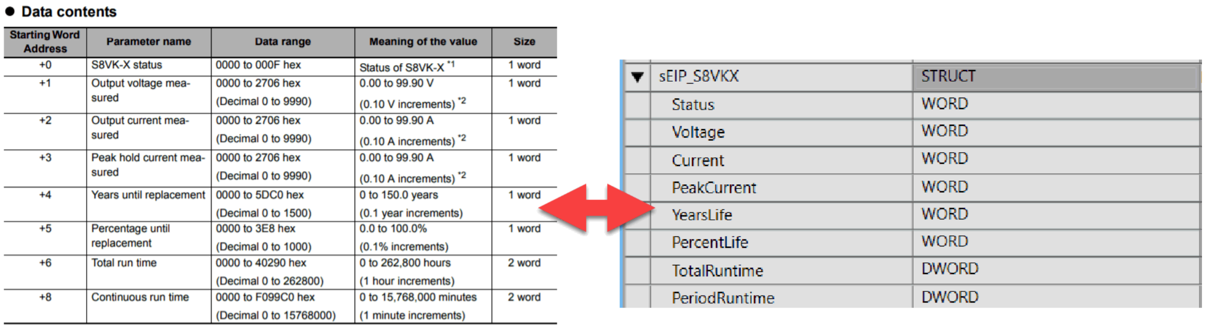
Bit position	Status	Bit contents	
		0	1
0	Memory error	Not occurred	Occurrence
1	Product overheat abnormality	Not occurred	Occurrence
2	Current measurement error	Not occurred	Occurrence
3	Voltage measurement error	Not occurred	Occurrence
4	Overheating alarm	Not occurred	Occurrence
5	Reserved	-	-
6	Reserved	-	-
7	Reserved	-	-
8	Years until the replacement reached FUL	Other than FUL	FUL
9	Years until the replacement reached HLF	Other than HLF	HLF

However, these data should be converted in some way to variables.

The example attached collects the raw data above in variables , and dumps to a customized structure. It is done in the function toDCsourceStruct

How it works:

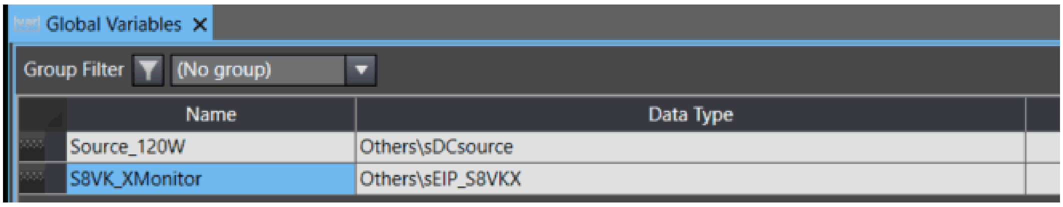
(1) First, we map a tag Link according the S8VK-X data map:



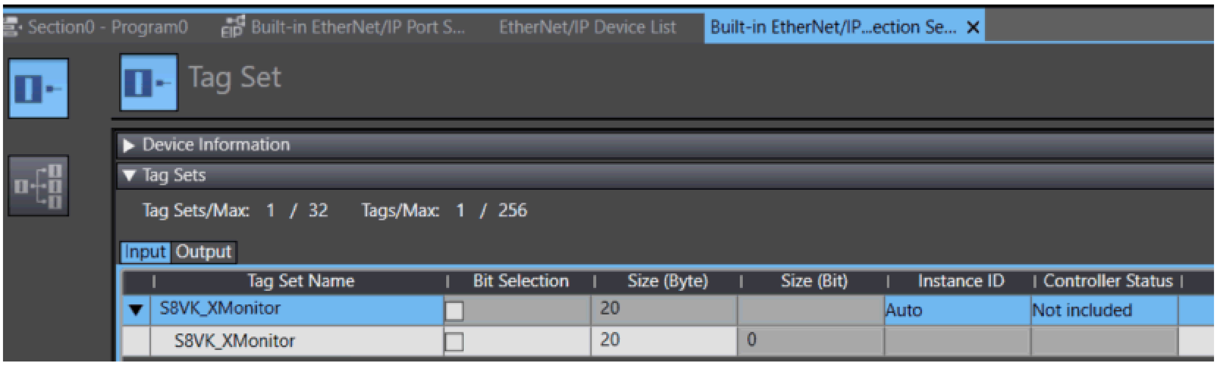
● **Data contents**

Starting Word Address	Parameter name	Data range	Meaning of the value	Size
+0	S8VK-X status	0000 to 000F hex	Status of S8VK-X [1]	1 word
+1	Output voltage measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 V (0.10 V increments) [2]	1 word
+2	Output current measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 A (0.10 A increments) [2]	1 word
+3	Peak hold current measured	0000 to 2706 hex (Decimal 0 to 9990)	0.00 to 99.90 A (0.10 A increments) [2]	1 word
+4	Years until replacement	0000 to 5DC0 hex (Decimal 0 to 1500)	0 to 150.0 years (0.1 year increments)	1 word
+5	Percentage until replacement	0000 to 3E8 hex (Decimal 0 to 1000)	0.0 to 100.0% (0.1% increments)	1 word
+6	Total run time	0000 to 40290 hex (Decimal 0 to 262800)	0 to 262,800 hours (1 hour increments)	2 word
+8	Continuous run time	0000 to F099C0 hex (Decimal 0 to 15768000)	0 to 15,768,000 minutes (1 minute increments)	2 word

▼ sEIP_S8VKX	STRUCT
Status	WORD
Voltage	WORD
Current	WORD
PeakCurrent	WORD
YearsLife	WORD
PercentLife	WORD
TotalRuntime	DWORD
PeriodRuntime	DWORD



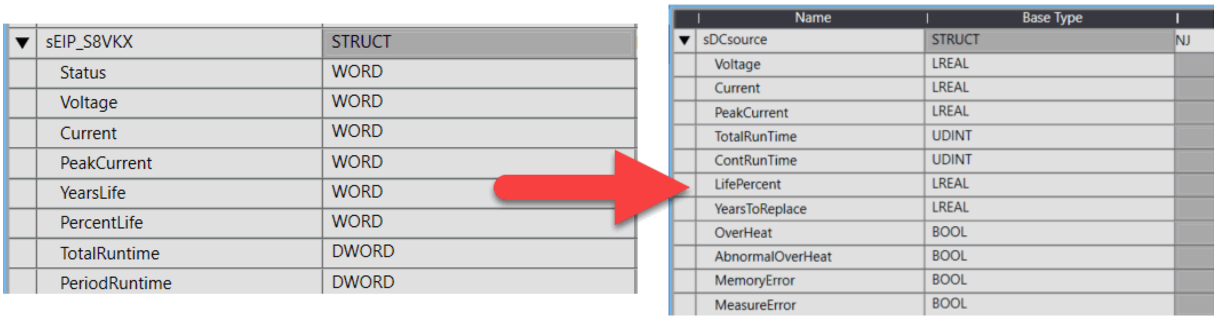
Name	Data Type
Source_120W	Others\sDCsource
S8VK_XMonitor	Others\sEIP_S8VKX



Tag Sets/Max: 1 / 32 Tags/Max: 1 / 256

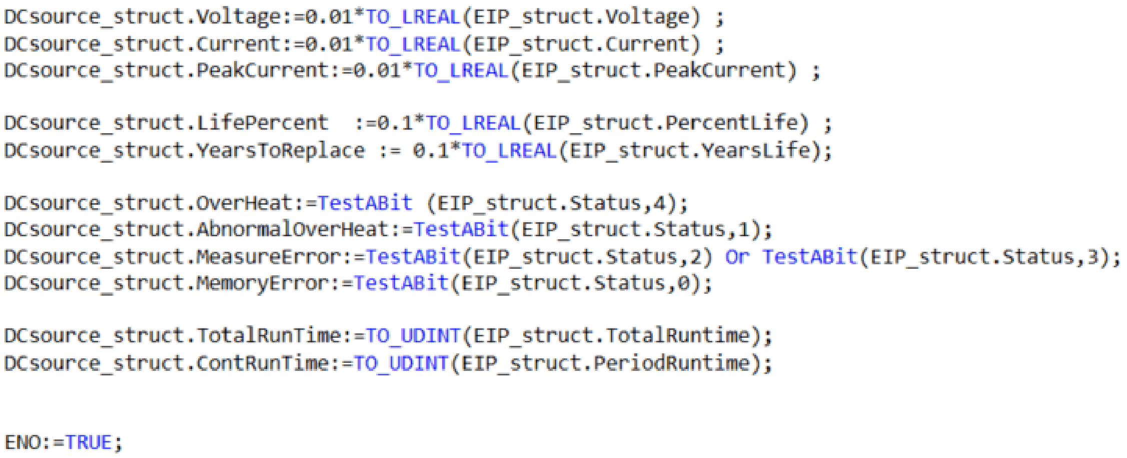
Tag Set Name	Bit Selection	Size (Byte)	Size (Bit)	Instance ID	Controller Status
▼ S8VK_XMonitor	□	20		Auto	Not included
S8VK_XMonitor	□	20	0		

(2) We create a conversion code from this raw EIP data to a custom structure, to do it more readable in structured variables:



▼ sEIP_S8VKX	STRUCT
Status	WORD
Voltage	WORD
Current	WORD
PeakCurrent	WORD
YearsLife	WORD
PercentLife	WORD
TotalRuntime	DWORD
PeriodRuntime	DWORD

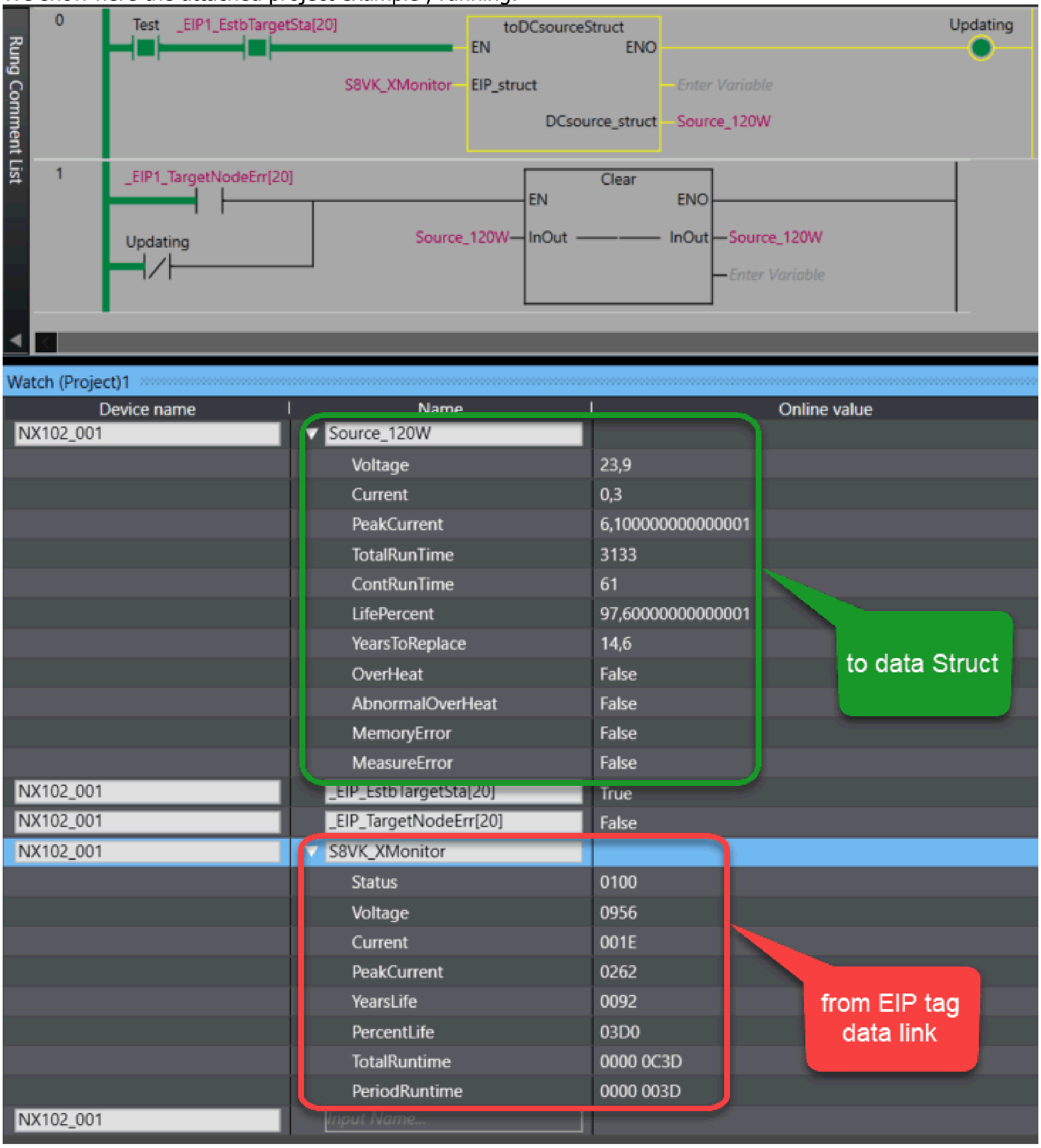
Name	Base Type	
▼ sDCsource	STRUCT	NJ
Voltage	LREAL	
Current	LREAL	
PeakCurrent	LREAL	
TotalRunTime	UDINT	
ContRunTime	UDINT	
LifePercent	LREAL	
YearsToReplace	LREAL	
OverHeat	BOOL	
AbnormalOverHeat	BOOL	
MemoryError	BOOL	
MeasureError	BOOL	



```
DCsource_struct.Voltage:=0.01*TO_LREAL(EIP_struct.Voltage) ;
DCsource_struct.Current:=0.01*TO_LREAL(EIP_struct.Current) ;
DCsource_struct.PeakCurrent:=0.01*TO_LREAL(EIP_struct.PeakCurrent) ;

DCsource_struct.LifePercent  :=0.1*TO_LREAL(EIP_struct.PercentLife) ;
DCsource_struct.YearsToReplace := 0.1*TO_LREAL(EIP_struct.YearsLife);

DCsource_struct.OverHeat:=TestABit (EIP_struct.Status,4);
DCsource_struct.AbnormalOverHeat:=TestABit(EIP_struct.Status,1);
DCsource_struct.MeasureError:=TestABit(EIP_struct.Status,2) Or TestABit(EIP_struct.Status,3);
DCsource_struct.MemoryError:=TestABit(EIP_struct.Status,0);

DCsource_struct.TotalRunTime:=TO_UDINT(EIP_struct.TotalRuntime);
DCsource_struct.ContRunTime:=TO_UDINT(EIP_struct.PeriodRuntime);


ENO:=TRUE;
```

(3) The data is converted to variables , inside the structure sDCsource.

We show here the attached project example , running:



Device name	Name	Online value
NX102_001	▼ Source_120W	
	Voltage	23,9
	Current	0,3
	PeakCurrent	6,100000000000001
	TotalRunTime	3133
	ContRunTime	61
	LifePercent	97,60000000000001
	YearsToReplace	14,6
	OverHeat	False
	AbnormalOverHeat	False
	MemoryError	False
	MeasureError	False
NX102_001	_EIP_EstbTargetSta[20]	True
NX102_001	_EIP_TargetNodeErr[20]	False
NX102_001	▼ S8VK_XMonitor	
	Status	0100
	Voltage	0956
	Current	001E
	PeakCurrent	0262
	YearsLife	0092
	PercentLife	03D0
	TotalRuntime	0000 0C3D
	PeriodRuntime	0000 003D
NX102_001	Input Name...	

to data Struct

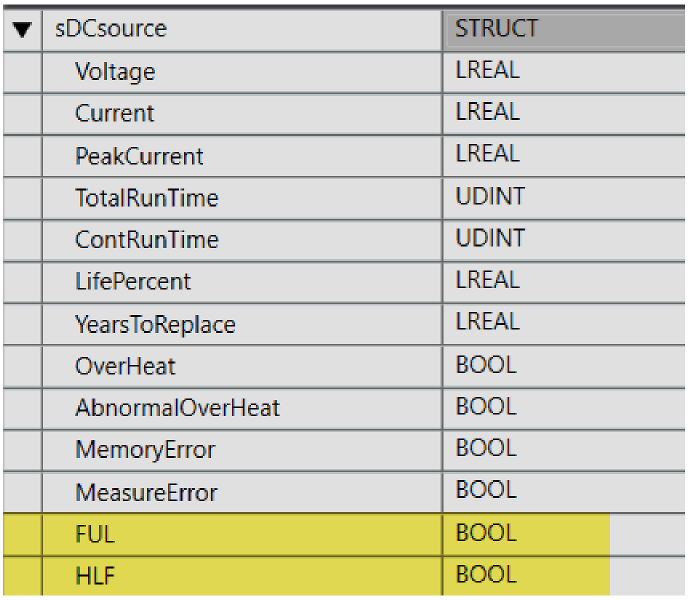
from EIP tag data link

Sysmac programming example

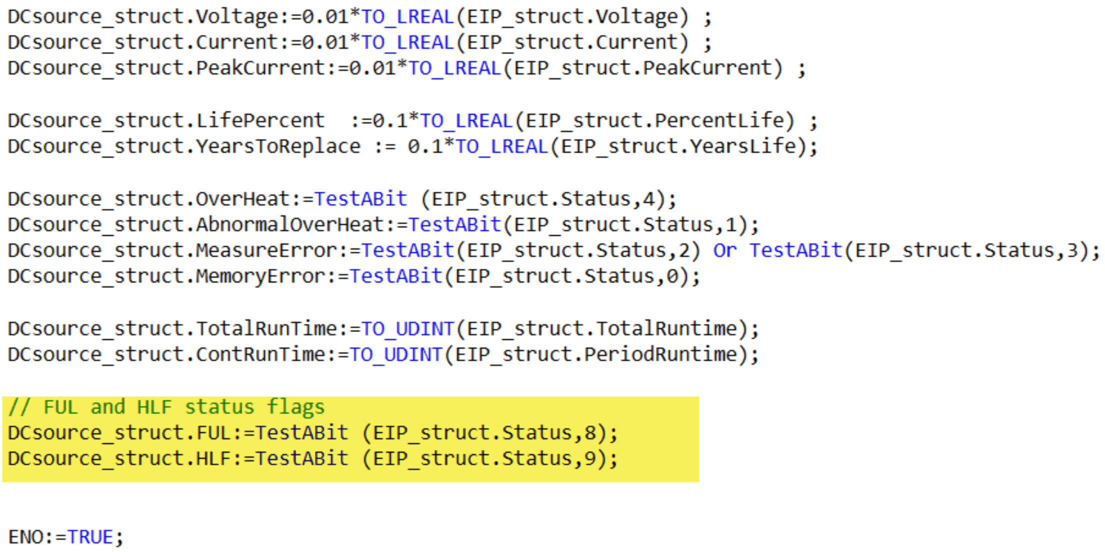
S8VK_Example_2.smc2

It includes the flag status for HLF and FUL bits, from the S8VK-X Status word.

Two lines to the function and the structure have been added as well.



▼	sDCsource	STRUCT
	Voltage	LREAL
	Current	LREAL
	PeakCurrent	LREAL
	TotalRunTime	UDINT
	ContRunTime	UDINT
	LifePercent	LREAL
	YearsToReplace	LREAL
	OverHeat	BOOL
	AbnormalOverHeat	BOOL
	MemoryError	BOOL
	MeasureError	BOOL
	FUL	BOOL
	HLF	BOOL



```
DCsource_struct.Voltage:=0.01*TO_LREAL(EIP_struct.Voltage) ;
DCsource_struct.Current:=0.01*TO_LREAL(EIP_struct.Current) ;
DCsource_struct.PeakCurrent:=0.01*TO_LREAL(EIP_struct.PeakCurrent) ;

DCsource_struct.LifePercent  :=0.1*TO_LREAL(EIP_struct.PercentLife) ;
DCsource_struct.YearsToReplace := 0.1*TO_LREAL(EIP_struct.YearsLife);

DCsource_struct.OverHeat:=TestABit (EIP_struct.Status,4);
DCsource_struct.AbnormalOverHeat:=TestABit(EIP_struct.Status,1);
DCsource_struct.MeasureError:=TestABit(EIP_struct.Status,2) Or TestABit(EIP_struct.Status,3);
DCsource_struct.MemoryError:=TestABit(EIP_struct.Status,0);

DCsource_struct.TotalRunTime:=TO_UDINT(EIP_struct.TotalRuntime);
DCsource_struct.ContRunTime:=TO_UDINT(EIP_struct.PeriodRuntime);

// FUL and HLF status flags
DCsource_struct.FUL:=TestABit (EIP_struct.Status,8);
DCsource_struct.HLF:=TestABit (EIP_struct.Status,9);


ENO:=TRUE;
```