BEDROCK IDE STRUCTERED TEXT REFERENCE SHEET

Data Types

Data Type	Lower Limit	Upper Limit	Memory
BOOL	FALSE	TRUE	1 bit
BYTE	0	255	8 bit
WORD	0	65535	16 bit
DWORD	0	4294967295	32 bit
LWORD	0	2^{64}-1	64 bit
SINT	-128	127	8 bit
USINT	0	255	8 bit
INT	-32768	32767	16 bit
UINT	0	65535	16 bit
DINT	-2147483648	2147483647	32 bit
UDINT	0	4294967295	32 bit
LINT	-263	2^{63}-1	64 bit
ULINT	0	2^{64}-1	64 bit
TIME	0	4294967295	32 bit
(TOD) TIME_OF_DAY	0 (00:00:00:000)	4294967295 (11:59:59 PM:999)	32 bit
DATE	0 (01.01.1970)	4294967295 (2106-02-07)	32 bit
(DT) DATE_AND_TIME	0 (1970-01-01, 00:00:00)	4294967295 (2106-02-07, 06:28:15)	32 bit

Standard Library Functions

Scope	Name	Type	Comment
TON (On-Delay Timer) *TOF*, TP*, LTON**, LTOF**, LTP**			
Input	IN	BOOL	starts timer with rising edge
	PT	TIME	time to pass, before Q is set
Output	Q	BOOL	gets TRUE when ET > PT
	ET	TIME	elapsed time since IN := TRUE
R_TRIG (Rising Edge Trigger) *F_TRIG**			
Input	CLK	BOOL	Signal to detect
Output	Q	BOOL	rising edge at signal detected
CTU (Count UP) *CTD*, CTUD**			
Input	CU	BOOL	Count Up
	RESET	BOOL	Reset Counter to 0
	PV	WORD	Counter Limit
Output	Q	BOOL	Counter reached the Limit
	CV	WORD	Current Counter Value

* = Additional Similar Functions

**= Standard64 Library needed

Define Variables

```
Var : DataType := Initial_Value;
arrVar : Array[X..Y] Of DataType;
strVar : WSTRING (Length); //Default Length = 80
funcVar : Function;
```

Assignment of Variables

```
Var := Expression;
arrVar[X] := Expression;
strVar := "String Text";
funcVar(Input1 := Var, Input2 := Var);
Var := funcVar.Output;
```

Expressions

Equal	=	Not Equal	<>
Greater Then	>	Greater or Equal	>=
Less Then	<	Less or Equal	<=
And	AND	Or	OR
Negation	NOT		

Conversions

```
VarDataType1 := DataType2_TO_Datatype1(VarDataType2);
```

EX.

```
iVar : INT;
rVar : REAL;
rVar := INT_TO_REAL(iVar);
```

Constructs

IF…THEN

```
IF Boolean_Expression1 THEN
        …
ELSIF Boolean_Expression2 THEN
        …
ELSE
        …
END_IF
```

CASE…OF

```
CASE Numeric_expression OF
1:
        …
2, 3:
        …
4..6:
        …
7, 9..11:
        …
ELSE
        …
END_CASE
```

FOR…DO

```
FOR count := Int_Val TO Fin_Val BY Incr DO
        …
        IF Boolean_Expression THEN
                EXIT;
        END_IF
END_FOR
```

WHILE…DO

```
WHILE Boolean_Expression DO
        …
        IF Boolean_Expression THEN
                EXIT;
        END_IF
END_WHILE
```

REPEAT…UNTIL

```
REPEAT
        …
        IF Boolean_Expression THEN
                EXIT;
        END_IF
UNTIL Boolean_Expression
END_REPEAT
```