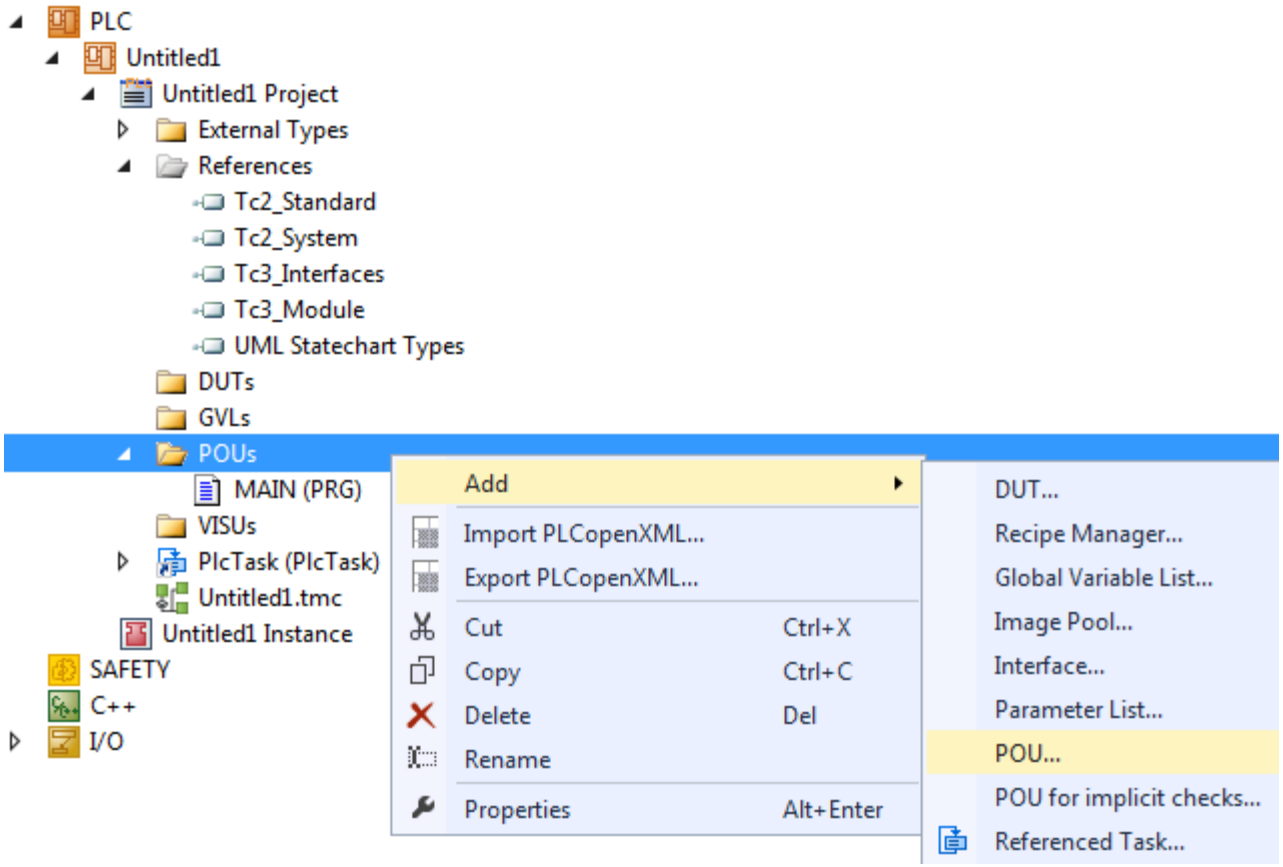
BECKHOFF


BECKHOFF
BECKHOFF
EK1100
TwinCAT
The Windows Control and Automation Technology
Version 3
BECKHOFF
BECKHOFF

Adding a new Function Block

BECKHOFF

- To add a Function Block, ‘Right-Click’ on the ‘POUs’ folder
  - Then Select ‘Add’ → ‘POU’



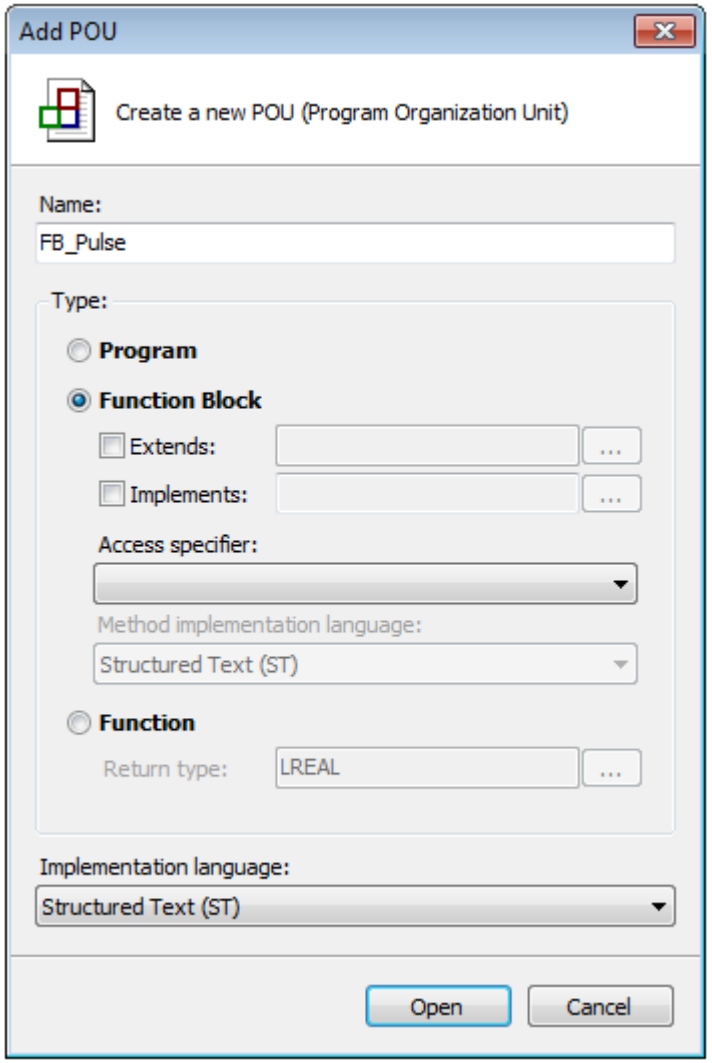
Adding a new Function Block

BECKHOFF

In **Add POU** Dialog window

- Change **Name** to ‘FB_Pulse’
- Set **Type** Radio Button to ‘Function Block’
- Set the **Implementation Language** to ‘Structured Text’
- Click ‘Open’

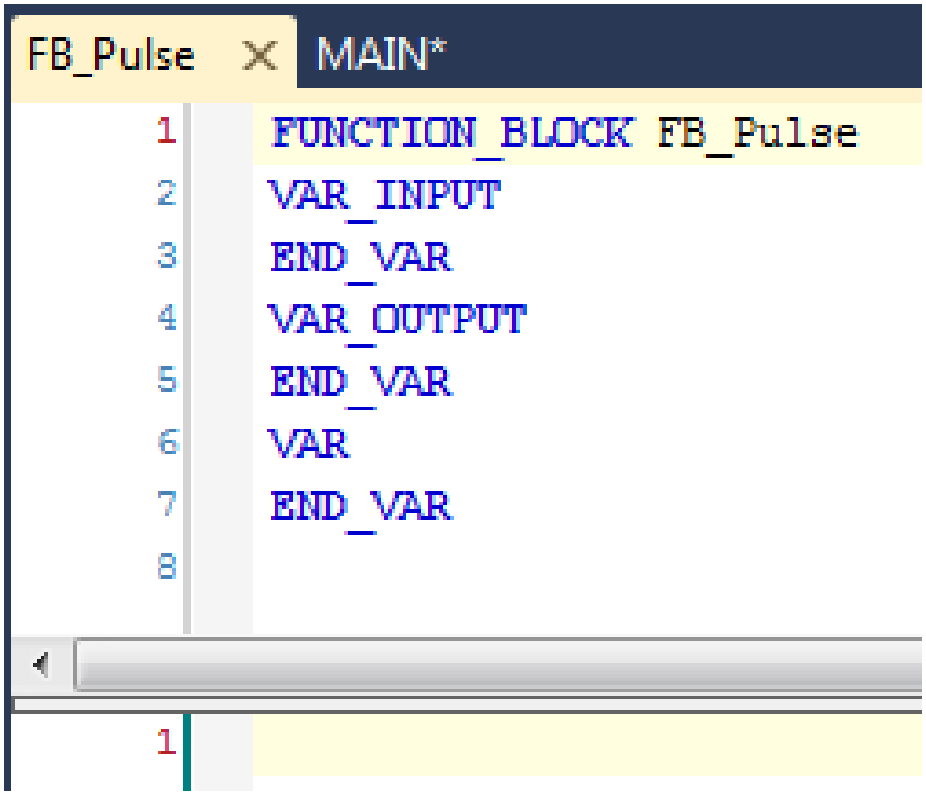
Note: Extends, Implements, and Access Specifier are used with OOP and should be left empty



Adding a new Function Block

BECKHOFF

You should now see the following



Parts of a Function Block

BECKHOFF

- Declaration
- Code
- Use

Parts of a Function Block : Declaration

BECKHOFF

Declaration of a Function Block contains 4 parts

- Name of the Function Block
- Variables passed into the Function Block
- Variables passed out of the Function Block
- Variables that are internal to the Function Block

The Body/Code of the Function Block

- Empty by default

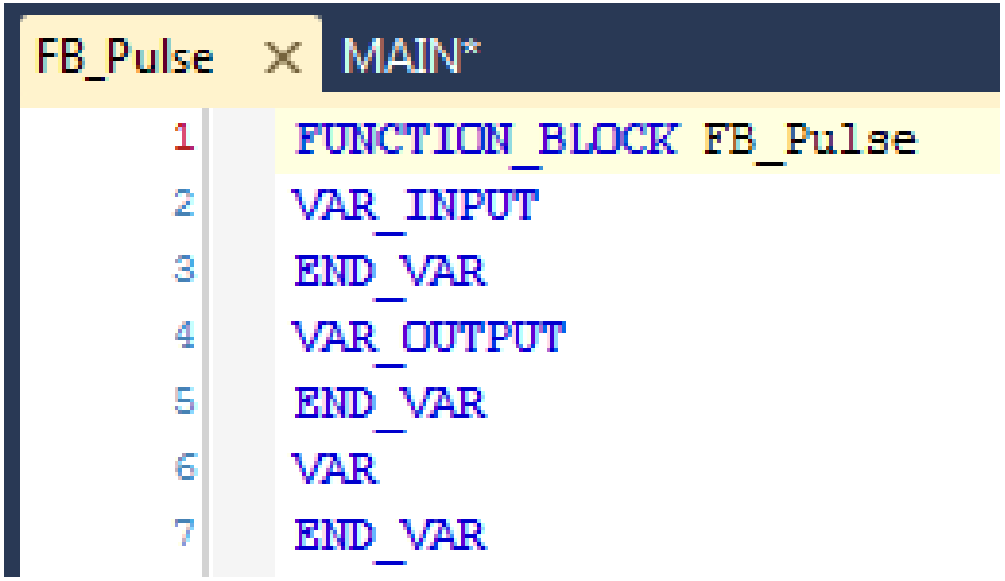
Parts of a Function Block : Declaration

BECKHOFF

Name of the Function Block

- Beckhoff coding convention recommends
  - Function Block prefix **FB_**

- IEC variable name rules apply to naming of Function Blocks



Parts of a Function Block : Declaration

BECKHOFF

- The Variables to be passed into the Function Block
- Below the Enable, Time On, and Time Off values are being passed into the Function Block

FB_Pulse* MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8  END_VAR
9  VAR
10 END_VAR
11
```

Parts of a Function Block : Declaration

BECKHOFF

- The Variables to be passed out of the Function Block
- Below the Output variable ‘bPulse’ has been added

FB_Pulse* | MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8      bPulse : BOOL;
9  END_VAR
10 VAR
11 END_VAR
12
```

Parts of a Function Block : Declaration

BECKHOFF

- The Variables that are internal to the Function Block
- Below the two timers to be used have been instantiated
- fbTON is of type TON
- fbTOF is of type TOF

FB_Pulse* | MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8      bPulse : BOOL;
9  END_VAR
10 VAR
11     fbTON : TON;
12     fbTOF : TOF;
13 END_VAR
14
```

Parts of a Function Block: Code

BECKHOFF

- Code of the Function Block
- Timers are called with their instance name
- := symbol signifies that a value of the variable is being passed into the FB and the => symbol signifies that a value of the variable is being passed out of the FB

FB_Pulse * | MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8      bPulse : BOOL;
9  END_VAR
10 VAR
11     fbTON : TON;
12     fbTOF : TOF;
13 END_VAR
```

```
1  fbTON(IN:= bEnable AND NOT fbTOF.Q, PT:= tTimeOff, Q=> , ET=> );
2  fbTOF(IN:= fbTON.Q, PT:= tTimeOn, Q=> bPulse , ET=> );
3
```

Parts of a Function Block: Code

BECKHOFF

- IN of fbTON is TRUE if bEnable is TRUE and fbTOF.Q is FALSE
- What is fbTOF.Q?
- Any the . symbol is used it signifies that the variable on the right exists inside of the variable on the left.
- Q is an output of TOF, therefore calling the instance name fbTOF followed by . will allow access to the variables that are declared inside of fbTOF

FB_Pulse * | MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8      bPulse : BOOL;
9  END_VAR
10 VAR
11     fbTON : TON;
12     fbTOF : TOF;
13 END_VAR
```

```
1  fbTON(IN:= bEnable AND NOT fbTOF.Q, PT:= tTimeOff, Q=> , ET=> );
2  fbTOF(IN:= fbTON.Q, PT:= tTimeOn, Q=> bPulse , ET=> );
3
```

Parts of a Function Block: Code

BECKHOFF

- fbTON.Q is passed into fbTOF, this will cause the two timers to toggle based on the values of tTimeOff and tTimeOn
- Finally the output of fbTOF is passed to bPulse. bPulse is the output of FB_Pulse
- Produces the same result
  bPulse := fbTOF.Q;

FB_Pulse * | MAIN*

```
1  FUNCTION_BLOCK FB_Pulse
2  VAR_INPUT
3      bEnable : BOOL;
4      tTimeOn : TIME;
5      tTimeOff : TIME;
6  END_VAR
7  VAR_OUTPUT
8      bPulse : BOOL;
9  END_VAR
10 VAR
11     fbTON : TON;
12     fbTOF : TOF;
13 END_VAR
```

```
1  fbTON(IN:= bEnable AND NOT fbTOF.Q, PT:= tTimeOff, Q=> , ET=> );
2  fbTOF(IN:= fbTON.Q, PT:= tTimeOn, Q=> bPulse , ET=> );
3
```

Parts of a Function Block: Use

BECKHOFF

- fbPulse1 is of type FB_Pulse
- fbPulse1 is an instance of FB_Pulse
- bSwitch is passed into the bEnable input of fbPulse1
- tTimeOn & tTimeOff are assigned literal values in of TIME data type
- bPulse is passed out of fbPulse1 into bLight

FB_Pulse | MAIN*

```
1  PROGRAM MAIN
2  VAR
3      bSwitch : BOOL;
4      bLight : BOOL;
5      fbPulse1 : FB_Pulse;
6  END_VAR
7
```

```
1  fbPulse1(bEnable:= bSwitch, tTimeOn:= T#1S , tTimeOff:= T#2S , bPulse=> bLight );
2
```

Parts of a Function Block: Summary

BECKHOFF

- The Declaration of a Function Block contains 4 parts
  - The Name of the Function Block
  - The Variables to be passed into the Function Block
  - The Variables to be passed out of the Function Block
  - The Variables that are internal to the Function Block
- The Body/Code of the Function Block