

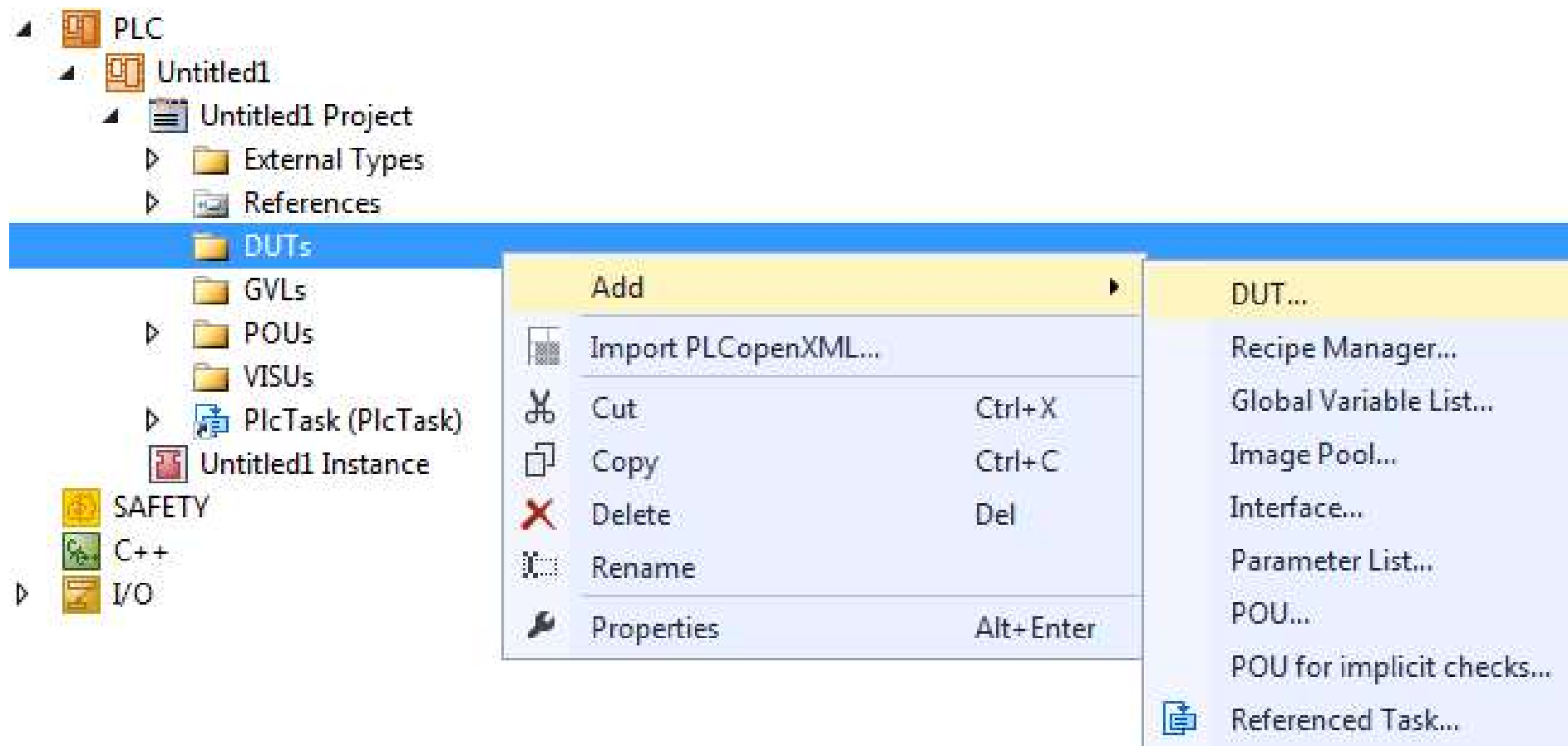
BECKHOFF



- Enumerations can be used to assign a variable name to a number
- Enumerations can be used in two different ways; with or without being instantiated
- If the Enumeration is not instantiated then the Enumeration works similar to a list of constants
- If the Enumeration is instantiated then the instance of the enumeration will hold the variable of the current value of the Enumeration

Enumerations

- To add an Enumeration, 'Right-Click' on the 'DUTs' folder
 - Then Select 'Add' -> 'DUT'



- Change the Name to E_Mode
- Select the Radio button for Enumeration
- Click the 'Open' button

Add DUT

Create a new data unit type

Name:
E_Mode

Type:

☐ **Structure**
☐ Extends: ...

☒ **Enumeration**

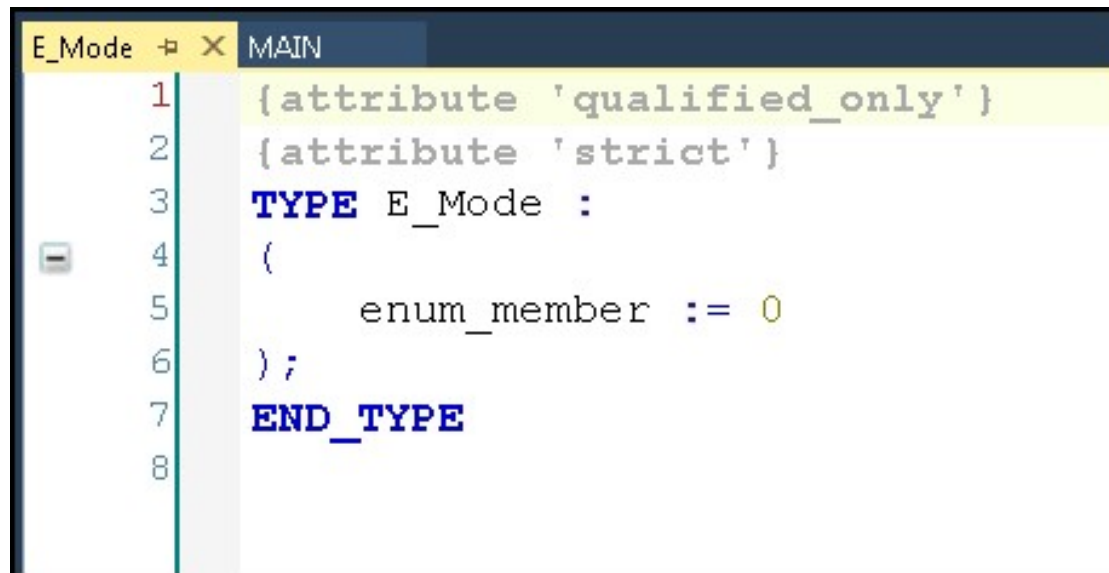
☐ **Alias**
Base type: >

☐ **Union**

Open Cancel

Enumerations

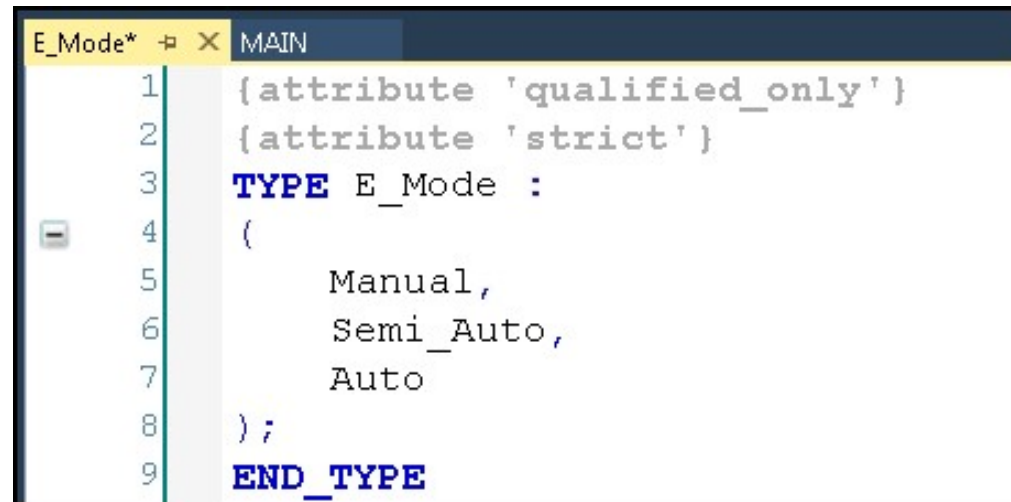
- The basic framework of an Enumeration looks like the following



```
1 {attribute 'qualified_only'}
2 {attribute 'strict'}
3 TYPE E_Mode :
4 (
5     enum_member := 0
6 );
7 END_TYPE
8
```

Enumerations

- When the Enumeration is defined the first variable in the list will be assigned a value of 0, the variables following will be assigned their values in ascending order
- Manual = 0
- Semi_Auto = 1
- Auto = 2



```
E_Mode*  ✕ MAIN
1 {attribute 'qualified_only'}
2 {attribute 'strict'}
3 TYPE E_Mode :
4 (
5     Manual,
6     Semi_Auto,
7     Auto
8 );
9 END_TYPE
```

Enumerations

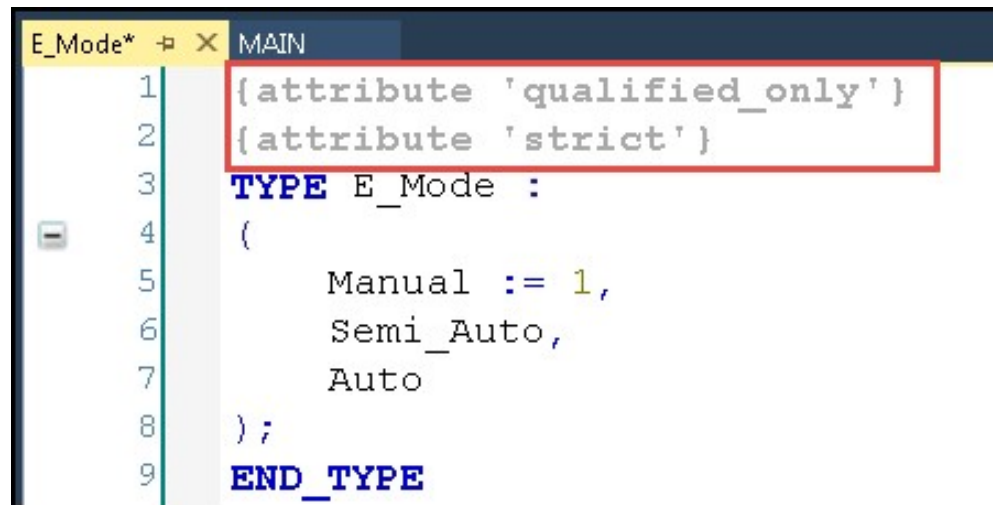
- If a variable in the list is explicitly assigned a value then the following variables will be incremented from this value
- Manual = 1
- Semi_Auto = 2
- Auto = 3
- Manual = 1
- Semi_Auto = 2
- Auto = 3
- Maintenance = 10
- Unknown = 11

```
E_Mode*  ➤ X MAIN
1 {attribute 'qualified_only'}
2 {attribute 'strict'}
3 TYPE E_Mode :
4 (
5     Manual := 1,
6     Semi_Auto,
7     Auto
8 );
9 END_TYPE
```

```
E_Mode*  ➤ X MAIN
1 {attribute 'qualified_only'}
2 {attribute 'strict'}
3 TYPE E_Mode :
4 (
5     Manual := 1,
6     Semi_Auto,
7     Auto,
8     Maintenance := 10,
9     Unknown
10 );
11 END_TYPE
```

Enumerations

- The gray texts included are called Pragmas
- The Pragmas influence the properties of the defined enumerations and affect how they are used within POU's
- If the programmer does not want the Pragmas, they can be commented out or simply deleted*

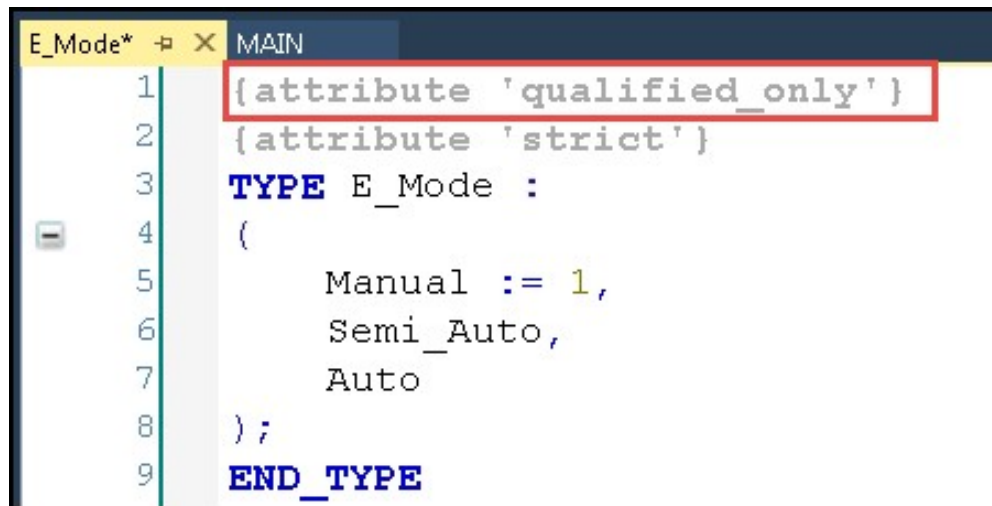


```
E_Mode*  MAIN
1  {attribute 'qualified_only'}
2  {attribute 'strict'}
3  TYPE E_Mode :
4  (
5      Manual := 1,
6      Semi_Auto,
7      Auto
8  );
9  END_TYPE
```

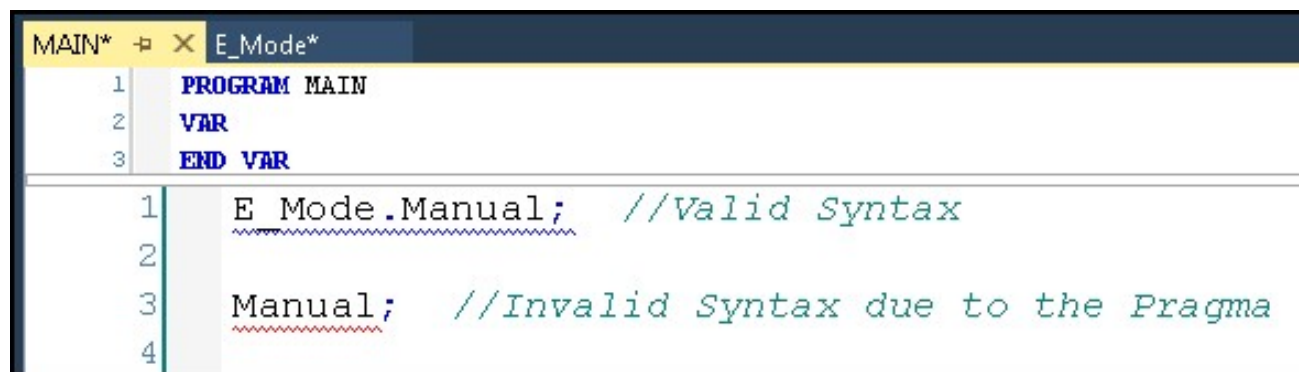
*The Pragmas are included by default and are recommended to add traceability; but not required

Enumerations

- The 'qualified_only' attribute Pragma requires the enumeration to be called by its Enumeration DUT name when being used within another POU



```
1 {attribute 'qualified_only'}  
2 {attribute 'strict'}  
3 TYPE E_Mode :  
4 (  
5     Manual := 1,  
6     Semi_Auto,  
7     Auto  
8 );  
9 END_TYPE
```



```
1 PROGRAM MAIN  
2 VAR  
3 END VAR  
  
1 E_Mode.Manual; //Valid Syntax  
2  
3 Manual; //Invalid Syntax due to the Pragma  
4
```

Enumerations

- The 'strict' attribute Pragma prevents alternate values from being assigned and any arithmetic operation with variables of the enumeration type

```
E_Mode*  MAIN
1 {attribute 'qualified_only'}
2 {attribute 'strict'}
3 TYPE E_Mode :
4 (
5     Manual := 1,
6     Semi_Auto,
7     Auto
8 );
9 END_TYPE
```

Error List

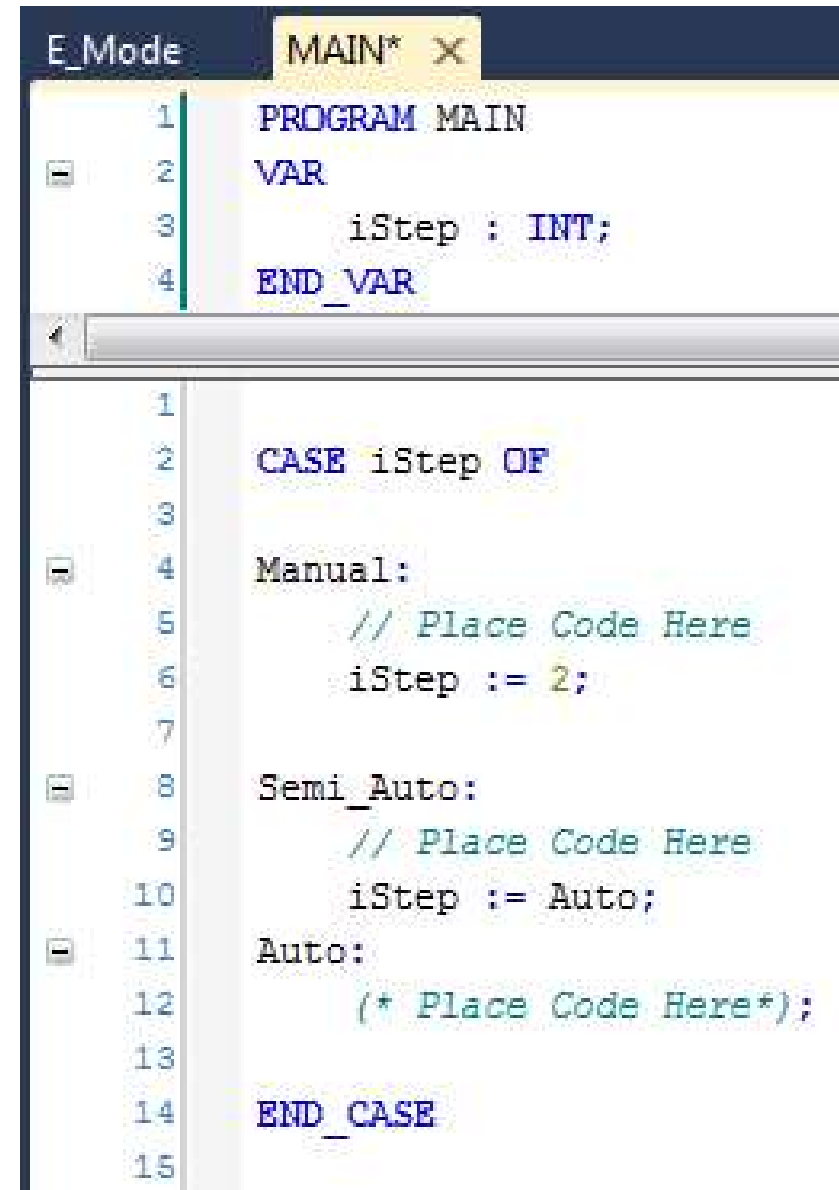
✕ 1 Error | ⚠ 4 Warnings | ⓘ 16 Messages | Clear

Description
✕ 18 '100' is not a valid value for strict ENUM type 'E_Mode'

```
MAIN  E_Mode
1 PROGRAM MAIN
2 VAR
3 END_VAR

1 E_Mode.Manual := 100; //Invalid Syntax
2
```

- Using Enumerations allows for easier understanding of the code when it is being read
- In this Case statement the Variables in the Enumeration are used to represent the number equivalent of iStep.
- As iStep changes in value the Case statement will change states
- iStep can be assigned a numerical value or an Enumeration variable
- The displayed online value of iStep1 will always be an INT value because iStep1 is declared as an INT



The screenshot shows the Beckhoff LAD editor interface. The top bar indicates the current mode is 'E_Mode' and the active project is 'MAIN*'. The editor displays a ladder logic program with the following code:

```
1 PROGRAM MAIN
2 VAR
3     iStep : INT;
4 END_VAR

5
6 CASE iStep OF
7
8     Manual:
9         // Place Code Here
10        iStep := 2;
11
12     Semi_Auto:
13         // Place Code Here
14        iStep := Auto;
15
16     Auto:
17         (* Place Code Here*);
18
19 END_CASE
```

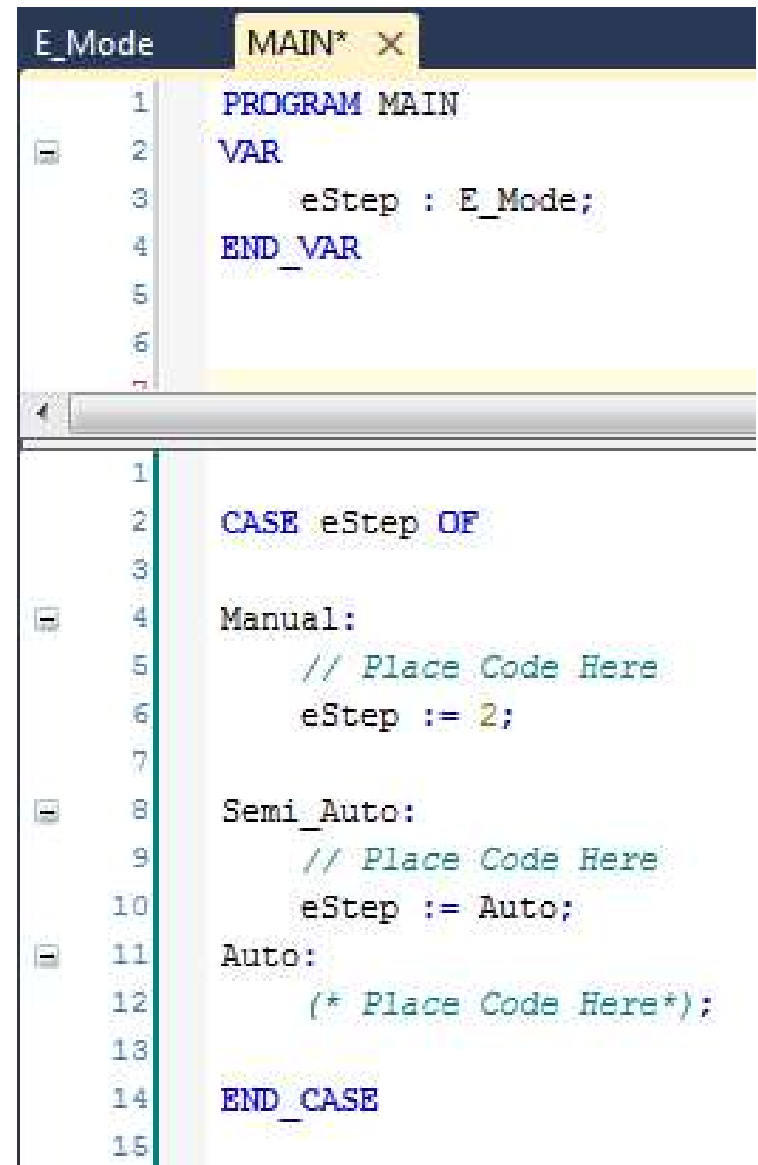
Enumerations

- The displayed online value of iStep will always be an INT value because iStep is declared as an INT

```
1  
2 CASE iStep 3 OF  
3  
4 Manual 1 :  
5     // Place Code Here  
6     iStep 3 := 2;  
7  
8 Semi_Auto 2 :  
9     // Place Code Here  
10    iStep 3 := Auto 3 ;  
11 Auto 3 :  
12     (* Place Code Here*);  
13  
14 END_CASE  
15
```

Enumerations

- If an instance of the Enumeration is declared then the instance of the Enumeration holds the variable name of the value of the Enumeration



The screenshot shows a Beckhoff LAD editor window with two tabs: 'E_Mode' and 'MAIN*'. The 'MAIN*' tab is active, displaying a ladder logic program. The program starts with a 'PROGRAM MAIN' statement, followed by a 'VAR' declaration for 'eStep' of type 'E_Mode', and an 'END_VAR' statement. Below this, a 'CASE' statement is shown, with three cases: 'Manual', 'Semi_Auto', and 'Auto'. Each case has a comment indicating where to place code. The 'Manual' case sets 'eStep' to 2. The 'Semi_Auto' case sets 'eStep' to 'Auto'. The 'Auto' case has a comment indicating where to place code. The program ends with an 'END_CASE' statement.

```
1 PROGRAM MAIN
2 VAR
3     eStep : E_Mode;
4 END_VAR
5
6
7
8 CASE eStep OF
9     Manual:
10         // Place Code Here
11         eStep := 2;
12
13     Semi_Auto:
14         // Place Code Here
15         eStep := Auto;
16
17     Auto:
18         (* Place Code Here*);
19
20 END_CASE
```

- If eStep is of type E_Mode then eStep holds either 'Manual', 'Semi_Auto', or 'Auto'

The screenshot shows the TwinCAT IDE interface. At the top, there's a tab labeled 'E_Mode' and another tab labeled 'MAIN [Online]*'. Below the tabs, the project name 'TwinCAT_Device.Untitled1.MAIN' is displayed. A table shows the variable 'eStep' of type 'E_MODE' with a value of 'Auto'. Below the table, a CASE statement is shown in the code editor, defining three states: Manual, Semi_Auto, and Auto. Each state has a corresponding code block for placement.

Expression	Type	Value	Prepared value
eStep	E_MODE	Auto	

```
1
2 CASE eStep Auto OF
3
4 Manual 1 :
5     // Place Code Here
6     eStep Auto := 2;
7
8 Semi_Auto 2 :
9     // Place Code Here
10    eStep Auto := Auto 3;
11 Auto 3 :
12    (* Place Code Here*);
13
14 END_CASE
15
```

Beckhoff Automation LLC

USA Headquarters
13130 Dakota Avenue
Savage, MN 55378
USA

Phone: 1 952 890 0000
Fax: 1 952 890 2888
E-Mail: beckhoff.usa@beckhoff.com
Web: www.beckhoffautomation.com

Beckhoff Automation GmbH & Co. KG

Global Headquarters
Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
Fax: +49 5246 963-198
E-Mail: info@beckhoff.com
Web: www.beckhoff.com

© Beckhoff Automation GmbH & Co. KG

All images are protected by copyright. The use and transfer to third parties is not permitted.

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this presentation may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

The information provided in this presentation contains merely general descriptions or characteristics of performance which in case of actual application do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressly agreed in the terms of contract.