Using variant pointer in TIA V15 and the new simplified ‘?= ‘ operator

Reference Link : https://www.linkedin.com/pulse/programming-s71500-using-tia-v15-small-tips-trix-part1-mattias-lindh-1/?trackingId=CAJ0slV9SzqBtIJommMeLg%3D%3D

Denilson Pegaia

Platinum Expert

Joined: 9/23/2005

Last visit: 3/3/2022

Posts: 4055

Rating:

★★★★☆ (1263)

Technical Forum EXPERT

Hi,

As from TIA V15 it is possible to use references in program blocks. "References" provides a new type of pointer. References are typed pointers that refer to a specific data type.

With this features a variable address could be stored in “references” variables (of type REF_TO_ + data type, for example, REF_TO_int for references to integer variables).

To transfer the address of one variable to a “reference” variable it is used “REF(variable name)”.

To access the value of a “pointered” variable by a “reference” variable it is used the “reference” variable + “^”.

The instruction “?=” could be used to convert Variant in “reference”.

It is possible to compare if a “reference” variable is equal (or different) to NULL to check if it contain a valid reference (i.e. a reference for the same data type).

Requirements:

STEP 7 >=15

CPU S7-1500 FW >= 2.5

Use in FC: In, Out, Temp, Return

Use in FB: Temp

Data types: basic data types (exception of bools), UDT, SDT.

The attached sample program exemplify this new feature (see FB1)

Attachment

References.zip (816 Downloads)

Variant pointer kullanmak için minimum gereksinimler yukarıda resimde var

What is an Variant pointer?

An Variant pointer is nothing else than a pointer that can point to a data area in your controller. Quite similar with the old any pointer but with a more symbolic approach. It can point on a datatype that you created yourself or any other predefined data type and even other blocks. You can easily check which datatype the any pointer points at by using the instruction: TypeOf. E.g.

```
// The Code
CASE TypeOf(#In_MyVariant) OF
    MyDataType01:;
        // Do Something;
    MyDataType02:;
        //Do Something else;
END_CASE; // Case my variant of.
```

An Example Calculating area of different objects.

We want to create a simple function that can calculate the area of different objects, in this example we use Circle, Rectangle and Triangle. First, we define our three different objects by means of creating three different data types:

DataType #01: TCircle: 2 reals one radius and one area variable:

TCircle

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Radius	Real	0.0	☑	☑	☑	☐	The Radius of the circle
2	Area	Real	0.0	☑	☑	☑	☐	Calculated area

DataType #02: TRectangle: 3 reals *one Height, Base and one Area.*

TRectangle

Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
Height	Real	0.0	☑	☑	☑	☐	Height of rectangle
Base	Real	0.0	☑	☑	☑	☐	Base of rectangle
Area	Real	0.0	☑	☑	☑	☐	Area of rectangle

DataType #03 Triangle, *Same data as above.*

TTriangle

Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
Base	Real	0.0	☑	☑	☑	☐	Base of triangle
Height	Real	0.0	☑	☑	☑	☐	Height of triangle
Area	Real	0.0	☑	☑	☑	☐	Calculated area

Now we want to create our function for calculating the area of any of the above objects. We create a new FC called *CalcArea* , We define parameters in head as shown below

variantPointerProject ▸ PLC_2 [CPU 1511-1 PN] ▸ Program blocks ▸ CalcArea [FC1]

CalcArea

	Name	Data type	Default value	Comment
1	Input			
2	<Add new>			
3	Output			
4	<Add new>			
5	InOut			
6	Object	Variant		
7	Temp			
8	TmpCircle	REF_TO "TCircle"		
9	TmpRectangle	REF_TO "TRectangle"		
10	TmpTriangle	REF_TO "TTriangle"		
11	Constant			
12	CF_PI	Real	3.141593	PI Value
13	<Add new>			
14	Return			
15	CalcArea	Real		

We define one in/Out Parameter That is defined as a variant to be able to point on any object from which we want to calculate the area. Our function will return the calculated area and update the object .area variable. We define 3 Temp variables as references to the defined objects.

REF_TO “TCircle” , REF_TO “TRectangle” , REF_TO “TTriangle”
Defined as a reference to a circle. (Note the REF_TO in front of the declaration, this indicates that this is to be a reference to a variable.

Operators are used in function

1. `?=` operator used for assign reference to a object
2. `^` operator used for accessing elements of a reference

```
1
2
3  CASE TypeOf(#Object) OF
4      TCircle:
5          #TmpCircle ?= #Object;
6          #CalcArea := #TmpCircle^.Area := #TmpCircle^.Radius ** 2 * #CF_PI;//use ** operator for power of a number
7      TRectangle:
8          #TmpRectangle ?= #Object;
9          #CalcArea := #TmpRectangle^.Area := #TmpRectangle^.Height * #TmpRectangle^.Base;
10     TTriangle:
11         #TmpTriangle ?= #Object;
12         #CalcArea := #TmpTriangle^.Area := (#TmpTriangle^.Height * #TmpTriangle^.Base) / 2.0;
13
14     ELSE
15         ;
16 END_CASE;
17
```