DIAView Industrial Configuration Software Script Usage Specification

Abstract

DIAView industrial configuration software script contains two parts: event script and background script. The event script is mainly about triggering by mouse including clicking and releasing left button, clicking and releasing right button, entering, leaving, rolling, moving, pressing and window operation event (loaded, timetricked, closed); the background script contains condition script and time script.

Key words

DIAView Industrial Configuration Software Script Event Script Background Script

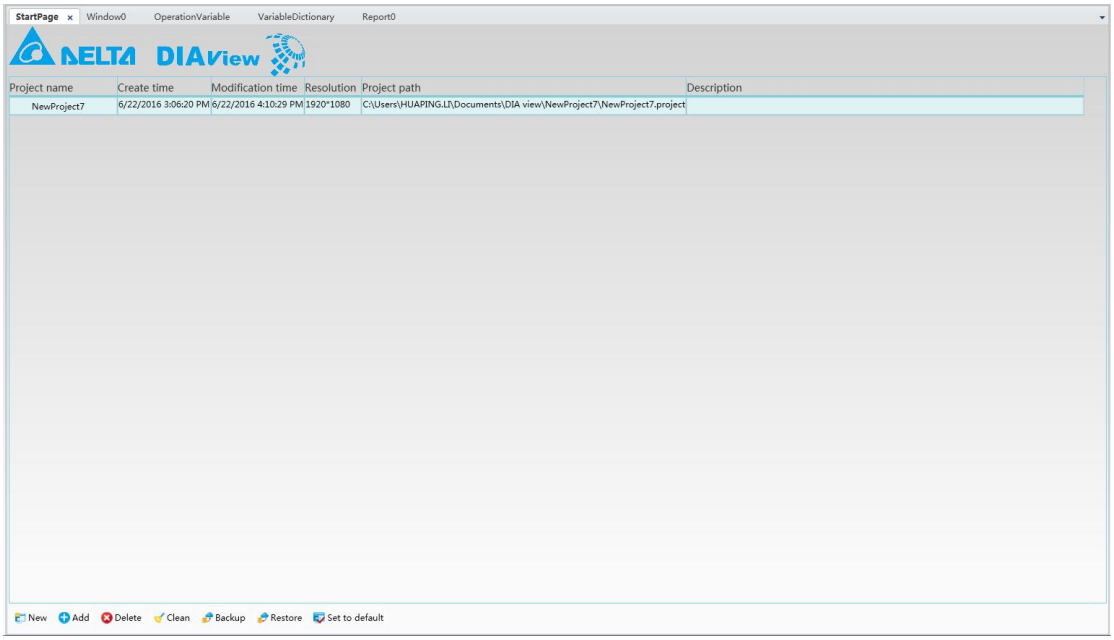
DIAView configuration requirements

configuration	minimum requirements
CPU	Frequency: 2.0GHz
Memory	2GB
Hard Disk	20GB
Monitor	Resolution 1024 x 768
Operation System	Windows XP SP3、Windows 7 Pro、Windows 8 Pro
	Bit:32/64 bit
	Language: Simplified Chinese、Traditional Chinese、English
System Rights	Windows User must have administrator rights
Operation Platform	Install Microsoft .NET Framework4.0 at least

Operation Steps

1. Event script operation

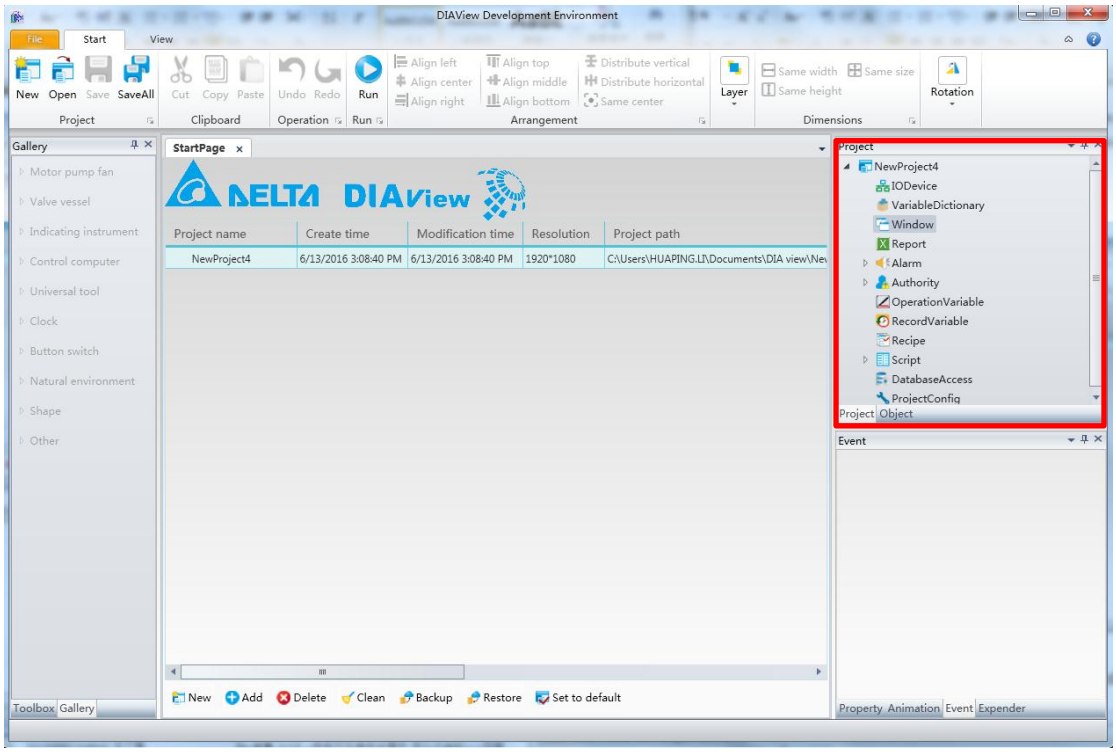
1.1 Start DIAView development environment, enter the main interface of the software as shown in the figure:



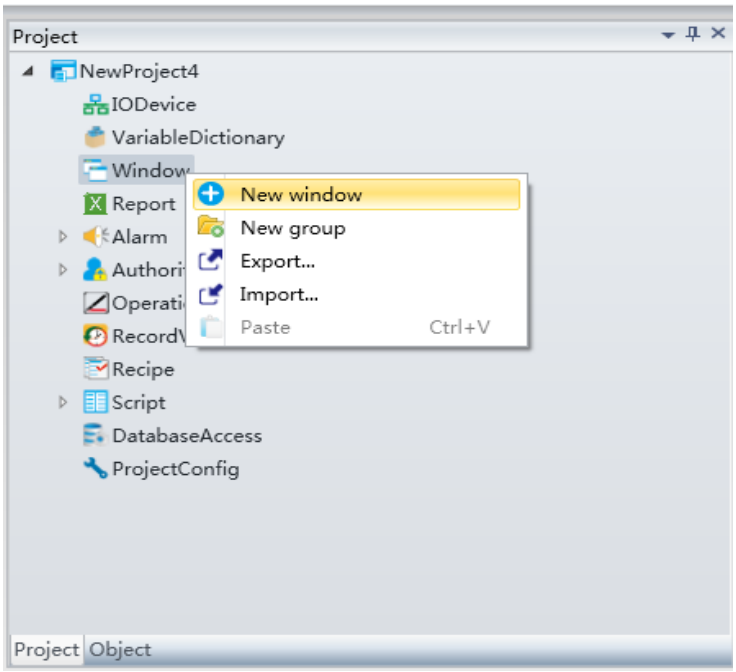
The components of DIAView development environment

1) Menu bar : provide the basic function in the development
2) Tool bar: the shortcut button of operation
3) Tool box: provide the common basic graphic and controls
4) Gallery ： provide the common graphic element and the custom graphic is allowed
5) Board workspace : the area of project management and the area of drawing and editing picture
6) Project window: display all the component of a project, provide the access of all the operation configuration
7) Object window: display all the objects in the window
8) Property window: display the property of objects, provide the access of modifying the property
9) Animation window：entrance of object configuration animation
10) Event window: entrance of object configuration event

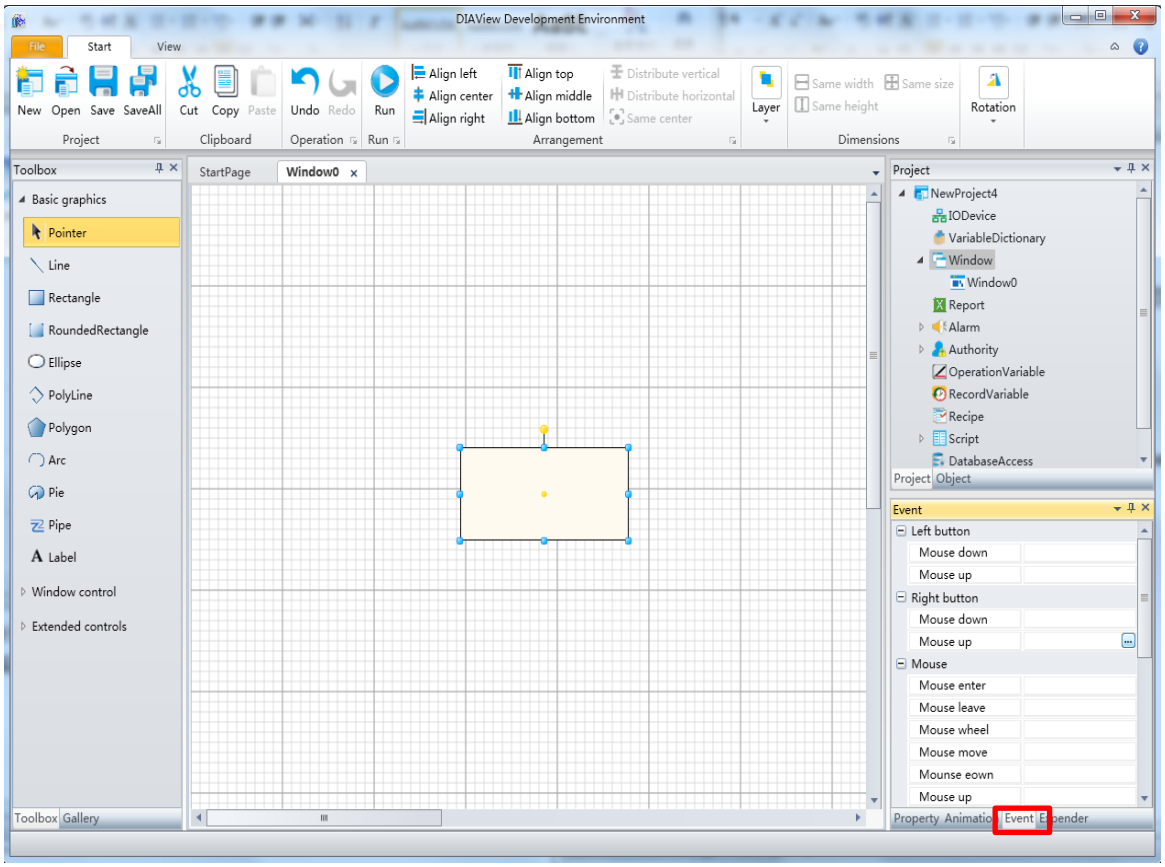
1.2 Create a new project as shown in the figure



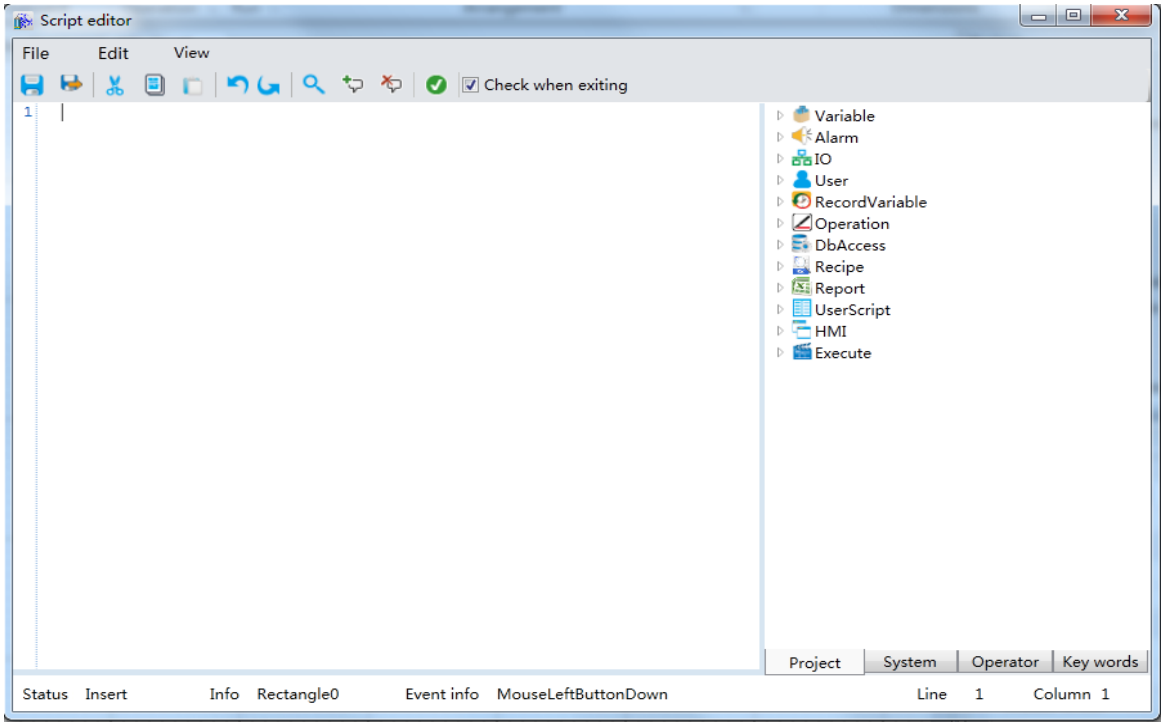
1.3 Create a new window, right-click on “Window”, then click “New project” in the right-click menu as shown in the figure:



1.4 Open the window and draw the basic graphic, open the event window as shown in the figure:

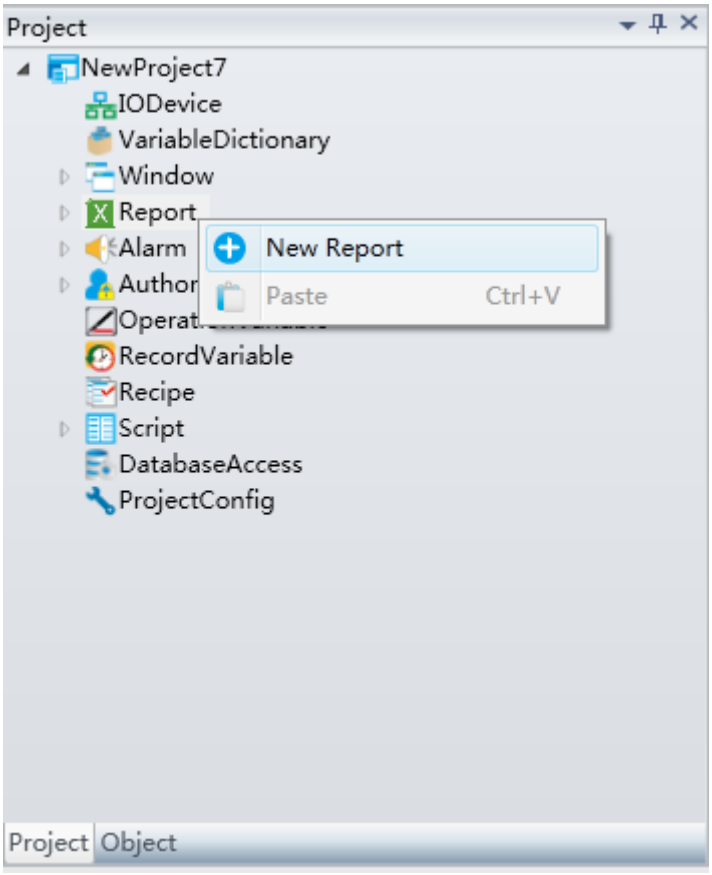


1.5 Take left-button-down event as an example, open script editor, input script(the script of DIAView is Visual Basic Script, the script editor will check the syntax errors automatically)as shown in the figure:

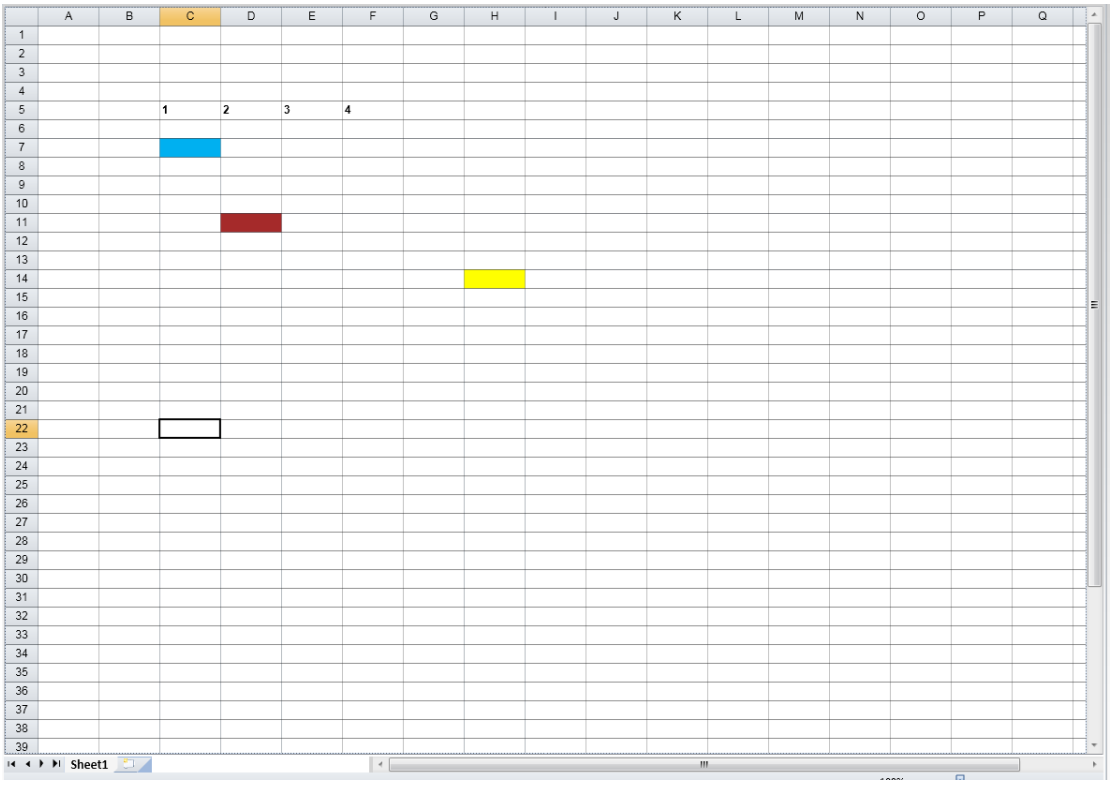


1.6 Create a new report, right-click on “Report”, then click “New report” in the

right-click menu as shown in the figure:



Open the report and perform various operations

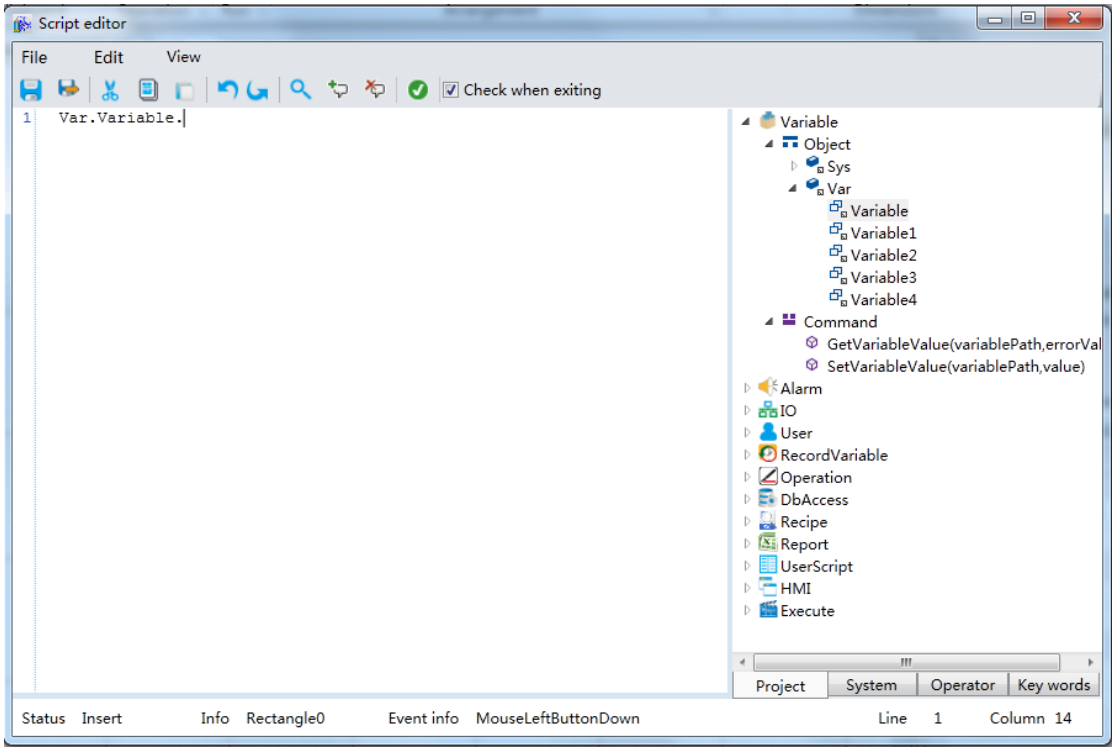


2. The script in the script editor mainly contains two parts, property

script and command script:

2.1 Script usage method for variable:

Click the variable on the right side of script editor, choose the corresponding variable and then edit the property of the corresponding variable in the script editor as shown in the figure:



The specific meaning of the variable property script is as follows:

Name	Type	Description
ValueAsDouble	Real	Variable value, double type
ValueAndUnit	String	Gets the variable values, and adds the engineering unit
DecimalPlaces	Integer	The number of decimal digits is 0~15
DeadBand	Real	Dead band,the minimum variation range of variable
InitValue	Real	Variable initial value
MinValue	Real	Variable minimum
MaxValue	Real	Maximum variable

Set the value of variable： Var.Variable.Value=True (type: discrete)
Var.Variable.Value=5 (type: analog)
Var.Variable.Value=”DIAView”(type: string)
Get the label of variable： Var.Variable.UniqueIdentifier(read only, cannot be set)
Get the name of variable： Var.Variable.Name(read only, cannot be set)
Get or set the description of variable：Var.Variable.Description=” analog variable”
Get the initial value of variable： Var.Variable.InitValue
Get the value of variable： Var.Variable.ValueAsBool(type: discrete)
Var.Variable1.ValueAsDouble(type: analog)

Var. Variable2.ValueAsString(type: string)

Get the engineering unit of variable：Var.Variable1.EngineeringUnit

Set whether to save the current value when exiting：

Var.Variable.IsInitValueSaved=True(True presents Save, False presents Not)

Set the dead band of variable：Var.Variable1.DeadBand=5(type: analog)

Set the maximum and minimum value of variable：Var.Variable1.MaxValue=1000,

Var.Variable1.MinValue=-1000(type: analog)

Set the decimal digits of variable：

Var.variable1.DecimalPlaces=2(type:analog,range:0~15)

Converted to string in specified format：Var.Variable1.ToString("C"), result: ￥0.0

The specific meaning of conversion format: C: converted into currency

E: converted into scientific notation

e: converted into scientific notation

F: fixed point, like 25.ToString("F2"), result is 25.00

G: normal

N: number, like 25000.ToString("N"), result is 25,000.00

P: percent

R: round trip, to ensure the precision

#,000.000: converted into specified format

0.###E-000: converted into specified format

The specific meaning of the variable command script is as follows:

Get the variable value of the designated variable path：

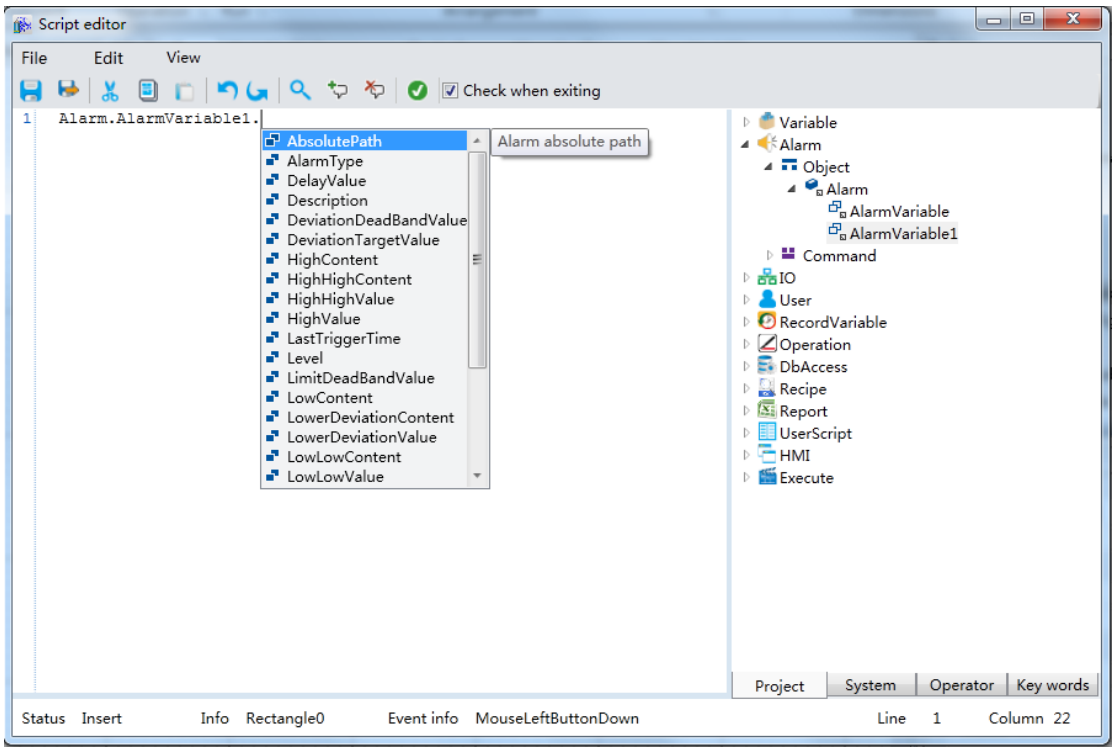
VarCmd.GetVariableValue("Var.analog","002"), Var.analog presents the designated variable path(VariablePath); 002 presents error code (errorValue), when executing fails， errorValue is returned.

Set the variable value of the designated variable path：

VarCmd.SetVariableValue("Var.analog",5), Var.analog presents the designated variable path(VariablePath); 5 presents set value(Value), the return value is True or False.

Alarm script usage specification：

Click the variable on the right side of script editor, choose the corresponding variable and then edit the property of the corresponding variable in the script editor as shown in the figure:



Alarm is divided into digital alarm and analog alarm by alarm variable;

The specific meaning of analog alarm is as follows：

Name	Type	Description
LastTriggerTime	Date	The time of the last alarm was triggered
DelayValue	Integer	Analog variable deviation or delay alarm time, unit second
LowLowValue	Real	Analog lolo alarm value
LowLowContent	String	Analog lolo alarm text
LowValue	Real	Analog low alarm value
LowContent	String	Analog low alarm text
HighValue	Real	Analog high alarm value

Get the absolute path of the alarm: Alarm.AnalogAlarm.AbsolutePath(read only)

Get the last triggering time of the alarm: Alarm.AnalogAlarm.LastTriggerTime(read only)

Get the parent group of the alarm: Alarm.AnalogAlarm.ParentGroup(read only)

Get the unique ID of the alarm: Alarm.AnalogAlarm.UniqueIdentifier(read only)

Get or Set the description of the alarm: Alarm.AnalogAlarm.Description=”Alarm Description”(read only)

Get or Set the time of deviation or delay alarm: Alarm.AnalogAlarm.DelayValue=5 (second)

Get or Set the deviation dead band of the alarm:

Alarm.AnalogAlarm.DeviationDeadBandValue=5

Get or Set the target of the deviation alarm:

Alarm.AnalogAlarm.DeviationTargetValue=5

Get or Set the type of the alarm: Alarm.AnalogAlarm.AlarmType=1 (Lowlow:1; Low:2; High:4; Higher:8; Minor:16; Major:32; Rate of change:64; Dead band of Overage:128; Delay:256; On:512;Off:1024;Displacement:2048; Deviation dead

band:4096. If the alarm type should be set as low(2) and High(4),the value is 6, Alarm.AnalogAlarm.AlarmType=6)

Get or Set the content of low low alarm:

Alarm.AnalogAlarm.LowLowContent=”lowlow alarm”

Get or Set the content of low alarm: Alarm.AnalogAlarm.LowContent=”low alarm”

Get or Set the content of high alarm: Alarm.AnalogAlarm.LowContent=”high alarm”

Get or Set the content of higher alarm:

Alarm.AnalogAlarm.LowContent=”high high alarm”

Get or Set the value of low low alarm: Alarm.AnalogAlarm.LowLowValue=5

Get or Set the value of low alarm: Alarm.AnalogAlarm.LowValue=20

Get or Set the value of high alarm: Alarm.AnalogAlarm.HighValue=80

Get or Set the value of higher alarm: Alarm.AnalogAlarm.HighHighValue=100

Get or Set the level of the alarm: Alarm.AnalogAlarm.Level=10 (Alarm level, the top level is 999, 0-199:Slight, 200-399:Lighter, 400-599:General, 600-799:Heavy, 800-999:Serious)

Get or Set the value of dead band limit: Alarm.AnalogAlarm.LimitDeadBandValue=5

Get or Set the content of deviation alarm’s lower limit:

Alarm.AnalogAlarm.LowerDeviationContent=”Lower deviation alarm”

Get or Set the content of deviation alarm’s upper limit:

Alarm.AnalogAlarm.UpperDeviationContent=”Upper deviation alarm”

Get or Set the content of rate of change alarm:

Alarm.AnalogAlarm.RateOfChangeContent=“Rate of change alarm”

Get or Set the value of deviation alarm’s lower limit:

Alarm.AnalogAlarm.LowerDeviationContent=5

Get or Set the value of deviation alarm’s upper limit:

Alarm.AnalogAlarm.LowerDeviationContent=50

Get or Set the value of rate of change alarm:

Alarm.AnalogAlarm.RateOfChangeValue=20

Get or Set the unit of rate of change alarm:

Alarm.AnalogAlarm.RateOfChangeUnit=60 (The unit of rate of change, one second is 1,one minute is 60, one hour is 3600)

The specific meaning of digital alarm is as follows：

Name	Type	Description
OnAlarmContent	String	Digital on alarm text
OffAlarmContent	String	Digital off alarm text
OnToOffAlarmContent	String	Digital on to off alarm text
OffToOnAlarmContent	String	Digital off to on alarm text
UniqueIdentifier	Integer	Alarm of unique identification
Name	String	Alarm name
AbsolutePath	String	Alarm absolute path

The different usage of digital alarm is that it is discrete alarm, it can be set as OffAlarm, OnAlarm and displacement Alarm, the usage is as follows:

Get or Set the content of OffAlarm: Alarm.DigitalAlarm.OffAlarmContent=”OffAlarm”

Get or Set the content of OnAlarm: Alarm.DigitalAlarm.OnAlarmContent=”OnAlarm”

Get or Set the content of OnAlarm to OffAlarm:

Alarm.DigitalAlarm.OnToOffAlarmContent=”OnAlarm to OffAlarm”

Get or Set the content of OffAlarm to OnAlarm:

Alarm.DigitalAlarm.OffToOnAlarmContent=”OffAlarm to OnAlarm”

The specific meaning of property script of alarm group is as follows:

Name	Type	Description
UniqueIdentifier	Integer	Alarm group of unique identification
Name	String	Alarm group name
AbsolutePath	String	Alarm group absolute path
HasAlarm	Discrete	Current alarm group whether there is alarm
Description	String	Alarm group description
AlarmCount	Integer	The current number of alarm which in alarm status
AckedAlarmCount	Integer	The current number of alarm which in respose status

The property script of alarm group is aimed at the operation of the whole group, take Alarm as an example:

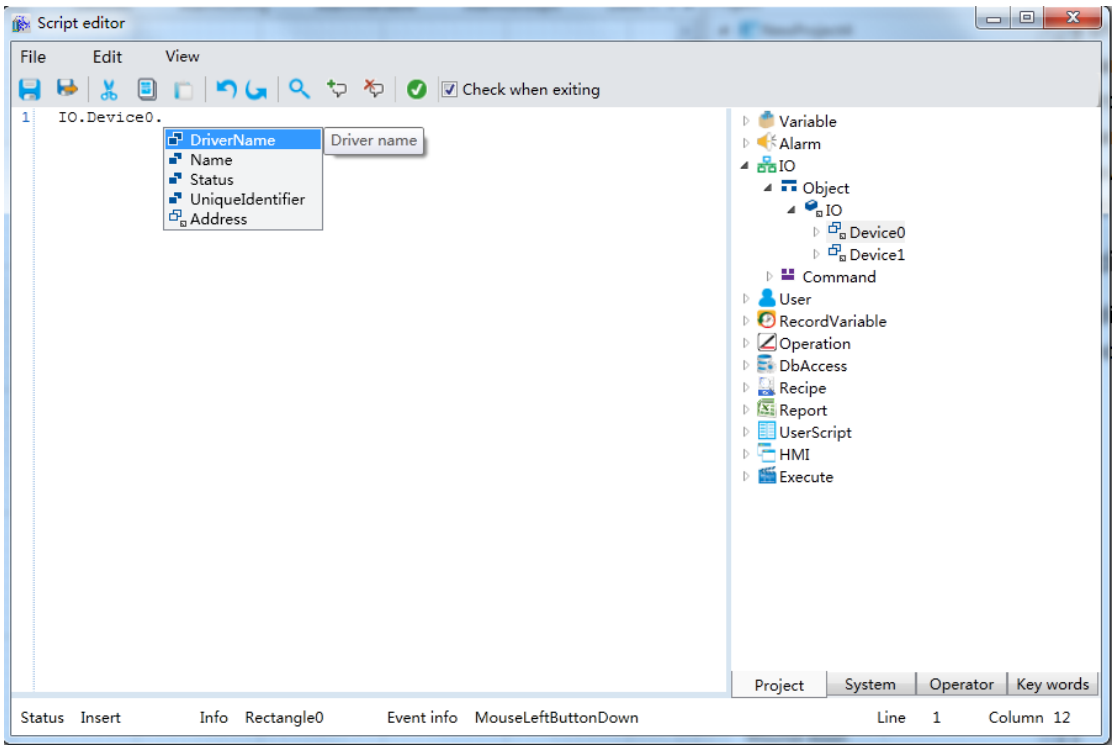
1) Whether there is an alarm in this group: Alarm.HasAlarm(read only)
2) Get the alarm count in alarm status in this group:
3) Alarm.AlarmCount(read only)
4) Get the alarm count in answer status in this group:
5) Alarm.AckedAlarmCount(read only)
6) Get all the alarms in alarm and answer status in this group:
7) Alarm.AllAlarmCount(read only)

The specific meaning of property script of alarm group is as follows:

1) Get all the alarms in answer status in all groups:
2) AlarmCmd.AckedAlarmCount(read only)
3) Get the alarm count in alarm status in this group:
4) AlarmCmd.AlarmCount(read only)
5) Get all the alarms and answers in this group:
6) AlarmCmd.AllAlarmCount(read only)
7) Save alarm configuration: AlarmCmd.SaveAlarmConfig(), it is used to save the changed alarm configuration in the runtime based on the requirement.

Communication Script usage specification：

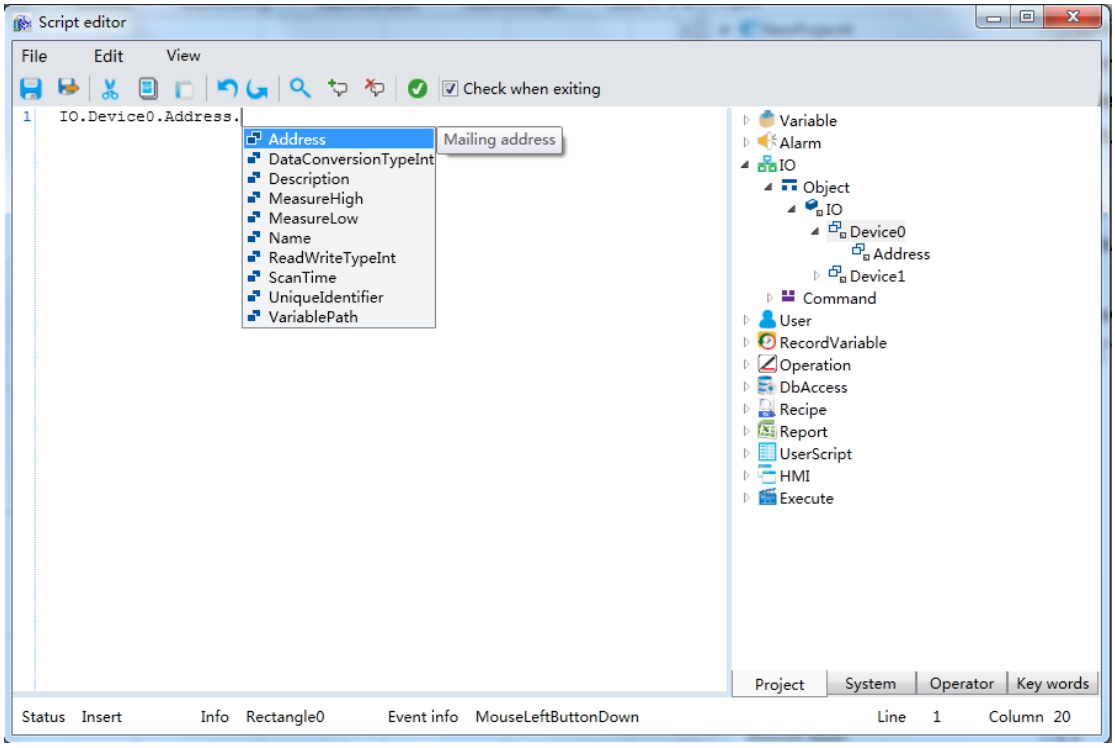
Click the IO on the right side of script editor, choose the corresponding IO device and then edit the property of the corresponding device in the script editor as shown in the figure:



The specific meaning of device property script is as follows:

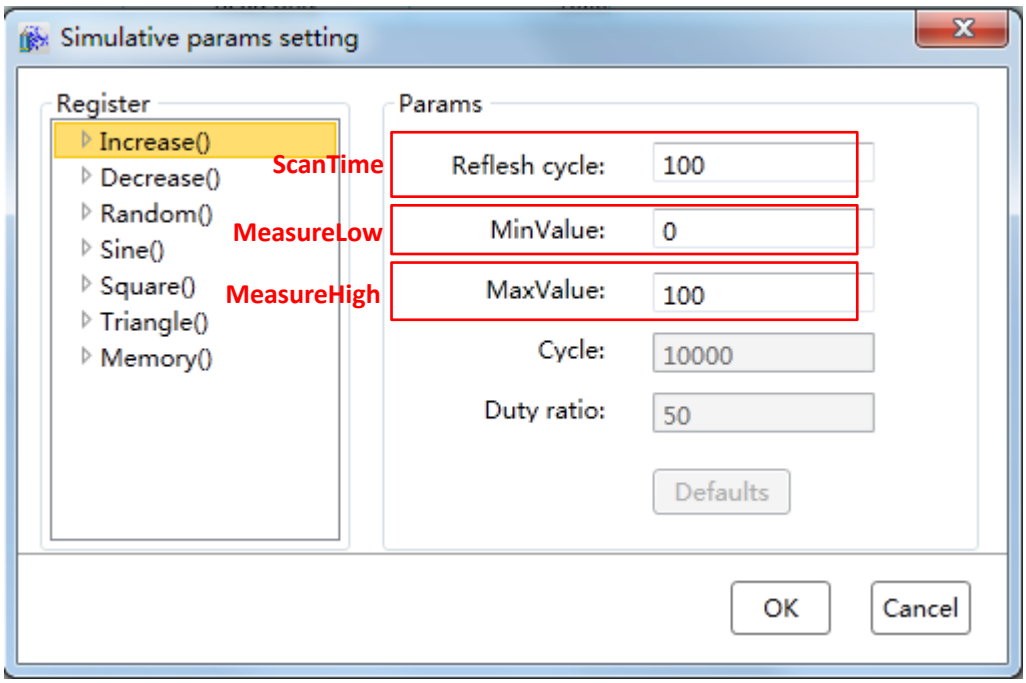
Name	Type	Description
UniqueIdentifier	Integer	Unique identification
Name	String	Device name
DriverName	String	Driver name
Status	Integer	Access to the device status, 1 normal 0, off, bigger than 1 stop
StatusVariablePath	String	Device state binding variable, used to bind variables to show the state of IO Device
IsStartVariablePath	String	Equipment start and stop binding variable, the binding variable control settings start and

1) Get the only ID of device: IO.Device0.UniqueIdentifier(read only)
2) Get the name of device: IO.Device0.Name(read only)
3) Get the driver name of device: IO.Device0.DriverName=”Analog type”(read only)
4) Set the status of device: IO.Device0.Status =0(positive integer, 0: connected, 1: disconnected, others: Stopped)
5) Set the status binding variable of device: IO.Device0.StatusVariablePath=“var.NewVariable”(“String, the bound variable is the status value of device ”)
6) Set the start-stop binding variable of device: IO.Device0. IsStartVariablePath=”Var.NewVariable”(“String, the bound variable can be used to start or stop device ”)



The specific meaning of property script of device address is as follows:

1) Get IO communication address: IO.Device0.Address.Address(read only)
2) Set IO data Conversion Type:
   IO.Device0.Address.DataConversionTypeInt=0("0-none, 1-linear transformation, 2-square transformation")
3) Set IO description: IO.Device0.Address.Description="Sample IO"(String)
4) Set the maximum value of IO project: IO.Device0.Address.MeasureHigh=100
   Set the minimum value of IO project: IO.Device0.Address.MeasureLow=0
5) Set the scan time of IO: IO.Device0.Address.ScanTime=100



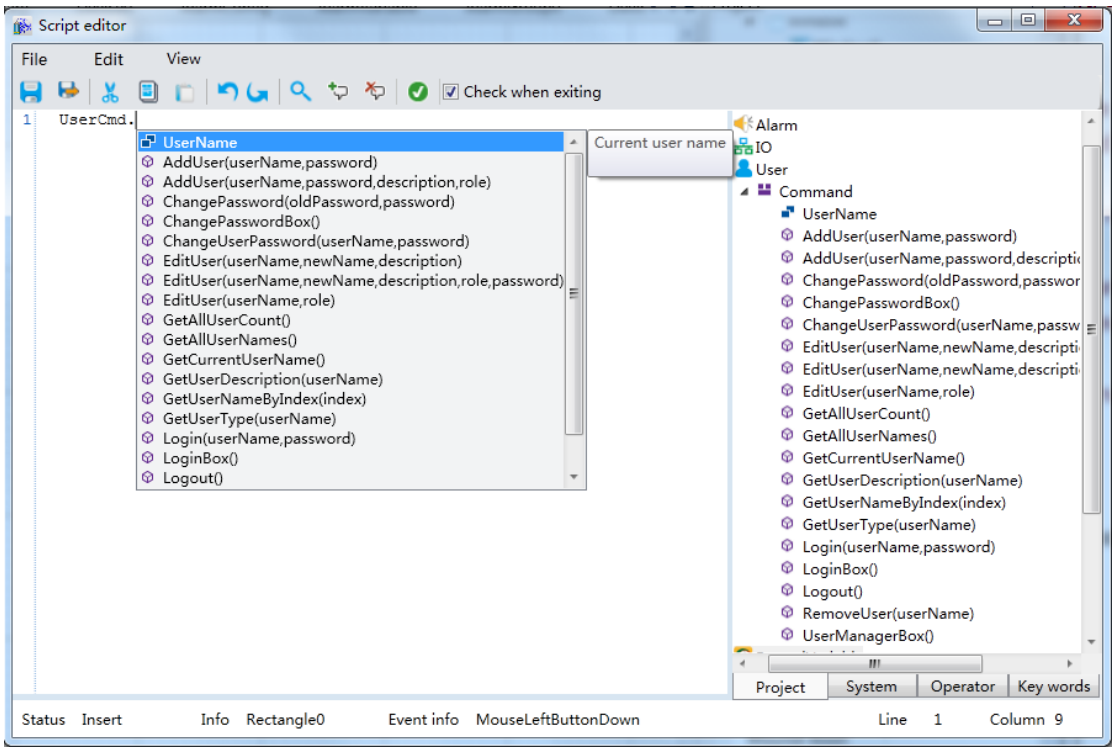
6) Get the name of IO: IO.Device0.Address.Name(read only)
7) Set the type of reading and writing: IO.Device0.Address.ReadWriteTypeInt=0(“0-read and write, 1-read only, 2-write only”)
8) Set the only ID of IO: IO.Device0.Address.UniqueIdentifier(“read only”)
9) Set the bound variable path of IO: IO.Device0.Address.VariablePath=”Var.NewVariable”(String)

The specific meaning of communication command script：

1) Get the status of device: IOCmd.GetDeviceStatus(“Device0”)(parameter is the device name, String; the return value : 0 is connected, 1 is disconnected and others are stopped)
2) Set the status of device: IOCmd.SetDeviceStatus(“Device0”,0)(parameter 1 is device name, String; parameter 2 is integer: 0 is connected, 1 is disconnected and others are stopped)
3) Start all devices in the specified driver: IOCmd.StartAllDevicesByDriver(“IO.Device0.DriverName”)(parameter is the driver, String, it is recommended that the chosen driver name of the device should be taken as parameter)
4) Start the specified device: IOCmd.StartDevice(“Device0”)(parameter is the name of device, String)
5) Stop all devices in the specified driver: IOCmd.StopAllDevicesByDriver(parameter is the driver, String, it is recommended that the chosen driver name of the device should be taken as parameter)
6) Stop the specified device: IOCmd.StopDevice(“Device0”)(parameter is the name of device, String)

User Script usage specification：

Click the User on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The usage of the variable command script：

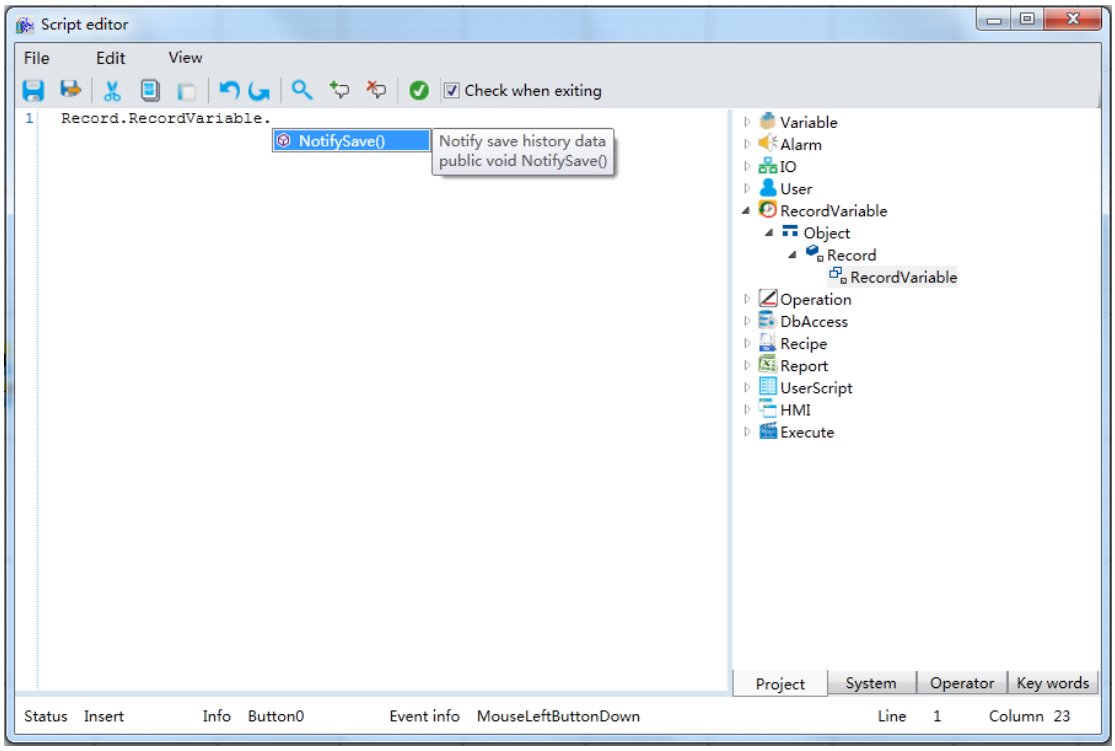
1) Current user name: UserCmd.UserName (), the type of return value is String.
2) Add a user: UserCmd.AddUser(engineer, 123), engineer presents user name(userName); 123 presents password(password); the type of return value is Bool.
3) Add a user: UserCmd.AddUser(engineer, 123,”I am an engineer”, ”Admin”), engineer presents user name(userName); 123 presents password(password); ”I am an engineer” presents description(description), “Admin” presents user role(role); the type of return value is Bool.
4) Change the password of current user: UserCmd.ChangePassword(123,111), 123 presents the old password, 111 presents the new password; the type of return value is Bool.
5) Change the password: UserCmd.ChangeUserPasswordBox(),the type of return value is Bool.
6) Change the user password: UserCmd.ChangeUserPassword(engineer, 111), engineer presents user name(userName); 111 presents new password(password), the type of return value is Bool.
7) Edit the name and description of user(only System Administrator has the rights):
8) UserCmd.EditUser(engineer, engineerLi, “The name has been changed”), engineer presents user name(“userName”); engineerLi presents new user name(edituserName); “The name has been changed” presents description (description), the type of return value is Bool.
9) Edit user: UserCmd.EditUser(engineer, engineerLi, “The name has been changed”, “Admin”, 111), engineer presents user name (userName); engineerLi presents new user name(edituserName); “The name has been changed” presents description (description); “Admin” presents user role(role); 111 presents

password(password), the type of return value is Bool.

10) Edit user type: UserCmd.EditUser(engineer, “Admin”), engineer presents user name (userName); “Admin” presents user type, the type of return value is Bool.
11) Get the number of user: UserCmd.GetAllUserCount(),the type of return value is Int.
12) Get the name list of user: UserCmd.GetAllUserNames(),the type of return value is String.
13) Get the name of current user: UserCmd.GetCurrentUserName(),the type of return value is String.
14) Get the description of current user: UserCmd.GetUserDescription(engineer), engineer presents the current user(userName), the type of return value is String.
15) Get the user name by index: UserCmd.GetUserNameByIndex(3), 3 presents the user index(index), the type of return value is String.
16) Get the type of current user: UserCmd.GetUserType(engineer), engineer presents current user(userName), the type of return value is String.
17) User login: UserCmd.Login(engineer,123), engineer presents user name (userName); 123 presents password (password), the type of return value is Bool.
18) User login : UserCmd.LoginBox(), pop up a message box for user login.
19) User login out: UserCmd.Logout().
20) Remove user: UserCmd.RemoveUser(engineer), the type of return value is Bool.
21) User management: UserCmd.USerManagerBox (), pop up a message box to manage user.

Record variable script usage specification：

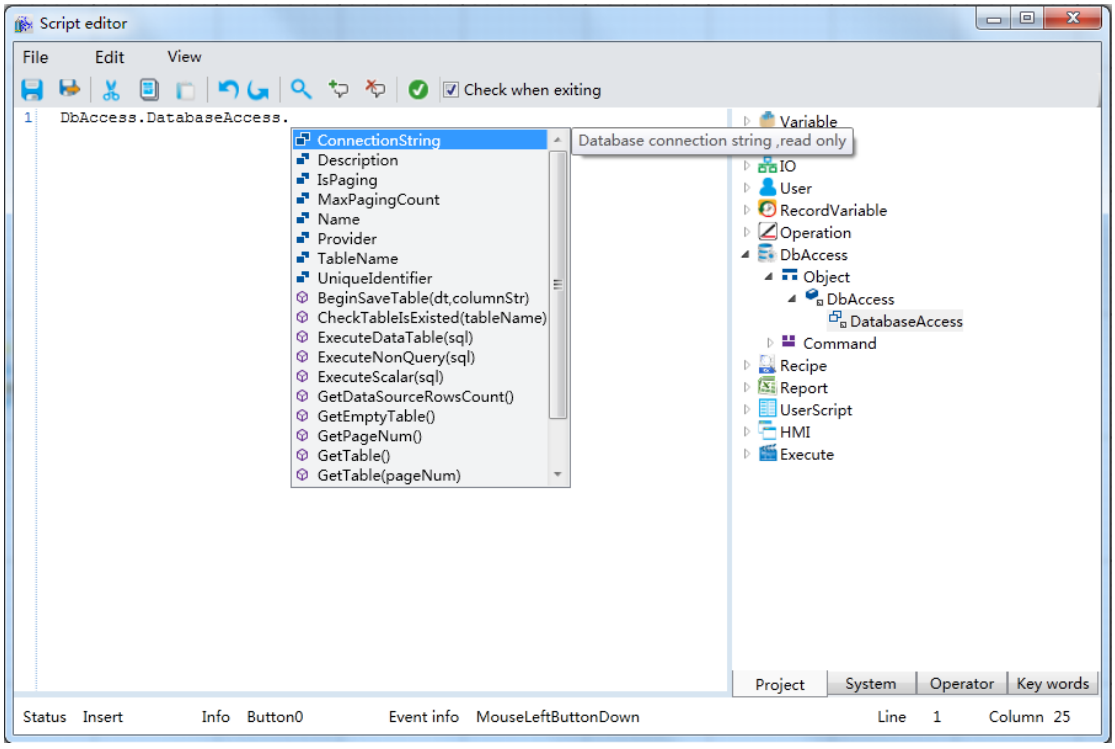
Click the” RecordVariale” on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



Notification save history data: Record.RecordVariable.NotifySave ()

Database access script usage specification：

Click the" DbAccess" on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of database access property script is as follows：

Name	Type	Description
UniqueIdentifier	Integer	Unique Id,read-only
Name	String	Database access name,read-only
IsPaging	Discrete	Is paging
MaxPagingCount	Integer	Table max count
Description	String	Description
ConnectionString	String	Database connection string ,read only
Provider	String	Display provider of the database connection,read-only
TableName	String	Table name,read-only

Get the only ID of database access: DbAccess.DatabaseAccess.UniqueIdentifier
Get the name of database access: DbAccess.DatabaseAccess.Name
Whether to do pagination: DbAccess.DatabaseAccess.IsPaging = True (True is to page, False is not).
The max paging count of database access:
DbAccess.DatabaseAccess.MaxPagingCount = 10000
Set the description of database access: DbAccess.DatabaseAccess.Description
Database connection string: DbAccess.DatabaseAccess.ConnectionString
Display the provider of database connection: DbAccess.DatabaseAccess.Provider (read only)
Get the name of table 0f database access: DbAccess.DatabaseAccess.TableName

(read only)

The database access usage examples：

1. Display the content of database:

Way 1:

Script: dt = DbAccess.DatabaseAccess.GetTable ()

```
Call report0.ShowDataTableForReport (1, 1, dt)
```

Way2:

Script: sql = “select * from” + DbAccess.DatabaseAccess.TableName

```
dt = DbAccessCmd.ExexuteDataTable
(“DbAccess.DatabaseAccess”,”qslservercompact”,sql)
call report0.ShowDataTableForReport (1, 1, dt)
```

2. Create a new data table:

Script: dt = DbAccessCmd.CreateTable (“newTable”)

```
call DbAccessCmd.AddColumn (dt, “Id”,”Id”, 1,500, false, true, true)
call DbAccessCmd.AddColumn (dt, “Name”,” name”, 4,500, false, false, false)
call DbAccessCmd.AddColumn (dt, “Age”,” age”, 1,500, false, false, false)
call DbAccessCmd.SetPrimary (dt, “Id”)
returnValue = DbAccessCmd.ExecuteCreateDatabaseTable
(“DbAccess.DatabaseAccess”,”sqlservercompact”,dt,””)
if returnValue = true, Then
    MsgBox ”Create data table successfully”
End if
```

3. Add data row of data table:

Script: dt = DbAccess.DatabaseAccess.GetTable(“newTable”)

```
row = dt.NewRow()
row(“Name”) = “Lily”
row(“Age”) = 15
call dt.Rows.Add(row)
call DbAccessCmd.ExecuteSave
(“DbAccess.DatabaseAccess ”, ”sqlservercompact”,dt,””)
dt1 = DbAccess.DatabaseAccess.GetTable(“newTable”)
call report0.ShowDataTableForReport(1,1,dt1)
```

4. Insert data row of data table:

Script: sql = “Insert into newTable([Name],[Age])Values(‘Lucy’,20)”

```
call DbAccessCmd.ExecuteNonQuery
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
dt = DbAccess.DatabaseAccess.GetTable(“New Table”)
call report0.ShowDataTableForReport(1,1,dt)
```

5. Get the data from data table and save in data table:

Way1:

Script: dt = report0.GetDataTableFromReport(1)

```
dt.tableName = “NewTable”
call DbAccessCmd.ExecuteSave
```

(“DbAccess.DatabaseAccess”,”sqlservercompact”,dt,””)
dt1 = DbAccess.DatabaseAccess.GetTable(“NewTable”)
call report0.ShowDataTableForReport(1,1,dt1)

Way2(save to specified column):

Script: dt = report0.GetDataTableFromReport(1)
dt.tableName = “NewTable”
RF = DbAccessCmd.ExecuteSave(“DbAccess.DatabaseAccess”,”
sqlservercompact”,dt,”Id,Name”)(when you update specified column, you must add the main key)
MsgBox RF
dt1 = DbAccess.DatabaseAccess.GetTable(“NewTable”)
call report0.ShowDataTableForReport(1,1,dt1)

6. Delete the whole data table:

Script: result =
DbAccessCmd.ExecuteDropDatabaseTable(“DbAccess.DatabaseAccess”,”
sqlservercompact”,”NewTable”)
if result = True Then
MsgBox ”Delete database successfully”
end if

7. Delete the specified columns

Script: sql = “delete NewTable where Id between 1 and 10”
call DbAccessCmd.ExecuteNonQuery
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)

8. Query data of the whole table:

Script: sql = “Select * from NewTable”
dt = DbAccessCmd.ExecuteDataTable
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
call report0.ShowDataTableForReport(1,1,dt)

9. Query data of specified rows and columns

Script: sql = “Select top 10 [Id],[Name] from NewTable”
dt = DbAccessCmd.ExecuteDataTable
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
call report0.ShowDataTableForReport(1,1,dt)

10. Query data of specified time range:

Script:
startTime = CStr(DateTimePicker0.ValueTime)
endTime = CStr(DateTimePicker1.ValueTime)
sql = “ SELECT * FROM ” + DbAccess.DatabaseAccess.TableName + “where TriggerTime between’” + startTime + “’and’” + endTime + “’”
dt = DbAccessCmd.ExecuteDataTable
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
call Report0.ShowDataTableForReport(1,1,dt)

11. Query the top 10 rows data

Script: dt = DbAccessCmd.ExecuteDataTable

(“DbAccess.DatabaseAccess”,”sqlservercompact”,”NewTable”,10)
call Report0.ShowDataTableForReport(1,1,dt)

12. Query between two data tables:

Script: sql = “Select * from NewTable,” + DbAccess.DatabaseAccess.TableName + “where NewTable.Id =1 and” + DbAccess.DatabaseAccess.TableName + “.Id = NewTable.Id”
dt = DbAccessCmd.ExecuteDataTable
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
call Report0.ShowDataTableForReport(1,1,dt)

13. Return the value of first row and first column

Script: sql = “Select * from NewTable”
button0.Content = DbAccessCmd.ExecuteScalar
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)

14. Update data with SQL

Script: sql = “Update newTable set [Name] = ‘updateName’ where Id = 5”
call DbAccessCmd.ExecuteNonQuery
(“DbAccess.DatabaseAccess”,”sqlservercompact”,sql)
dt = DbAccess.DatabaseAccess.GetTable(“NewTable”)
call Report0.ShowDataTableForReport(1,1,dt)

15. Get the data of report and update data to the corresponding columns of database

Script: dt = Report0.GetReportDataTable(0)
SQL = “update NewTable set Name =@Name, Age =@Age where Id =@Id”
call DbAccessCmd.ExecuteNonQuery
(“DbAccess.DatabaseAccess”,”sqlservercompact”,dt,SQL)
dt1 = DbAccess.DatabaseAccess.GetTable(“NewTable”)
call Report0.ShowDataTableForReport(1,1,dt)

16. Batch update data to database:

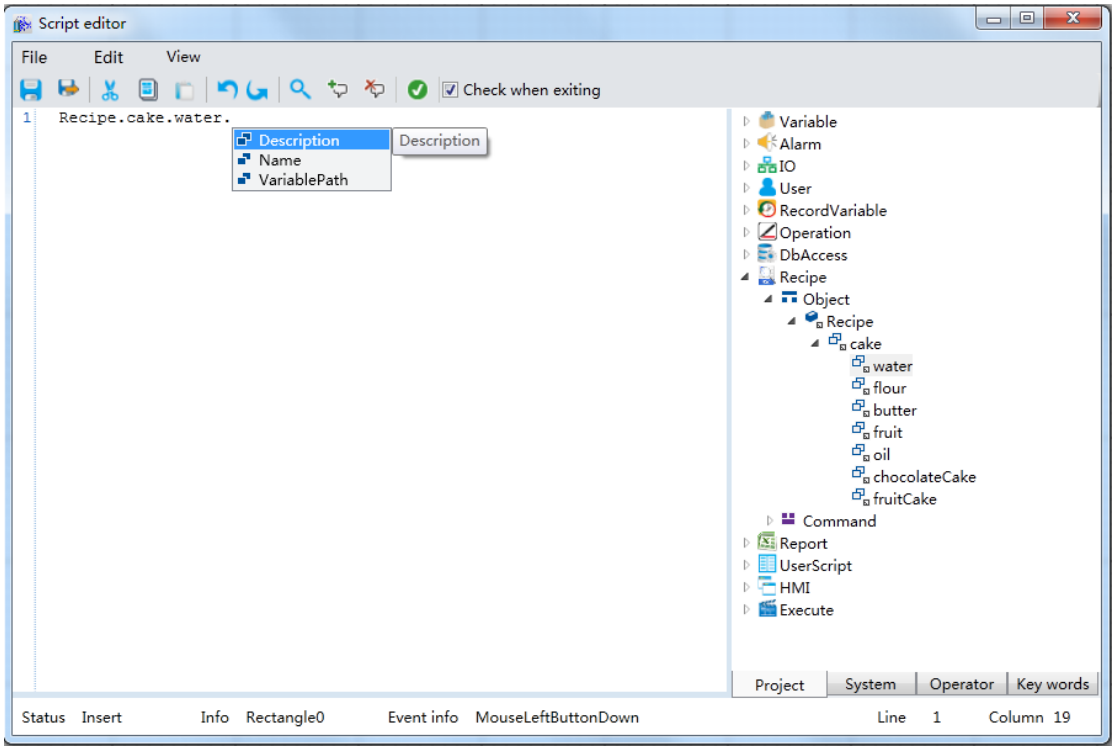
Script: dt = Report0.GetReportDataTable(0)
cc = DbAccess.DatabaseAcess.BatchInsertData(dt)

17. Sort the data table:

Script: dt = Report0.GetReportDataTable(0)
newdt = DbAccessCmd.SelectDT(dt,”Id > 16050”, “Id Desc”)
call Report0.ShowDataTableForReport(1,1,newdt)

Recipe Script usage specification：

Click the Recipe on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of recipe property script is as follows：

1) Get the description of recipe element: Recipe.Cake.water.Description(read only)
2) Get the name of recipe element: Recipe.Cake.water.Name(read only)
3) Get the related variable of recipe element: Recipe.Cake.water.VariablePath(read only)

The usage of recipe command script：

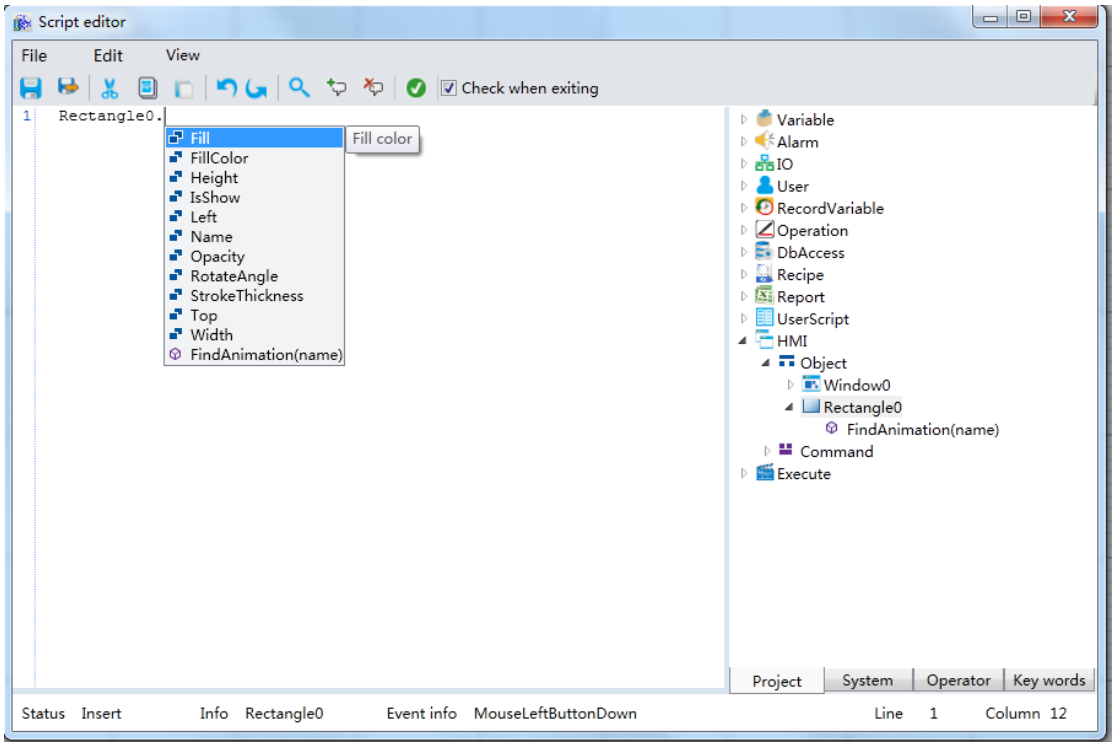
1) Add recipe: RecipeCmd.AddRecipe(“cake”, ”flavors”), “cake” presents recipe name (recipeName), “flavors” presents recipe description(description).
2) Add recipe element : RecipeCmd.AddRecipe(“cake”, ”water”, ” cake element”), “cake” presents recipe name (recipeName), ”water” presents recipe element Name (recipeElementName), ”add cake element” presents recipe element description (description).
3) Add recipe item: RecipeCmd.AddRecipeItem(“cake”, “chocolateCake”, “cake item”), “cake” presents recipe name (recipeName), “chocolateCake” presents recipe item name, “cake item” presents recipe item description (description).
4) Export recipe to Excel file: RecipeCmd.ExportRecipeToExcel(“cake”, ”D:1.xlsx”, “Excel 2010”), “case” presents recipe name(recipeName), “D:\1.xlsx”presents the file path(filePath), “Excel 2010”prsents the version of Excel(excelVersion), the type of return value is Bool.
5) Get the item of current recipe: RecipeCmd.GetCurrentRecipeItem(“cake”), “cake” presents recipe name(recipeName), the type of return value is String.
6) Get the element value of specified recipe item: RecipeCmd.GetRecipeItemValue(“cake”, ”chocolateCake”, “water”), “cake” presents recipe name(recipeName), “chocolateCake” presents recipe item name

(recipeitemName), “water” presents recipe element name(recipeElementName).

7) Export Excel to specified recipe: RecipeCmd.ImportRecipeFromExcel(“D:\1.xlsx”, “cake”, “Excel 2010”), “D:\1.xlsx”presents the exported file path(filePath), “cake” presents recipe name(“recipeName”), “Excel 2010” presents excel version(excelVersion), the type of return value is Bool.
8) Load recipe item to variable: RecipeCmd.LoadRecipeItem(“cake”, ”chocolateCake”), “cake” presents recipe name(“recipeName”), “chocolateCake” presents recipe item name (recipeitemName).
9) Remove recipe: RecipeCmd.RemoveRecipe(“cake”), “cake” presents recipe name (recipeName).
10) Remove recipe element: RecipeCmd.RemoveRecipeElement(“cake”, “water”), “cake” presents recipe name(“recipeName”), ”water” presents recipe element name (recipeElementName).
11) Remove recipe item: RecipeCmd.RemoveRecipeItem(“cake”, “chocolateCake”), “cake” presents recipe name(recipeName), “chocolateCake” presents recipe item name (recipeitemName).
12) Save variable value to recipe item: RecipeCmd.SaveToRecipeItem(“cake”, “chocolateCake”), “cake” presents recipe name(recipeName), “chocolateCake” presents recipe item name (recipeitemName).
13) Set recipe element value of specified recipe item: RecipeCmd.SetRecipeItemValue (“cake”, “chocolateCake”, ”water”, 30), “cake” presents recipe name(recipeName), “chocolateCake” presents recipe item name (recipeitemName), ”water” presents recipe element name (recipeElementName), 30 presents the set value(value), the type of return value is Bool.

Window Script usage specification：

Basic graphs script usage specification(take rectangle as an example):



Set the color of rectangle: Rectangle0.Fill = Color.Black

Set the color of rectangle: Rectangle0.Fillcolor = ”#FFF000”(“#FFFF0000” presents the element of color).

Set the height of rectangle: Rectangle0.Height = 100

Set the display status of rectangle: Rectangle0.isShow = True (True is to show, False is to hide).

Set the left coordinate of rectangle: Rectangle0.Left = 50

Get the name of rectangle: Rectangle0.Name(read only)

Set the opacity of rectangle: Rectangle0.Opacity = 0.5 (less than 0: transparent, more than 1: opaque)

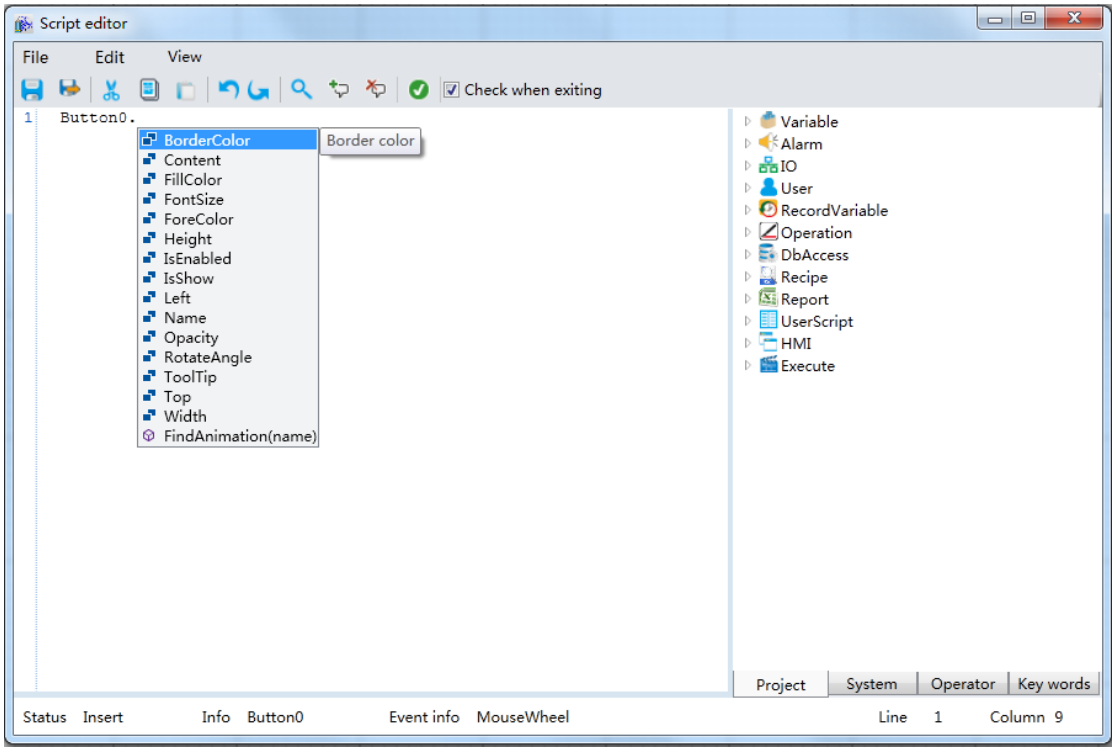
Set the rotating angle of rectangle: Rectangle0.RotateAngle = 45

Set the line width of rectangle: Rectangle0.StrokeThickness = 5

Set the top coordinate of rectangle: Rectangle0.Top = 100

Set the width of rectangle: Rectangle0.Width = 100

Window control script usage specification(take rectangle as an example):

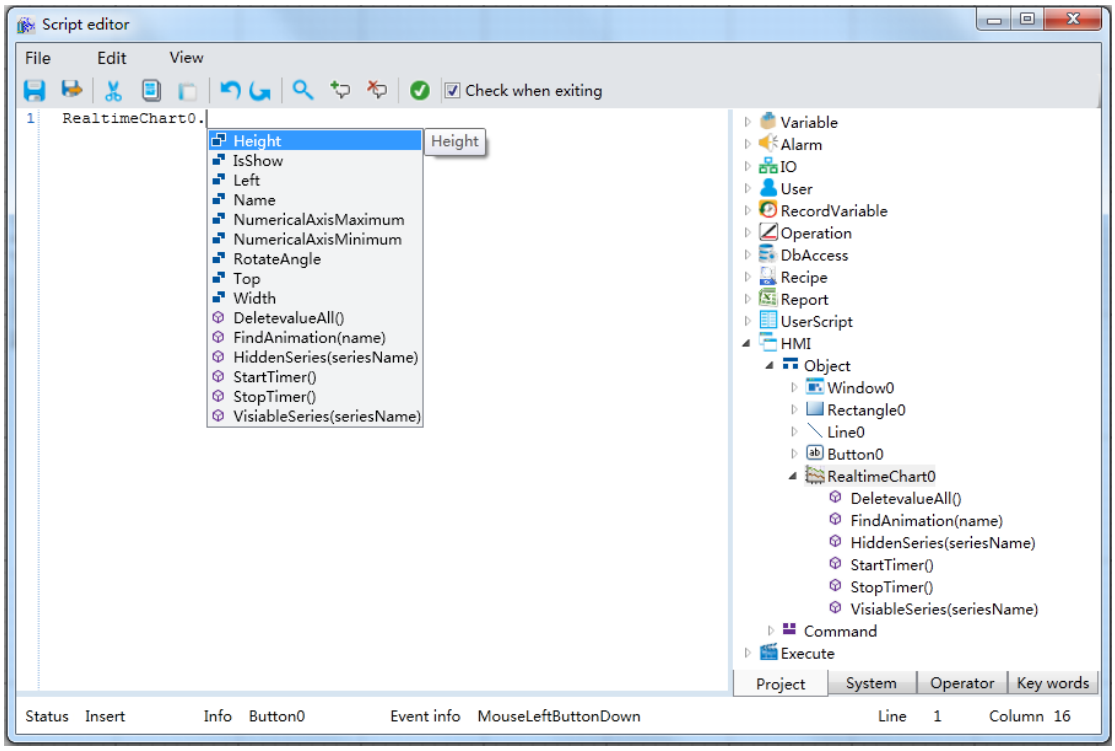


1) Set the border color of button: Button0.BorderColor = "#FFFF0000"(“#FFFF0000” presents the element of color).
2) Set the content of button: Button0.Content = “Button”(String)
3) Set the filled color of button: Button0.FillColor = “#000000”
    1. Button0.FillColor = “#FF000000”
    2. (String, either is OK)
4) Set the font size of button: Button0.FontSize =16
5) Set the fore color of button: Button0.ForeColor = "#FFFF0000"(“#FFFF0000” presents the element of color).
6) Set the height of button: Button0.Height = 100
7) Set the operability of button: Button0.isEnabled = True (True is operable, False is not operable)
8) Set the display status of button: Button0.IsShow = True (True is to display, False is to hide)
9) Set the left coordinate of button: Button0.Left = 50
10) Get the name of button: Button0.Name(read only)
11) Set the opacity of button:Button0.Opacity = 0.5 (less than 0: transparent, more than 1: opaque)
12) Set the rotating angle of button: Button0.RotateAngle = 45
13) Set tip text of button: Button0.ToolTip = “control button” (String, when mouse stays on the button, the text will be showed)
14) Set the top coordinate of button: Button0.Top = 100
15) Set the width of button: Button0.Width = 100

Extended controls usage specification：

Real time chart script usage specification:

Click the real time chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of real time chart property script is as follows：

Name	Type	Description
NumericalAxisMinimum	Real	NumberAxis MinValue
NumericalAxisMaximum	Real	NumberAxis MaxValue
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

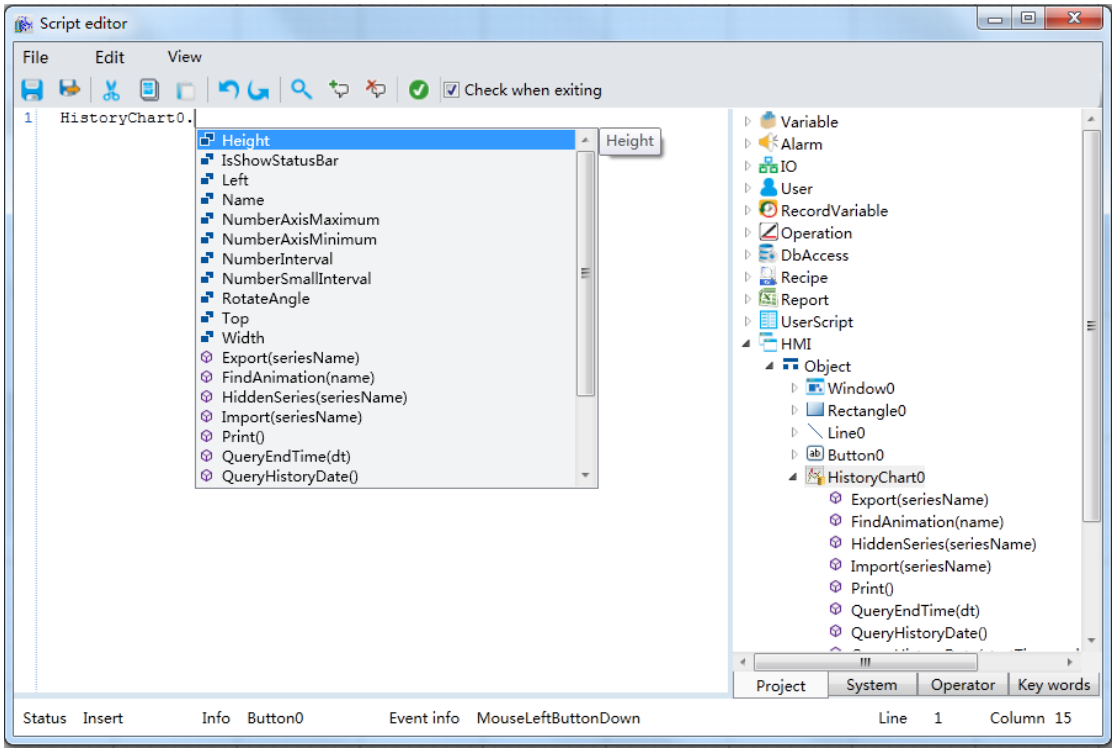
1) Set the height of real time chart: RealTimeChart0.Height = 300(read and write)
2) Set the display status of real time chart: RealTimeChart0.IsShow = True (True is to display, False is to hide)
3) Set the left coordinate of real time chart: RealTimeChart0.Left = 10 (read and write)
4) Set the name of real time chart: RealTimeChart0.Name(read only)
5) Set the maximum value of numerical axis of real time chart: RealTimeChart0.NumericalAxisMaximum = 100 (read and write)
6) Set the minimum value of numerical axis of real time chart: RealTimeChart0.NumericalAxisMinimum = 0 (read and write)
7) Set the rotating angle of real time chart: RealTimeChart0.RotateAngle = 45
8) Set the top coordinate of real time chart: RealTimeChart0.Top = 50
9) Set the width of real time chart: RealTimeChart0.Width = 50

Real time chart command script usage specification：

1) Delete all the value: RealTimeChart0.DeletevalueAll()
2) Hide the series of real time chart: RealTimeChart0.HiddenSeries(“series0”) (“series0” presents the series of the real time chart)
3) Display the series of real time chart: RealTimeChart0.VisibleSeries(“series0”) (“series0” presents the series of the real time chart)
4) Start timer: RealTimeChart0.StartTimer()
5) Stop timer: RealTimeChart0.StopTimer()

History chart script usage specification:

Click the history chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of history chart property script is as follows：

Name	Type	Description
IsShowStatusBar	Discrete	Whether to display toolbar
NumberAxisMinimum	Real	NumberAxis MinValue
NumberAxisMaximum	Real	NumberAxis MaxValue
NumberInterval	Real	NumberAxis MaxScale
NumberSmallInterval	Real	NumberAxis MinScale
Name	String	Name attribute read only
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

1) Set the visibility of status bar: HistoryChart0.IsShowStatusBar = True (True is to

show, False is to hide)

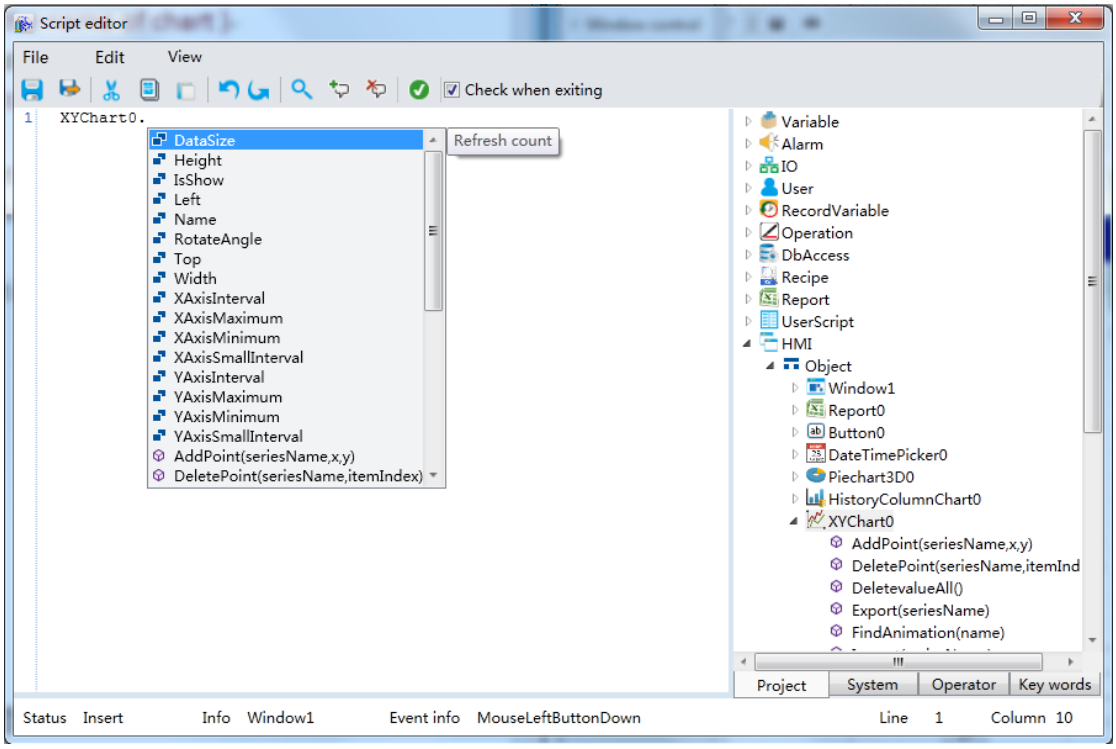
2) Set the minimum value of numerical axis of history chart: HistoryChart0.NumericalAxisMinimum = 0 (read and write)
3) Set the maximum value of numerical axis of history chart: HistoryChart0.NumericalAxisMaximum = 100 (read and write)
4) Set the number interval of history chart: HistoryChart0.NumberInterval = 4
5) Set the small number interval of history chart: HistoryChart0.NumberSmallInterval = 10
6) Set the name of history chart: HistoryChart0.Name (read only)
7) Set the width of history chart: HistoryChart0.Width = 50(read and write)
8) Set the height of history chart: HistoryChart0.Height = 300(read and write)
9) Set the left coordinate of history chart: HistoryChart0.Left = 100(read and write)
10) Set the top coordinate of history chart: HistoryChart0.Top = 200(read and write)
11) Set the rotation angle of history chart: HistoryChart0.RotateAngle = 60(read and write)

History chart command script usage specification：

1) Export series of history chart: HistoryChart0.Export(“series0”)(“series0” presents the chart that you want to export).
2) Hide the series of history chart: HistoryChart0.HiddenSeries (“series0”)(“series0” presents the chart that you want to hide).
3) Import series of history chart: HistoryChart0.Import(“series0”)(“series0” presents the chart that you want to Import).
4) Print the series of history chart: HistoryChart0.Print().
5) Set the end time of the query data of history chart: HistoryChart0.QueryEndTime = "6/16/2016 17:30:00"
6) Set the start time of the query data of history chart: HistoryChart0.QueryStartTime = "6/16/2016 16:30:00"
7) Historical data query: HistoryChart0.QueryHistoryDate()
8) Historical data query: HistoryChart0.QueryHistoryDate("6/16/2016 16:30:00"
9) , "6/16/2016 17:30:00", 1000) (“6/16/2016 16:30:00" presents start time, "6/16/2016 17:30:00" presents end time, 1000 presents time interval).
10) Save series of history chart: HistoryChart0.Save()
11) Save series to specified path of history chart: Save(“D:\123”, “series0”)(“D:\123” presents the path of file, “series0” presents the name of chart)
12) Set the visibility of series of history chart: HistoryChart0.VisibleSeries(“series0”) (“series0” presents the chart that you want to Import).

XY chart script usage specification:

Click the XY chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of XY chart property script is as follows:

Name	Type	Description
DataSize	Real	Refresh count
XAxisMinimum	Real	X axis minmum value
XAxisMaximum	Real	X axis max value
YAxisMinimum	Real	Y axis minmum value
YAxisMaximum	Real	Y axis max value
XAxisInterval	Real	X axis max scale
XAxisSmallInterval	Real	X axis minmum scale
YAxisInterval	Real	Y axis max scale
YAxisSmallInterval	Real	Y axis minmum scale
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

1) Set the rotation angle of XY chart: XYChart0.RotateAngle = 60 (read and write)
2) Set the refreshing count of XY chart: XYChart0.DataSize = 5 (read and write, default is 20)
3) Set X axis minimum value of XY chart: XYChart0.XAxisMinimum = 0 (read and write)
4) Set X axis maximum value of XY chart: XYChart0.XAxisMaximum=10 (read and write)
5) Set X axis maximum scale of XY chart: XYChart0.XAxisInterval = 4 (read and write)
6) Set X axis minimum scale of XY chart: XYChart0.XAxisInterval = 5 (read and

write)

7) Set Y axis minimum value of XY chart: XYChart0.YAxisMinimum = 0 (read and write)
8) Set Y axis maximum value of XY chart: XYChart0.YAxisMaximum = 10 (read and write)
9) Set Y axis maximum scale of XY chart: XYChart0.XAxisInterval = 4 (read and write)
10) Set Y axis minimum scale of XY chart: XYChart0.XAxisInterval = 5 (read and write)
11) Set the name of XY chart: XYChart0.Name(read only)
12) Set the display status of XY chart: XYChart0.IsShow = True (True is to display, False is to hide)
13) Set the width of XY chart: XYChart0.width = 100 (read and write)
14) Set the Height of XY chart: XYChart0.Height = 50 (read and write)
15) Set the left coordinate of XY chart: XYChart0.Left = 100(read and write)
16) Set the top coordinate of XY chart: XYChart0.Top = 200(read and write)

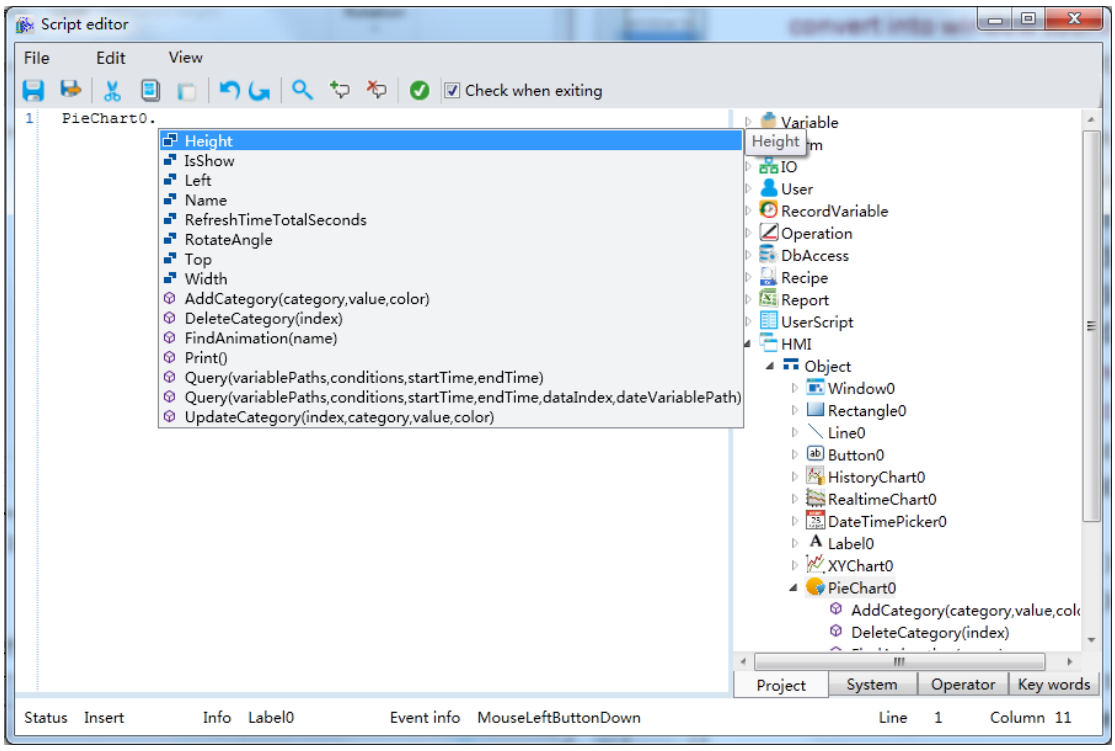
XY chart command script usage specification:

1) Add the new data point: XYChart0.AddPoint("SeriesConfig0", 200, 500) ("SeriesConfig0" is the name of chart, 200 is the X coordinate of new point, 500 is the Y coordinate of new point)
2) Delete the chosen point: XYChart0.DeletePoint("SeriesConfig0", 2)("SeriesConfig0" is the name of chart, 2 is the item index).
3) Delete all the point: XYChart0.DeleteValueAll()
4) Export the series: XYChart0.Export("SeriesConfig0")
5) Import the series: XYChart0.Import("SeriesConfig0")
6) Get the bottom coordinate: XYChart0.MarginBottom()(XY chart bottom coordinates convert into window coordinates)
7) Get the left coordinate: XYChart0.MarginLeft()(XY chart left coordinates convert into window coordinates)
8) Get the right coordinate: XYChart0.MarginRight()(XY chart right coordinates convert into window coordinates)
9) Get the top coordinate: XYChart0.MarginTop()(XY chart top coordinates convert into window coordinates)
10) Print the chart: XYChart0.Print()
11) Save the chart: XYChart0.Save()
12) Save the Series to specified path: XYChart0.Save("D:\ ", "Series Config0")
13) Start the Timer: XYChart0.StartTimer()
14) Stop the Timer: XYChart0.StopTimer()
15) Update the point: XYChart0.UpdatePoint("Series Config0", 2, 25, 75)("Series Config0" presents the name of chart, 2 presents item index, 25 presents X coordinate, 75 presents Y coordinate)
16) XY data point coordinates converted into window coordinates: XYChart0. XyValueTransform("20", "30")("20" presents the X coordinate, "30"presents the Y coordinate).

17) Ratio of window to X axis: XYChart0.XAxisTransformRatio()
18) Ratio of window to Y axis: XYChart0.YAxisTransformRatio()

Pie chart script usage specification：

Click the Pie chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of pie chart property script is as follows:

Name	Type	Description
RefreshTimeTotalSeconds	Integer	Refresh time by default is 1 second
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

1) Set the refreshing time of pie chart: PieChart0.RefreshTimeTotalSeconds = 2(read and write, the default time is one second, the type is integer).
2) Set the name of pie chart: PieChart0.Name(read only)
3) Set the display status of pie chart: PieChart0.IsShow = true (True is to display, False is to hide)
4) Set the width of pie chart: PieChart0.Width = 50
5) Set the height of pie chart: PieChart0.Height = 5
6) Set the left coordinate of pie chart: PieChart0.Left = 100
7) Set the top coordinate of pie chart: PieChart0.Top= 200
8) Set the rotation angle of pie chart: PieChart0.RotateAngle = 60

Pie chart command script usage specification：

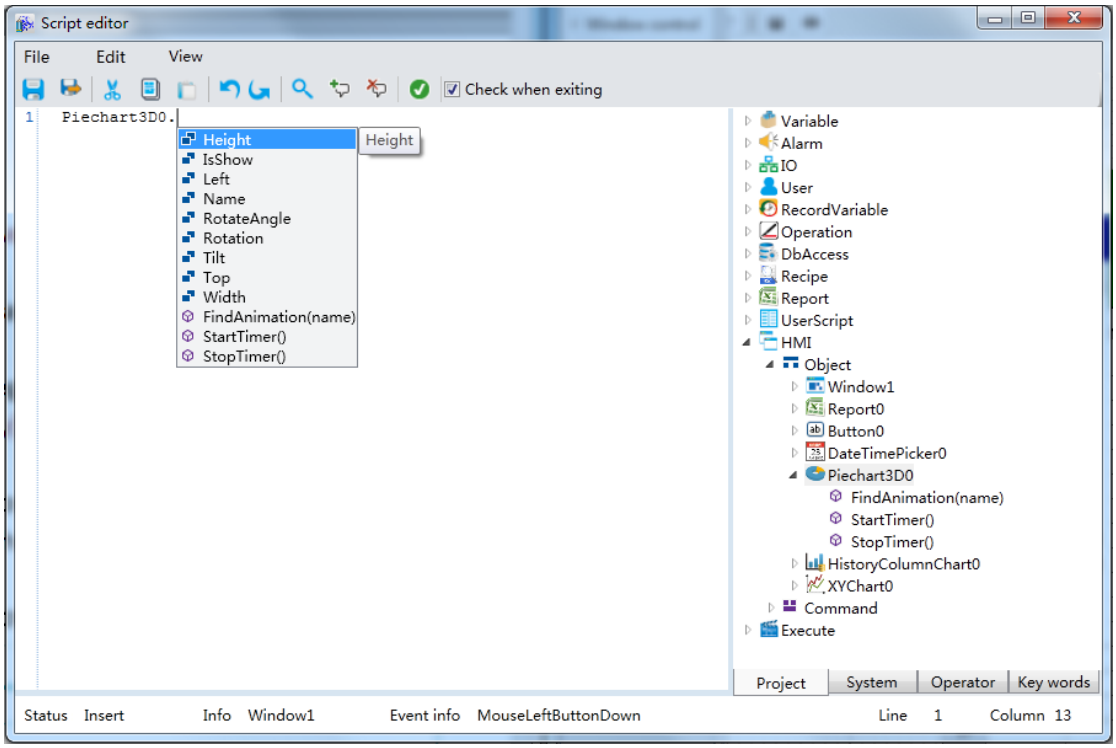
1) Add the new unit data: PieChart0.AddCategory(“3”, 34, “blue) (“3” presents new

unit (category), 34 presents new unit value(value), “blue” presents new unit color(color)).

2) Delete the specified unit: PieChart0.DeleteCategory(4)
3) Print the pie chart: PieChart0.Print()
4) Update the unit data: PieChart0.UpdateCategory(1, ”3”, 800, “blue”)(1 presents the unit index, “3”presents the unit(category), 800 presents the value(value), “blue” presents the color(color).

Pie chart 3D script usage specification:

Click the Pie chart 3D on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of pie chart 3D property script is as follows:

Name	Type	Description
Name	String	Name attribute read only
Tilt	Real	Set 3D pie chart tilt angle
Rotation	Real	Set 3D pie chart rotation angle
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide

Get the name of pie chart 3D: Piechart3D0.Name (read only)
Get or set the tilt angle of pie chart 3D: Piechart3D0.Tilt = 60
Get or set the rotation of pie chart 3D: Piechart3D0.Rotation = 45
Get or set the width of pie chart 3D: Piechart3D0.Width = 100
Get or set the height of pie chart 3D: Piechart3D0.Height = 200

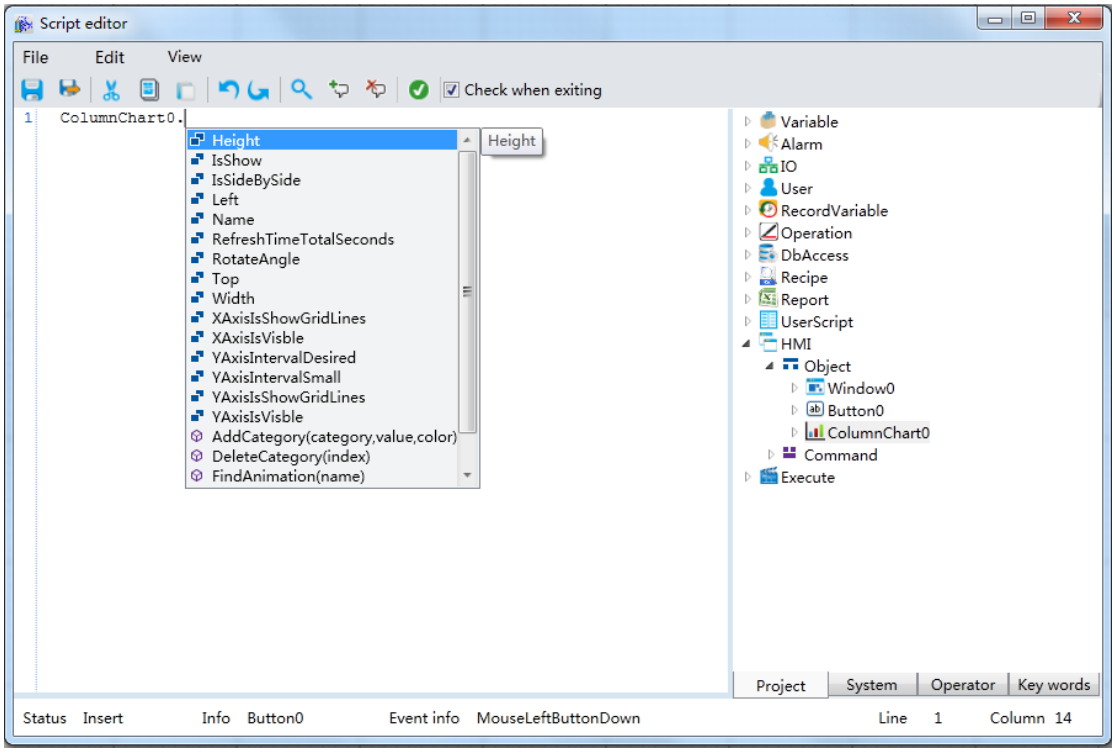
Get or set the left coordinate of pie chart 3D: Piechart3D0.Left = 50
Get or set the top coordinate of pie chart 3D: Piechart3D0.Top = 90
Get or set the display status of pie chart 3D: Piechart3D0.IsShow = True (True is to display, False is to hide)
Get or set the rotation angle of pie chart 3D: Piechart3D0.RotateAngle = 30

Pie chart 3D command usage specification：

Start the timer of pie chart 3D: Piechart3D0.StartTimer()
Stop the timer of pie chart 3D: Piechart3D0.StopTimer()

Column chart script usage specification：

Click the column chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of column chart property script is as follows:

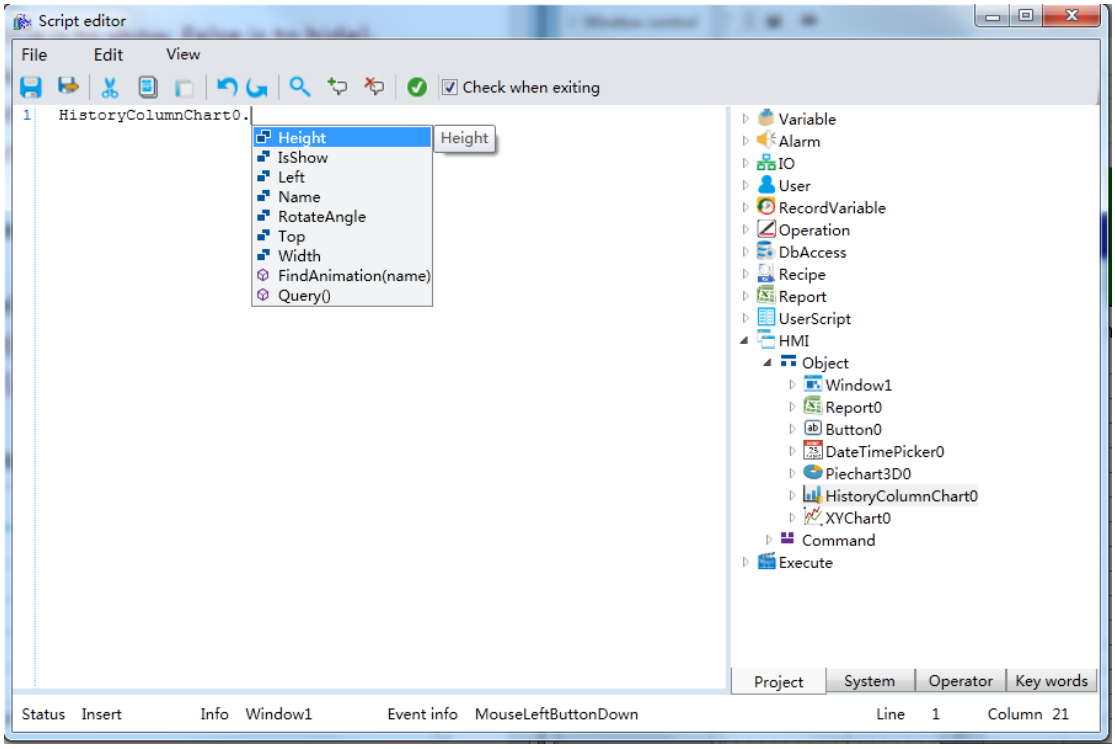
Name	Type	Description
XAxisIsVisble	Discrete	Whether to display X axis,True:display, False: hidden
XAxisIsShowGridLines	Discrete	Whether to display X axis grid,True:display, False: hidden
YAxisIntervalDesired	Real	Y axis large scale degree, int type
YAxisIntervalSmall	Real	Y axis smal scale degree, int type
YAxisIsVisble	Discrete	Whether to display Y axis,True:display, False: hidden
YAxisIsShowGridLines	Discrete	Whether to display Y axis grid,True:display, False: hidden
RefreshTimeTotalSeconds	Integer	Refresh time by default is 1 second
IsSideBySide	Discrete	Whether the left and right display the curve, True: left and right display, False: the merge
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

1) Get the name of column chart: ColumnChart0.Name(read only)
2) Set the width of column chart: ColumnChart0.Width = 200 (read and write)
3) Set the height of column chart: ColumnChart0.Height = 300 (read and write)
4) Set the left coordinate of column chart: ColumnChart0.Left = 10 (read and write)
5) Set the top coordinate of column chart: ColumnChart0.Top = 10 (read and write)
6) Set the display status of column chart: ColumnChart0.IsShow = True (True is to display, False is to hide).
7) Whether to display the column chart side by side: ColumnChart0.IsSideBySide=True
8) (True is to display, False is to hide).
9) Set the refreshing time of column chart: ColumnChart0.RefreshTimeTotalSeconds = 2 (read and write and the default time is one second, the type is integer).
10) Set the visibility of X axis of column chart: ColumnChart0.XAxisIsVisble = True (True is to show, False is to hide)
11) Set the visibility of X axis grid of column chart: ColumnChart0.XAxisIsShowGridLines=True(True is to show, False is to hide)
12) Set the Y axis maximum interval of column chart: ColumnChart0.YAxisIntervaldesire = 3 (read and write).
13) Set the Y axis minimum interval of column chart: ColumnChart0.YAxisIntervalSmall = 2 (read and write).
14) Set the visibility of Y axis of column chart: ColumnChart0.YAxisIsVisble = True (True is to show, False is to hide)
15) Set the visibility of Y axis grid of column chart: ColumnChart0.YAxisIsShowGridLines=True (True is to show, False is to hide)
16) **Column chart command script usage specification：**
17) Add the new unit data:ColumnChart0.AddCategory(“4”, 34, “blue) (“4” presents new unit(category) , 34 presents new unit data value(value), “blue” presents new unit color(blue).

18) Delete the specified unit: ColumnChart0.DeleteCategory(4)
19) Print the pie chart: ColumnChart0.Print()
20) Update the unit data: ColumnChart0.UpdateCategory(1, ”3”, 800, “blue”)(1 presents the unit index, “3”presents the unit(category), 800 presents the value(value), “blue” presents the color(color).

History column chart script usage specification:

Click the history column chart on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of history column chart property script is as follows:

Name	Type	Description
Name	String	Name attribute read only
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide

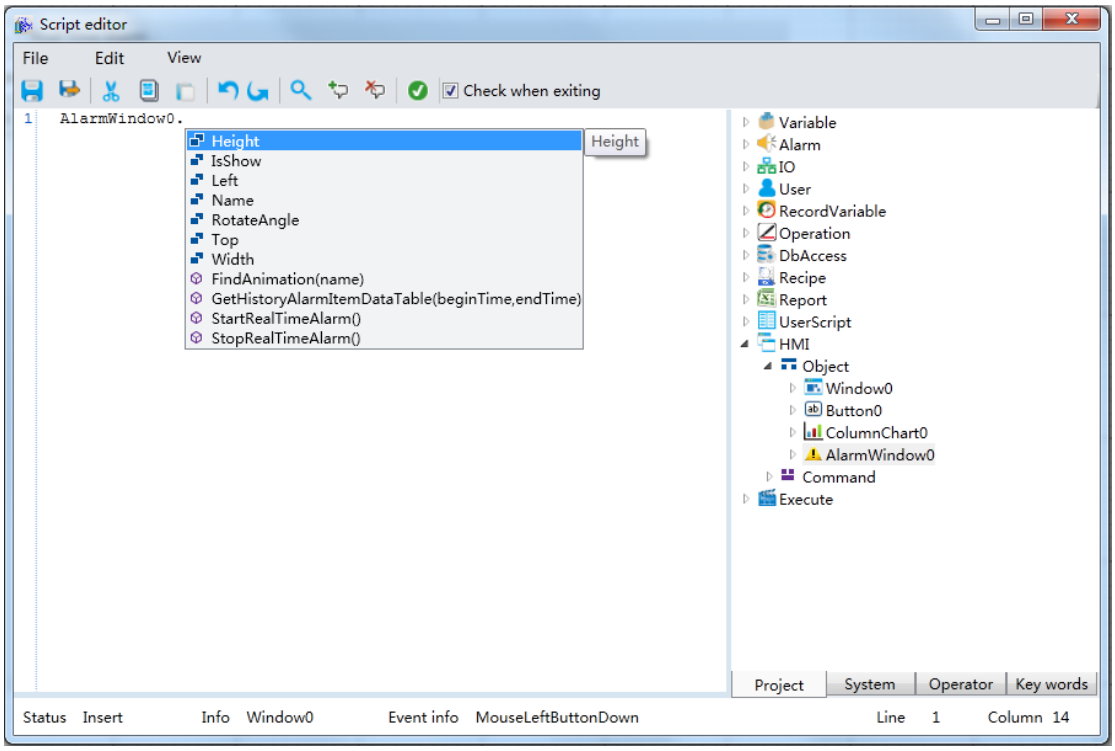
Get the name of history column chart: HistoryColumnChart0.Name(read only)
Get or set the width of history column chart: HistoryColumnChart0.Width = 70
Get or set the height of history column chart: HistoryColumnChart0.Height = 35
Get or set the left coordinate of history column chart: HistoryColumnChart0.Left = 60
Get or set the top coordinate of history column chart: HistoryColumnChart0.Top = 90
Set the display status of history column chart: HistoryColumnChart0.IsShow = True (True is to display, False is to hide)
Set the rotation angle of history column chart: HistoryColumnChart0.RotateAngle = 6

History column chart command script usage specification：

Query the data of history column chart: HistoryColumnChart0.Query()

Alarm window script usage specification：

Click the Alarm Window on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of alarm window property script is as follows:

Name	Type	Description
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

1) Get the name of alarm window: AlarmWindow0.Name(read only)
2) Set the width of alarm window: AlarmWindow0.Width = 200(read and write)
3) Set the height of alarm window: AlarmWindow0.Height = 300(read and write)
4) Set the left coordinate of alarm window: AlarmWindow0.Left = 10(read and write)
5) Set the top coordinate of alarm window: AlarmWindow0.Top = 30(read and write)
6) Set the rotation angle of alarm window: AlarmWindow0.RotateAngle = 60 (read and write)

Alarm window command script usage specification：

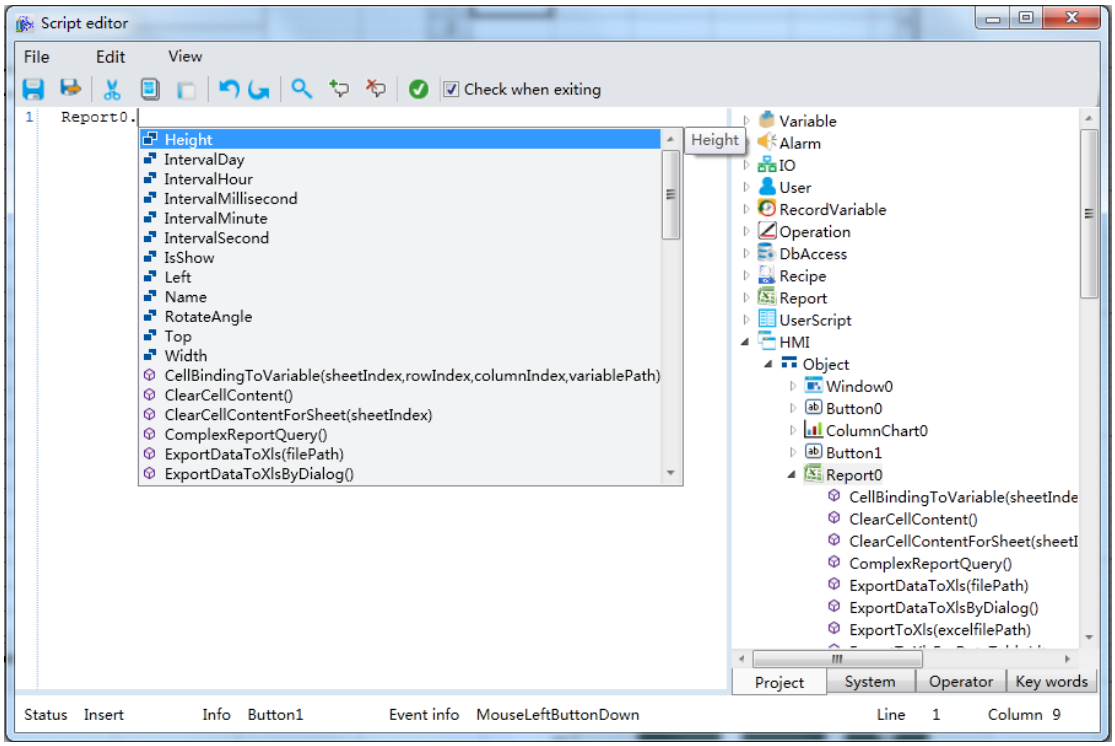
1) Get the history alarm data:

AlarmWindow0.GetHistoryAlarmItemDataTable(“6/17/2016 16:00:00”, "6/17/2016 16:30:00")(“6/17/2016 16:00:00” is start time, "6/17/2016 16:30:00" is end time).

2) Start loading real time alarm: AlarmWindow0.StartRealTimeAlarm()
3) Stop loading real time alarm: AlarmWindow0.StopRealTimeAlarm()

Report script usage specification：

Click the Report on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of report property script is as follows:

Name	Type	Description
Name	String	Name attribute read only
IntervalDay	N/A	Get or Set interval time by days
IntervalHour	N/A	Get or Set interval time by hours
IntervalMinute	N/A	Get or Set interval time by minutes
IntervalSecond	N/A	Get or Set interval time by seconds
IntervalMillisecond	N/A	Get or Set interval time by millonsecond
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide

Get the name of report: Report0.Name (read only)
Set interval day of report query: Report0.Day = 0
Set interval hour of report query: Report0.Day = 0
Set interval minute of report query: Report0.Day = 0

Set interval second of report query: Report0.Day = 1
Set the width of report: Report0.Width = 50
Set the Height of report: Report0.Height = 100
Set the left coordinate of report: Report0.Left = 56
Set the top coordinate of report: Report0.Top = 90
Set the display status of report: Report0.IsShow = True (True is to display, False is to hide)
Set the rotation angle of report: Report0.RotateAngle = 120

Report script command usage specification：

1) Binding cell to variable : Report0.CellBindingToVariable(0, 5, 5, “Var.report”) (0 presents the sheet index, 5 presents the row index, 5 presents the column index, “Var.report” presents the path of variable).
2) Clear all the content: Report0.ClearCellContent ().
3) Clear the specified content: Report0.ClearCellContentForSheet (1)(1 presents the sheet index)
4) Complex report query: Report0.ComplexReportQuery().
5) Common data report: Report0.ExportDataToXls (“D:\ 123.xlsx”)
6) Common data report and file path can be selected: Report0.ExportDataToXlsByDialog ()
7) Export data to Excel: Report0.ExportToXls(“D:\ 1.xlsx”)
8) Export data table to Excel: Report0. ExportToXlsForDataTable (dt, “D:\ 2.xls”, 1) (dt presents data source, “D:\ 2.xls” presents the absolute path, 1 presents the version of Excel, 0 is Excel 2003, 1 is Excel 2007).
9) Get the value of cell: Report0.GetCellValue (5, 7)(5 presents the row index, 7 presents the column index)
10) Get the value of current row: Report0.GetCurrentRowValue (6) (6 presents the column index).
11) Get the statistical data of data table: Report0.GetDataColumnStaticsValue (dt, “increase”, “avg”)(dt presents data source, “increase” presents the column name, “avg” presents the function name).
12) Get the data table from query result: Report0.GetDataTableExcludeHistoryData
13) Fill data table from report control: Report0.GetDataTableFromReport(0)(0 presents having no column name, others presents having column name)
14) Get all templates of current project: Report0.GetProjectReportTemplates()
15) Get data table from specified time: Report0.GetQueryHistoryData(0, “Var.random”, “Value”)(0 presents the sheet index, “Var.random” presents the conditions, “Value” presents the types)
16) Write data table to database: Report0.GetReportDataTable(1)(1 presents the main key)
17) Get the content of key column: Report0.GetSelectId()
18) Hide data table column: Report0.HiddenReportColumns(“name”)(“name” presents the name of column)
19) Import file to report: Report0.ImportDataToControl(“D:\ 2.xls”)
20) Data import by dialog: Report0.ImportDataToControlByDialog()

21) Open current report template: Report0.OpenDataTemplate()
22) Get project path: Report0.ProjectDirectory()
23) Report query alarm data: Report0.QueryAlarmData(1)(1 presents the sheet index)
24) query data from extern database: Report0.QueryDataFromExtern(0,dt)(0 presents sheet index, dt presents data table)
25) Query history data: Report0.QueryHistoryData()
26) Common history data query: Report0.QueryHistoryDataByCommon(0, “Var.hill”, “Value”)
27) History data query for different row count: Report0.QueryHistoryDataOverlay(0, “Var.hike”, “Value”)
28) History data query by variable data: Report0.QueryHistoryDataTimeDifferent(0, “Var.jog”)
29) Query system event data: Report0. QuerySystemEventData(0)(0 presents sheet index)
30) Query operation variables data: Report0. QueryVariableOperations(0,”Operation”)(0 presents sheet index, “Operation” presents variable path).
31) Register variable change to cell: Report0.RegisterVariableToCell(“Var.cell”, 5, 5)(“Var.cell” presents variable path, 5 presents row index, 5 present column index)
32) Edit and save current template: Report0.SaveReportAndOpenToCurrentTemplate()
33) Set active work sheet: Report0.SetActiveWorkSheet(0)
34) Set cell value: Report0.SetCellValue(sad, 5, 7)(23 presents the cell value, 5 presents row index, 7 presents column index)
35) Set column names: Report0.SetColumnNames(5, 6, “Column”)(5 presents start row, 5 presents start column, “Column” presents column name)
36) Set query time of complex report: Report0.SetComplexReportStartTime(0, DateTimePicker0.ValueTime)(0 presents working index, DateTimePicker0.ValueTime presents start time)
37) Set current report template: Report0.SetCurrentReportTemplate(“templateName”)
38) Set start position: Report0.SetQueryDataStartPosition(3, 7, 0)(3 presents start row, 7 presents start column, 0 is to hide column header, others is to show column header)
39) Whether to clear old content: Report0.SetRealTimeClearOld(0)(0 is to clear, 1 is not to clear)
40) Set main variable: Report0.SetRealTimeVariableChange(“Var.change”, 80)(“Var.change” presents the variable path, 80 presents total row count)
41) Set column number of key: Report0.SetTableIndensity(8)(8 presents column index)
42) Set query end time: Report0.SetWorkSheetEndTime(0, DateTimePicker1.ValueTime) (0 presents working sheet,

DateTimePicker1.ValueTime presents end time)

43) Set interval time by millisecond: Report0.SetWorkSheetInternelTime(0, 1000)
44) (0 presents working sheet, 1000 presents 1000 millisecond(1 s))
45) Set query start time:
Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)
46) (0 presents working sheet, DateTimePicker0.ValueTime presents start time)
47) Set query interval time: Report0.SetWorkSheetTotalInternelTime(0)(0 presents working sheet)
48) Show data table to report: Report0.ShowDataTableForReport(1, 1 , dt)(1 presents row index, 1 presents column index, dt presents data table)
49) Start update data: Report0.StartUpdateDataFromRt(1)(1 is to display multiple lines)
50) Stop real time variable change: Report0.StopRealTimeVariable()
51) Unregister event: Report0.UnBindingEventChange(0)(0 presents sheet index)

The report usage examples：

1. **Bind real time data and create real time report(Value change will display on cells)**

Script: Report0.StartUpdateDataFromRT (1)
Call Report0.SetRealTimeVariableChange ("Var.increase", 10)
Report0.SetRealTimeClearOld (0) ' whether to clear the existed data, 1 is to clear, 0 is not)

2. **Stop binding current data**

Script: Report0.StopRealTimeVariableChange()

3. **Bind real time data, data change in one cell**

Script: Report0.StartUpdateDataFromRT(0)

4. **Register variables to cells**

Script: Report0.SetActiveWorkSheet(1)
```
for i = 1 to 10
   call Report0.RegisterVariableToCell("Var.binding"&i,4,i)
Next
```

5. **Bind variables and cells**

Script: Report0.SetActiveWorkSheet(1)
```
For i = 1 to 10
   Call Report0.RegisterVariableToCell(1,3,i,"Var.binding"&i)
Next
```

6. **Set the content of one cell**

Script: Report0.SetActiveWorkSheet(1)
Call Report0.SetCellValue("NewValue",2,8)

7. Set the column name of report in specified cells

Script: Report0.SetActiveWorkSheet(1)
Call Report.SetColumnNames(1,2,"Value,Time,Age,Name")

8. **Query history data**

Script: Call Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)

```
Call Report0.SetWorkSheetEndTime(0,DateTimePicker1.ValueTime)
Report0.IntervalDay = 0
Report0.IntervalHour = 0
Report0.IntervalMinute =0
Report0.IntervalSecond = 1
Report0.IntervalMillisecond = 0
Report0.SetWorkSheetTotalInternelTime(0)
Report0.QueryHistoryData()
```

9. Query the maximum value of variables

Script:

```
Call Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)
Call Report0.SetWorkSheetEndTime(0,DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
dt = Report0.GetQueryHistoryData(0,"Var.increase", "1")
Call
Rport0.SetCellValue(Report0.GetDataColumnStaticsValue(dt,"increase","max"),1,7)
```

10. Query the minimum value of variables

Script:

```
Call Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)
Call Report0.SetWorkSheetEndTime(0,DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
dt = Report0.GetQueryHistoryData(0,"Var.increase", "1")
Call
Rport0.SetCellValue(Report0.GetDataColumnStaticsValue(dt,"increase","min"),1,8)
```

11. Query the average value of variables

Script:

```
Call Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)
Call Report0.SetWorkSheetEndTime(0,DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
dt = Report0.GetQueryHistoryData(0,"Var.increase", "1")
Call
Rport0.SetCellValue(Report0.GetDataColumnStaticsValue(dt,"increase","avg"),1,9)
```

12. Query the sum of variables

Script: Call Report0.SetWorkSheetStartTime(0,DateTimePicker0.ValueTime)
Call Report0.SetWorkSheetEndTime(0,DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
dt = Report0.GetQueryHistoryData(0,"Var.increase", "1")
Call
Rport0.SetCellValue(Report0.GetDataColumnStaticsValue(dt,"increase","sum"),1,10)

13. Filter query

Way1: Query by report instruction

Script:

Call Report0.SetWorkSheetStartTime(0,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(0, DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
Call Report0.SetQueryDataStartPosition(3,4,1)
(0 is to hide column head, more than 0 is not)
Call Report0.QueryHistoryDataByCommon
(0,"Var.increase,Var.decrease,Var.random,Var.triangle,Var.sine,Var.sine","1,1,1,1,1,0")
(1 is to query)

Way2: Variable overlay query

Script:

Call Report0.SetWorkSheetStartTime(0,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(0, DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
Call Report0.SetQueryDataStartPosition(3,4,0)
'(0 is to hide column head, more than 0 is not)
Call
Report0.QueryHistoryDataOverlay(0,"Var.increase,Var.decrease,Var.random,Var.triangle,Var.sine,Var.sine","1,1,1,1,1,0") '(1 presents that the query value is Value,0 presents that the query value is TriggerTime)

Way: Query by data table

Script:

```
Call Report0.SetWorkSheetStartTime(0,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(0, DateTimePicker1.ValueTime)
Report0.IntervalDay = 0 'day
Report0.IntervalHour = 0 'hour
Report0.IntervalMinute =0 'minute
Report0.IntervalSecond = 1 'second
Report0.IntervalMillisecond = 0 'millisecond
Report0.SetWorkSheetTotalInternelTime(0)
dt = Report0.GetQueryHistoryData
(0," Var.increase,Var.decrease,Var.random,Var.triangle,Var.sine,Var.sine","1,1,1,1,1,0")
'(Query data save in the data table )
Call Report0.ShowDataTableForReport(1,1,dt)
```

14. Hide the query data columns (only hide the data query by common)

Script: call Report0.HiddenReportColumns("increase, decrease")

15. display the query data columns

Script: call Report0.VisiableReportColumns("increase, decrease")

16. Bind external database and query the value

Script: dt = DbAccess.DatabaseAccess.GetTable()

```
Call Report0.QueryDataFromExtern(2, dt)
```

17. Query alarm data

Script:

```
Call Report0.SetActiveWorkSheet(1)
Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)
Call Report0.QueryAlarmData(1)
'(1 presents that the query value is Value,0 presents that the query value is
TriggerTime )
```

18. Query system events

Script:

```
Call Report0.SetActiveWorkSheet(1)
Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)
Call Report0.QueryEventData(1)
'(1 presents that the query value is Value,0 presents that the query value is
TriggerTime )
```

19. Query root group of operation variables

Script:

```
Call Report0.SetActiveWorkSheet(1)
Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)
Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)
Call Report0.QueryVariableOperations(1,"Operation")
```

20. Query OP1 of operation variables

Script:

Call Report0.SetActiveWorkSheet(1)

Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)

Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)

Call Report0.QueryVariableOperations(1,”Operation.op1”)

21. Query OP1 of operation variables

Script:

Call Report0.SetActiveWorkSheet(1)

Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)

Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)

Call Report0.QueryVariableOperations(1,”Operation.op2”)

22. Query all the operation variables

Script:

Call Report0.SetActiveWorkSheet(1)

Call Report0.SetWorkSheetStartTime(1,DateTimePicker 0.ValueTime)

Call Report0.SetWorkSheetEndTime(1, DateTimePicker1.ValueTime)

Call Report0.QueryVariableOperations(1,””)

23. complex report query (query various variables and values in one time)

Script: call Report0.SetComlexReportStartTime(0, DateTimePicker1.ValueTime**)**

Report0.ComplexReportQuery()

24. Get the cell value of report

Script: Button19.Content = Report0.GetCellValue(2,3)

25. Get the value of current row and specified column

Script: Button20.Content = Report0.GetCurrentRowValue(2)

26. Get the current row Id of report

Script: Button21.Content = Report0.GetSelectId()

27. Get the project directory of report

Script: Button22.Content = Report1.ProjectDirectory()

28. Get the current templates of report

Script: Button23.Content = Report1.GetProjectReportTemplates()

29. Export data to Excel(directory is optional)

Script: Call Report0.ExportDataToXlsByDialog()

30. Export data to Excel

Way1: Export by report property

Script1: Report0.ExportDataToXls(“E:\historyData.xlsx”)---only can export current page data

Script2: Report0.ExportToXls(“E:\HistoryDataIn.xlsx”)---can export specified time range history data, take the paging query method.(preferred method)

Script3: dt = Report0.GetDataTableExcludeHistoryData()

Label0.Text = DbAccessCmd.GetDataTableContent(dt)

Call Report0.ExportToXlsForDataTable(dt, ”E:\HistoryDataBase.xlsx”,1)---Export data query by data table

Way2: Export by command script

Script1: dt = Report0.GetDataTableFromReport(1)

Call Report.ExportToXlsForDataTable(dt, “E:\123.xls”,0) ’0 presents excel 2003, 1 presents excel 2007.

Script2: dt = DbAccessCmd.DatabaseAccess.GetTable()
Label9.Text = DbAccessCmd.GetDataTableContent(dt)
Call Report0.ExportToXlsForDataTable(dt, “E:\123.xls”,0) ’0 presents excel 2003, 1 presents excel 2007.

31. Export history data to excel by templates

Script: call ReportCmd.SetWorkSheetStartTime(0, DateTimePicker0.ValueTime)
Call ReportCmd.SetWorkSheetEndTime(0, DateTimePicker1.ValueTime)
call ReportCmd.SetWorkSheetInternelTime(0,1000)
call ReportCmd.ExportHistoryDataByTemplate(“HistoryReport”,”E:\EXREPORT.xls”) ’ HistoryReport presents the name of templates.

32. Import data to report(directory is optional)

Script: call Report0.ImportDataToControlByDialog()

33. Import data to report

Way1: Import external file to data table and display on the report
Script: dt = ReportCmd.DirectImportToDataTable(“E:\EXREPORT.xlsx”)
call Report0.ShowDataTableForReport(1,1,dt)

Way2: Import external file to report directly
Script: Report0.ImportDataToControl(“F:\project\table-20160321.xlsx”)

34. Turn the content of report into data table and print it
Script: dt = ReportCmd.DirectImportToDataTable(“E:\EXREPORT.xlsx”)
ReportCmd.DirectPrintDataTable(dt,8,0,0,0,0,1,100) ’ 1 is vertical, 2 is horizontal

35. Print the templates of report
Script: call ReportCmd.DirectPrintTemplateForPath (“E:\HistoryDataBase.xlsx”,8,0,0,0,0,1,100)

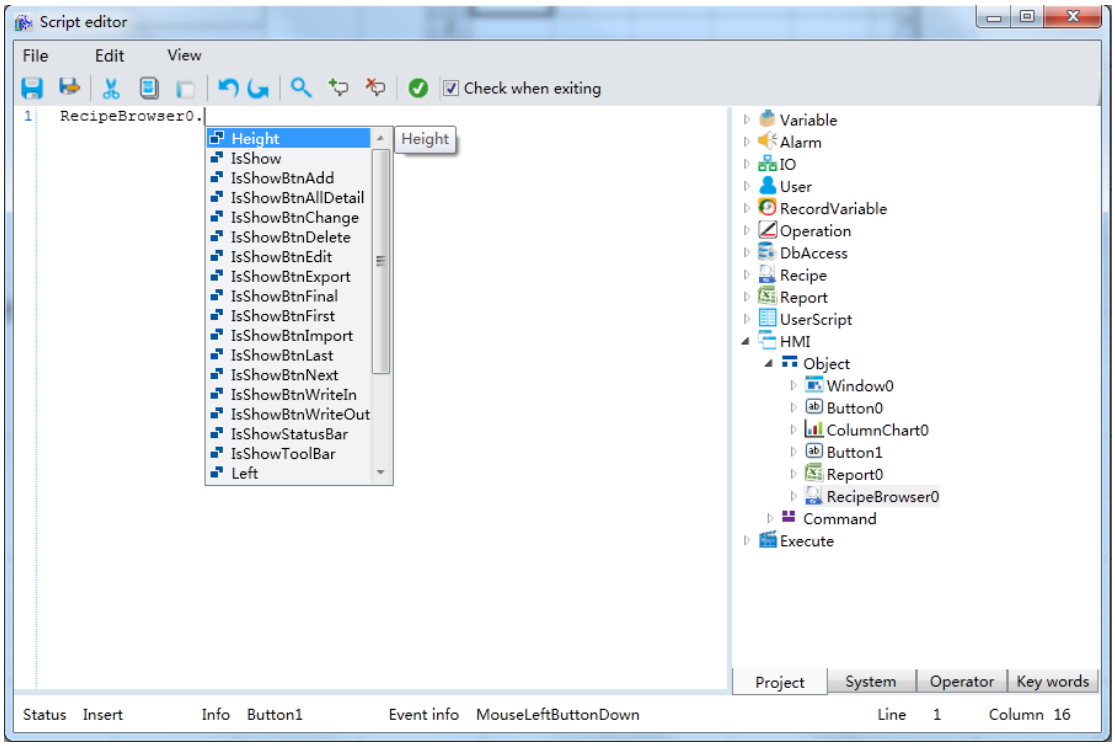
36. Set current templates of report online

Script: Report1.SetCurrentReportTemplate(“ComplexReport”)
Report1.OpenDataTemplate()

37. modify the open template and save it
Script: Report1.SaveReportAndOpenToCurrentTemplates()

Recipe browser script usage specification：

Click the Recipe Browser on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of recipe browser property script is as follows:

Name	Type	Description
RecipeName	String	Recipe Name
IsShowToolBar	Discrete	Set is show toolbar
IsShowBtnAllDetail	Discrete	Set is show all|detail button
IsShowBtnAdd	Discrete	Set is show adding button
IsShowBtnEdit	Discrete	Set is show editing button
IsShowBtnDelete	Discrete	Set is show delete button
IsShowBtnFirst	Discrete	Set is show first button
IsShowBtnNext	Discrete	Set is show next button
IsShowBtnLast	Discrete	Set is show previous button
IsShowBtnFinal	Discrete	Set is show last button
IsShowBtnImport	Discrete	Set is show importing button
IsShowBtnExport	Discrete	Set is show exporting button
IsShowBtnWriteIn	Discrete	Set is show writein button
IsShowBtnWriteOut	Discrete	Set is show writeout button
IsShowBtnChange	Discrete	Set is show change button
IsShowStatusBar	Discrete	Set is show statusbar
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width

1) Set the height of recipe browser: RecipeBrowser0.Height = 100
2) Set the display status of recipe browser: RecipeBrowser0.IsShow = True (True is to show, False is to hide)
3) Set the visibility of adding button of recipe browser: RecipeBrowser0.IsShowBtnAdd = True
4) Set the visibility of detail button of recipe browser:

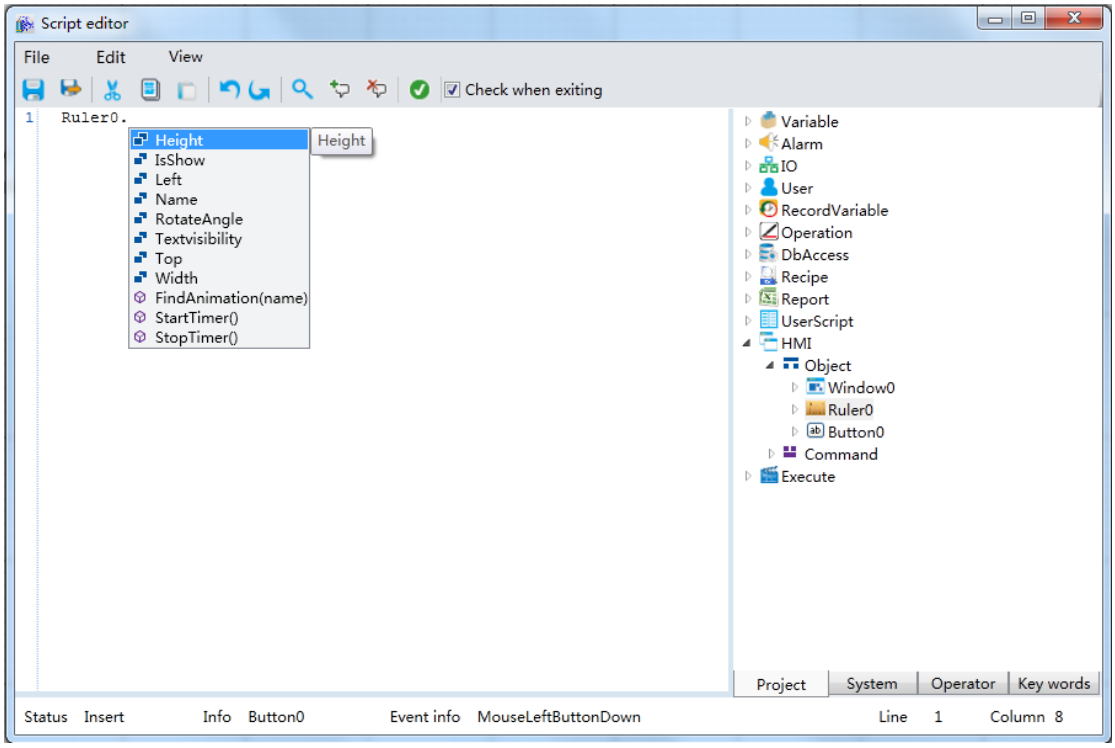
RecipeBrowser0.IsShowBtnAllDetail = True

5) Set the visibility of changing button of recipe browser:
   RecipeBrowser0.IsShowBtnChange = True
6) Set the visibility of deleting button of recipe browser:
   RecipeBrowser0.IsShowBtnDelete = True
7) Set the visibility of editing button of recipe browser:
   RecipeBrowser0.IsShowBtnEdit= True
8) Set the visibility of exporting button of recipe browser:
   RecipeBrowser0.IsShowBtnExport = True
9) Set the visibility of last button of recipe browser:
   RecipeBrowser0.IsShowBtnFinal = True
10) Set the visibility of first button of recipe browser:
    RecipeBrowser0.IsShowBtnFirst = True
11) Set the visibility of importing button of recipe browser:
    RecipeBrowser0.IsShowBtnImport = True
12) Set the visibility of previous button of recipe browser:
    RecipeBrowser0.IsShowBtnLast = True
13) Set the visibility of next button of recipe browser:
    RecipeBrowser0.IsShowBtnNext = True
14) Set the visibility of write-in button of recipe browser:
    RecipeBrowser0.IsShowBtnWriteIn = True
15) Set the visibility of write-out button of recipe browser:
    RecipeBrowser0.IsShowBtnWriteOut = True
16) Set the visibility of status bar of recipe browser:
    RecipeBrowser0.IsShowStatusBar = True
17) Set the visibility of tool bat of recipe browser: RecipeBrowser0.IsShowToolBar = True
18) Get the name of recipe browser: RecipeBrowser0.Name(read only).
19) Set the top coordinate of recipe browser: RecipeBrowser0.Top = 50
20) Set the left coordinate of recipe browser: RecipeBrowser0.Left = 150
21) Set the width of recipe browser: RecipeBrowser0.Width = 60
22) Set the recipe name of recipe browser: RecipeBrowser0.RecipeName = “Pie” (String)
23) Set the rotation angle of recipe browser: RecipeBrowser0.RotateAngle= 60

Recipe browser command script usage specification：

Refresh the content of recipe browser: RecipeBrowser0.Refresh().

Ruler script usage specification:



The specific meaning of ruler property script is as follows:

Name	Type	Description
Textvisibility	Discrete	StautBarVisibility
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location
RotateAngle	Real	Rotate Angle

Set the visibility of status bar of ruler: Ruler0.Textvisibility = True (True is to show, False is to hide)

Get the name of ruler: Ruler0.Name

Set the display status of ruler: Ruler0.IsShow = True (True is to show, False is to hide)

Set or get the width of ruler: Ruler0.Width = 70

Set or get the height of ruler: Ruler0.Height = 50

Set or get the left coordinate of ruler: Ruler0.Left = 66

Set or get the top coordinate of ruler: Ruler0.Top = 54

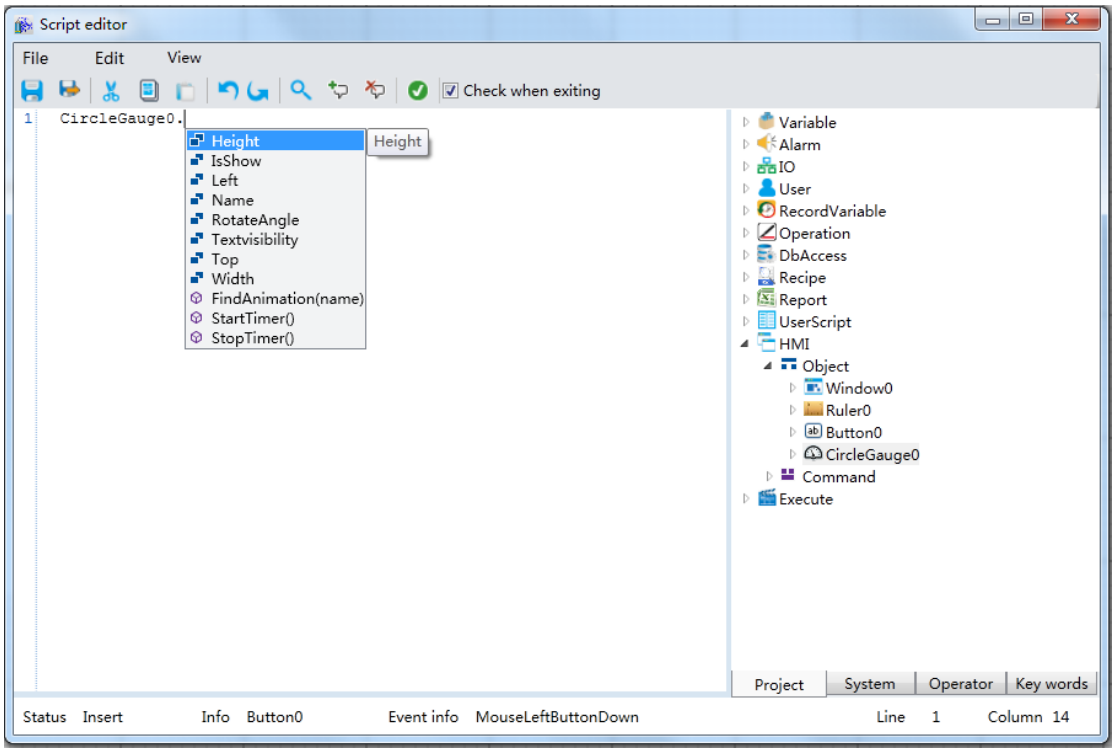
Set or get the rotation angle of ruler: Ruler0.RotateAngle = 70

Ruler command script usage specification：

Start the timer of ruler: Ruler0.StartTimer ()

Stop the timer of ruler: Ruler0.StopTimer ()

Circle gauge script usage specification:



The specific meaning of circle gauge property script is as follows:

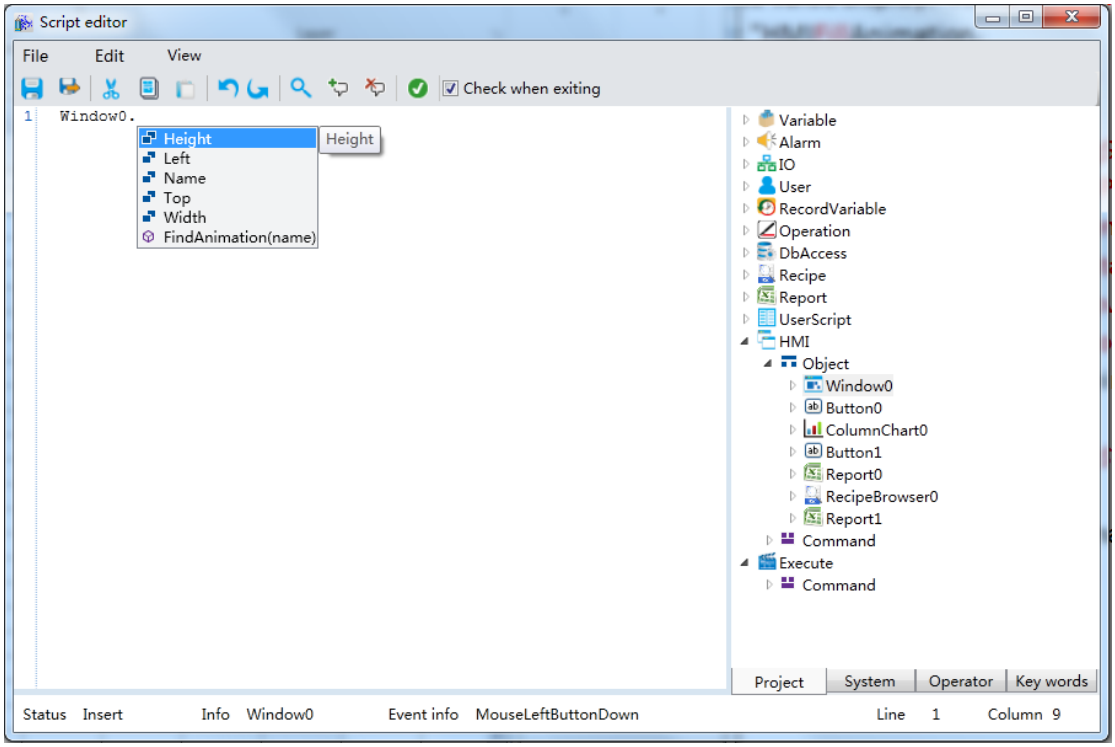
Name	Type	Description
Textvisibility	Discrete	StautBarVisibility
Name	String	Name attribute read only
IsShow	Discrete	The display parameters for the specified object: True: display, False: hide
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location
RotateAngle	Real	Rotate Angle

Set the visibility of status bar of circle gauge: CircleGauge0.Textvisibility = True (True is to show, False is to hide)
Get the name of circle gauge: CircleGauge0.Name
Set the display status of circle gauge: CircleGauge0.IsShow = True (True is to show, False is to hide)
Set or get the width of circle gauge: CircleGauge0.Width = 70
Set or get the height of circle gauge: CircleGauge0.Height = 50
Set or get the left coordinate of circle gauge: CircleGauge0.Left = 66
Set or get the top coordinate of circle gauge: CircleGauge0.Top = 54
Set or get the rotation angle of circle gauge: CircleGauge0.RotateAngle = 70
Ruler command script usage specification：
Start the timer of circle gauge: CircleGauge0.StartTimer ()
Stop the timer of circle gauge: CircleGauge0.StopTimer ()

All the controls have one same command script: .FindAnimation. The specific usage is as follows:

1. Text1.FindAnimation(“HMITextAnimation”).Expression = ”Var.NewVariable1”
2. Analog value display: “HMIAnalogValueDisplayAnimation”
3. Analog value string: “HMIAnalogValueStringAnimation”
4. Discrete value display: “HMIDiscreteValueDisplayAnimation”
5. Brush: "HMIFillAnimation
6. Paintbrush: "HMIStrokeAnimation"
7. Flowing: "HMIFlowingAnimation"
8. Start and stop flowing: "HMIFlowingStartStopAnimation"
9. Horizontal filling: "HMIHorizontalFillPercentAnimation"
10. Vertical filling: "HMIHorizontalMovementAnimation"
11. Horizontal movement: "HMIHorizontalMovementAnimation"
12. Vertical movement: "HMIVerticalMovementAnimation"
13. Horizontal Zoom: "HMIHorizontalZoomAnimation"
14. Vertical Zoom: "HMIVerticalZoomAnimation"
15. Rotation: "HMIRotateAnimation"
16. Rotation start and stop: *"HMIRotateStartStopAnimation"*
17. Blink: "HMIBlinkAnimation"
18. Show and hide: "HMIShowHideAnimation"

Window script usage specification：

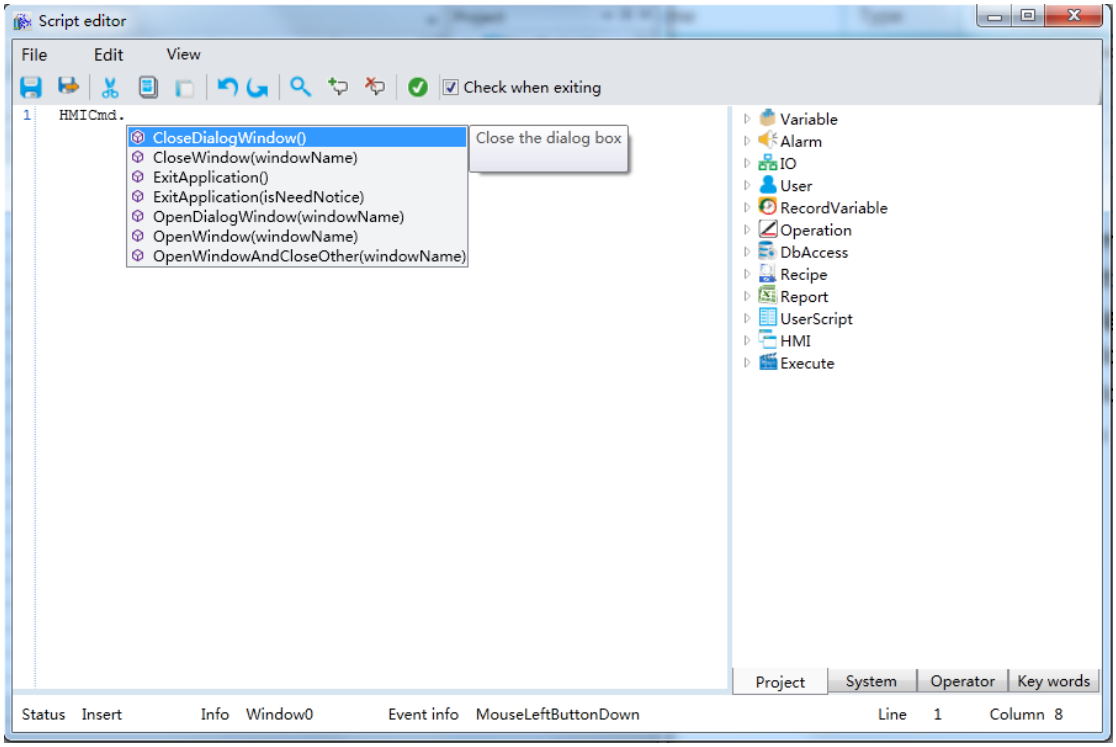


The specific meaning of window property script is as follows:

Name	Type	Description
Name	String	Name attribute read only
Width	Real	Width
Height	Real	Height
Left	Real	Left Location
Top	Real	Upper Location

Get the name of window: Window0.Name(read only)
Set the width of window: Window0.Width = 100 (read and write)
Set the height of window: Window0.Height = 00 (read and write)
Set the left coordinate of window: Window0.Left = 60 (read and write)
Set the top coordinate of window: Window0.Top = 40 (read and write)

Window command usage specification：



Close the dialog box: HMICmd.CloseDialogWindow()
Close the window: HMICmd.CloseWindow()
Exit procedures: HMiCmd.ExitApplication()
Exit procedures: HMiCmd.ExitApplication(True)(True is to show notice, False is to hide notice)
Open the window of the dialog box: HMICmd.OpenDialogWindow(“window0”)
Open current window and close others: HMICmd.OpenWindowAndClodeOther (“window1”)

User script usage specification:

User script is background script, it has two parts: time script and condition script.
Time script usage specification is as follows:

Name	Type	Description
StartTime	Date	Start time
IsEnableEndTime	Discrete	Whether the use of the end time
EndTime	Date	End time
WeekDays	String	Week date, comma separated, shaped like"0,1,2,3", 0-6 is representatives from Saturday
MonthDays	String	The date of the month, with a comma separated, shaped like"1,20,22,25"
UniqueIdentifier	Integer	The only marked ID, read only
Name	String	User script name, read only
IntTriggerMode	Integer	Trigger mode, integer typeValue ranges are as follows:<=0: value change, 1: while true, 2
Interval	Integer	Time interval, unit ms
ScriptContent	String	Script content
IsEnable	Discrete	Enable script
Description	String	Description

Get or set the start time: Script.TimeScript.StartTime = “6/21/2016 13:30:00” (“6/21/2016 13:30:00” presents the start time, DateTimePicker0.Value is OK)
Whether to use end time: Script.TimeScript.IsEnableEndTime = True (True is to use, False is not)
Get or set the end time: Script.TimeScript.EndTime = “6/21/2016 14:30:00” (“6/21/2016 14:30:00” presents the end time, DateTimePicker0.Value is OK)
Get or set the week date: Script.TimeScript.WeekDays = 5 (0~6 presents Sunday to Saturday respectively)
Get or set the date of the month: Script.TimeScript.MonthDays = 15 (0~31 presents the date of the month respectively)
Get the only ID of time script: Script.TimeScript.UniqueIdentifier (read only)
Get the name of time script: Script.TimeScript.Name (read only)
Get or set the trigger mode: Script.TimeScript.IntTriggerMode = 6 (the value is more than 4, 5 is to start, 6 is to stop, 7 is one time, 8 is daily, 9 is weekly, more than 10 is monthly)
Get or set the interval of time script: Script.TimeScript.Interval = 1000 (default time is 1000, unit is millisecond)
Get or set the script content of time script: Script.TimeScript.ScriptContent = “Var.TimeScript.Operation4 = Var.TimeScript.Operation4 +1” (Script supports the replacement of script content, the specific writing is as follows:

```
DIM CONTENT
CONTENT = Script.TimeScript.ScriptContent
CONTENT=Replace(CONTENT,“Var.TimeScript.Operation2”, ”Var.TimeScript.Operation 4”)
Script.TimeScript.ScriptContent =CONTENT
```

Above method only replace the writing of variable).
Whether to enable script of time script: Script.TimeScript.IsShow = True (True is to start, False is to stop)
Get or set the description of time script: Script.TimeScript.Description = “TimeScript”
Condition script usage specification is as follows:

Name	Type	Description
Expression	String	Expression
UniqueIdentifier	Integer	The only marked ID, read only
Name	String	User script name, read only
IntTriggerMode	Integer	Trigger mode, integer typeValue ranges are as follows:<=0: value change, 1: while true, 2
Interval	Integer	Time interval, unit ms
ScriptContent	String	Script content
IsEnable	Discrete	Enable script
Description	String	Description

Get or set the expression of condition script: Script.ConditionScript.Expression = “Var.Variable0 =2”

Get the only ID of condition script: Script.ConditionScript.UniqueIdentifier (read only)

Get the name of condition script: Script.ConditionScript.Name (read only)

Get or set the trigger mode of condition script: Script.ConditionScript.IntTriggerMode = 5(0~6 presents Sunday to Saturday respectively)

Get or set the interval of condition script: Script.ConditionScript.Interval = 1000 (default time is 1000, unit is millisecond)

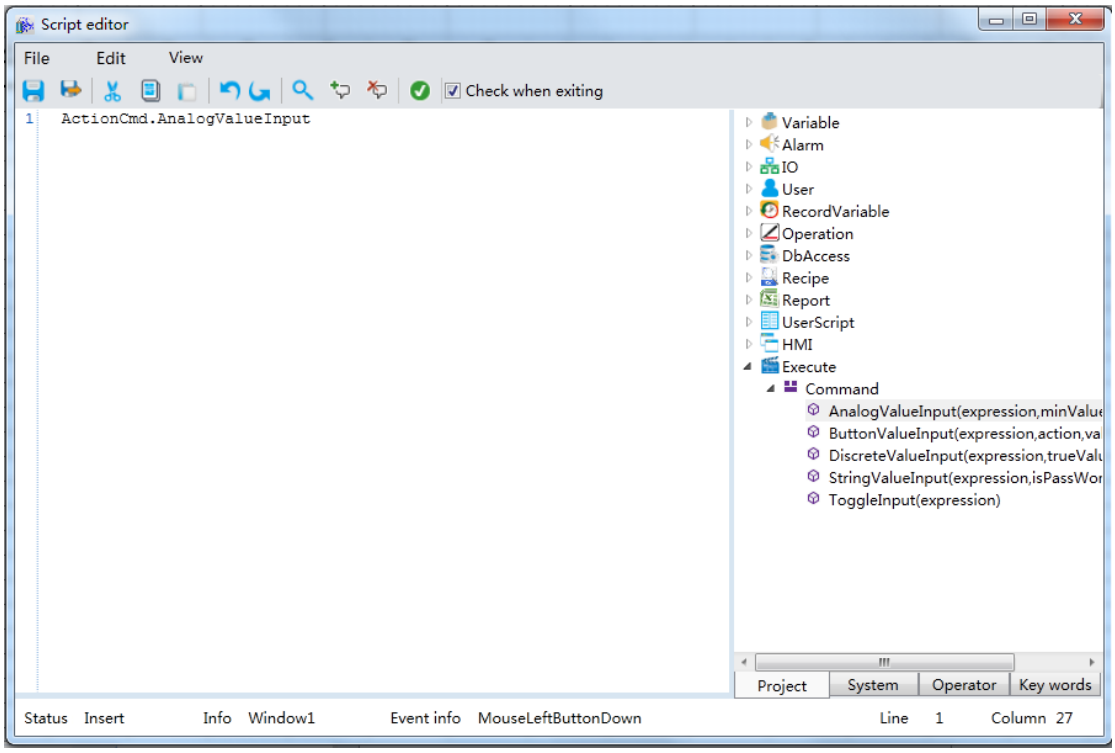
Get or set the script content of condition script: Script.ConditionScript.ScriptContent =”Var.Variable2 = Var.Variable1+2” (the same usage as time script).

Whether to enable script of condition script: Script.ConditionScript.IsShow = True (True is to start, False is to stop)

Get or set the description of time script: Script.TimeScript.Description = “ConditionScript”

Executing script usage specification:

Click “Execute” on the right side of script editor, choose and edit the corresponding command in the script editor as shown in the figure:



The specific meaning of execution command script is as follows:

Analog value input: ActionCmd.AnalogValueInput (“Var.Variable1”, 0, 100) (when the command is executed, analog input dialog will be popped up, the maximum value is 100, the minimum value is 0, the result saves in Var.Variable1)

Button input: ActionCmd.ButtonValueInput (“Var.Variable1”, 1, 10) (“Var.Variable1” is the operation variable, 1 is the operation type: 0-invert, 1-increase, 2-decrease, 3-mutiply, 4-divide; 10 is the operation value).

Discrete value input: ActionCmd.DiscreteValueInput (“Var.Variable2”, “open”,” close”) (when the command is executed, discrete input dialog will be popped up, the left side is “open”, the right side is “close”, the result saves in Var.Variable2).

String value input: ActionCmd.DiscreteValueInput (“Var.String”, False) (when the command is executed, string input dialog will be popped up, the result saves in Var.String, True is not password input , False is password input).

Toggle input:ActionCmd.ToggleInput(“Var.change”) (when the command is executed, toggle the value of Var.change, that is 1 changes into 0, 0 changes into none zero, True changes into False etc.)