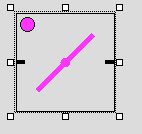
**Flap.cs**



ИСПОЛЬЗУЕМЫЕ БИБЛИОТЕКИ И КЛАССЫ

В этом разделе объявляются использованные библиотеки и пространства имен, подключенные файлу Flap.cs.

using ecl.commonClasses.BaseIm.ControlWnd;  
using ecl.commonClasses.VentVs;  
using ecl.visualControls.DiskretBox;  
using FB;  
using FB.VisualFB;  
using InSAT.Library.Interop;  
using System;  
using System.ComponentModel;  
using System.Runtime.InteropServices;

Помимо стандартной библиотеки System, подключены библиотеки классов, такие как: ecl.commonClasses.BaseIm.ControlWnd (из которой в дальнейшем используется метод SpecialCommandDescriptor) , ecl.commonClasses.VentVs (которая в дальнейшем используется наследником Flap), ecl.visualControls.DiskretBox (в дальнейшем используется тип контрола DiscreteIndicatorControl).

Пространство имен System.ComponentModel предоставляет классы, используемые для реализации поведения компонентов и элементов управления во время разработки и во время выполнения, в данном случае используется класс DisplayNameAttribute, для отображения имени свойства.

Пространство имен System.Runtime.InteropServices предоставляет разнообразные члены, поддерживающие COM-взаимодействие и службы вызова платформы, в данном случае это пространство имен используется для регистрации FB, как COM-объекта.

Пространство имен InSAT.Library.Interop используется для атрибута CatID.

Пространство имен FB.VisualFB и FB используются для атрибута VisualControls, который предназначен для создания связи между ФБ и контроллами.

ОПИСАНИЕ ФБ

[Serializable,  
ComVisible(true),  
Guid("73C7AEAE-1D37-45F0-A1B4-6F8476D6E92A"),  
CatID(CatIDs.**CATID\_OTHER**),  
DisplayName("Заслонка"),  
VisualControls(typeof(FlapControl), typeof(DiscreteIndicatorControl))  
]

Serializable - атрибут, определяющий метод сохранения ФБ в проекте MasterSCADA, для бинарного сохранения ФБ.

ComVisible и Guid - атрибуты для регистрации ФБ как COM объекта. Значение атрибута Guid уникально.  
 CatID и DisplayName - атрибуты для регистрации нашего класса как ФБ. Атрибут CatID принимает перечисление CatIDs и задает категории ФБ в которых он появиться в Палитре (у ФБ может быть несколько атрибутов CatID). После обновления палитры ФБ он будет добавлен в категории, которые вы указали с помощью атрибутов CatID.

VisualControls(typeof(FlapControl), typeof(DiscreteIndicatorControl)) – атрибут (VisualConrols) для создания связи между ФБ и контролами.

КЛАССЫ И МЕТОДЫ

public class Flap : VentVs

{

public override SpecialCommandDescriptor[] getSpecialCommands()

{

return new SpecialCommandDescriptor[]

{

new SpecialCommandDescriptor(Commands.CMDOPEN,"Открыть",true),

new SpecialCommandDescriptor(Commands.CMDCLOSE,"Закрыть",true)

};

}

}

Описываемый класс Flap наследуется от класса VentVs и содержит перегружаемый метод SpecialCommandDescriptor, у которого в свою очередь имеются команды «Открыть» ((Commands.CMDOPEN,"Открыть",true)) и «Закрыть» ((Commands.CMDCLOSE,"Закрыть",true)). Константа CMDOPEN равна 1, а константа CMDCLOSE равна 2.

**Flap.xml**

<Map>

<Items>

<!-- Стандартная часть-->

<Pin ID="1" Name="Status" Type="System.UInt32"/>

<Pout ID="2" Name="Command" Type="System.UInt16"/>

<Group Name="Пользовательские выходы" ID="101">

<Items>

<Pout ID="1010" Name="Бит 15" Type="Логический"/>

<Pout ID="1011" Name="Бит 16" Type="Логический"/>

<Pout ID="1012" Name="Бит 17" Type="Логический"/>

<Pout ID="1013" Name="Бит 18" Type="Логический"/>

<Pout ID="1014" Name="Бит 19" Type="Логический"/>

<Pout ID="1015" Name="Бит 20" Type="Логический"/>

</Items>

</Group>

<Group Name="Пользовательские Команды" ID="1100">

<Items>

</Items>

</Group>

<Pout ID="1110" Name="Имя" Type="Строковый"/>

<Pout ID="1016" Name="PreparingCommand" Type="Логический"/>

<!-- /Стандартная часть-->

</Items>

</Map>

Структура элемента <Map>…</Map> описывает структуру входов выходов ФБ. У всех элементов кроме <Items> и <ItemsRange> не обходимо указывать уникальный ID. У элемента <Map>может быть только один из элементов <Items> или <ItemsRange>.

<Pin/> - описание входа ФБ, в его структуре описывается уникальный номер (ID), имя (Name) и тип (Type). <Pout/> - описание выхода ФБ, его структура идентична <Pin/>. В данном случае на входе 1, а на выходе 2.

<Group> - описание группы ФБ, в данном случае это описание пользовательских выходов и пользовательских команд. Структура элемента <Group> полностью идентична структуре элемента < Map > , за исключением дополнительных атрибутов, таких как Name – имя группы и ID – уникальный номер группы.

<VisualMap>

<Items>

<!-- Стандартная часть-->

<Pout ID="1001" Name="Авария:e" Type="Логический"/>

<Pout ID="1002" Name="Неисправность:w" Type="Логический"/>

<Pout ID="1003" Name="Отключены команды OPC" Type="Нет"/>

<Pout ID="1004" Name="Сдвиг" Type="Нет"/>

<Pin ID="1005" Name="CustomCommand" Type="Нет"/>

<Pin ID="1006" Name="Command" Type="Нет"/>

<Pout ID="1007" Name="OPCQUALITY" Type="Нет"/>

<Pout ID="1008" Name="CurrentSource" Type="Нет"/>

<Pout ID="1010" Name="bit15" Type="Нет"/>

<Pout ID="1011" Name="bit16" Type="Нет"/>

<Pout ID="1012" Name="bit17" Type="Нет"/>

<Pout ID="1013" Name="bit18" Type="Нет"/>

<Pout ID="1014" Name="bit19" Type="Нет"/>

<Pout ID="1015" Name="bit20" Type="Нет"/>

<Pin ID="1020" Name="ComputerName" Type="Нет"/>

<Pin ID="1021" Name="UserName" Type="Нет"/>

<Pin ID="1009" Name="PreparingCommand" Type="Нет"/>

<Pin ID ="1030" Name="PrepareCommand" Type="Нет"/>

<Pout ID="1031" Name="PreparedCommand" Type="Нет"/>

<!-- /Стандартная часть-->

<Pout ID="10" Name="OnStatusBody" Type="Нет"/>

<Pout ID="11" Name="Accident" Type="Нет"/>

<Pout ID="12" Name="Warning" Type="Нет"/>

<Pout ID="13" Name="Местное управление" Type="Логический"/>

<Pout ID="14" Name="Дистанционное управление" Type="Логический"/>

<Pout ID="15" Name="Автоматическое управление" Type="Логический"/>

<Pout ID="16" Name="Ручное управление" Type="Логический"/>

<Pout ID="17" Name="WarningDisplace" Type="Нет"/>

<Pout ID="18" Name="Перегрузка:w" Type="Логический"/>

<Pout ID="19" Name="Нет питания:w" Type="Логический"/>

<Pout ID="20" Name="Заблокирован:w" Type="Логический"/>

<Pout ID="31" Name="Warning" Type="Нет"/>

<Pout ID="32" Name="No\_OPC\_operation" Type="Нет"/>

<Pout ID="33" Name="OnStatusErrorCMD" Type="Нет"/>

<Pout ID="34" Name="OnStatusCMD" Type="Нет"/>

<Pout ID="35" Name="breakCmdOn" Type="Нет"/>

<Pout ID="36" Name="breakCmdOff" Type="Нет"/>

<Pout ID="37" Name="CommandOn" Type="Нет"/>

</Items>

</VisualMap>

<VisualMap> …</VisualMap> - описание входов выходов и для обмена данными с контролом, содержит структуру <Items>…</Items>, где Pin – это данные на входе, а Pout – данные на выходе.

<Events>

<!-- Стандартная часть-->

<Event ID="1" Name="Команда" Category ="Information" Flags="DisableAck" />

<Event ID="2" Name="Ошибка команды" Category ="Warning" />

<Event ID="3" Name="Авария" Category ="Error"/>

<Event ID="4" Name="Неисправность" Category ="Warning" />

<Event ID="5" Name="Отключено дистанционное управление" Category ="Information" Flags="DisableAck" />

<Event ID="6" Name="Сдвиг без команды" Category ="Warning" />

<!-- Номера зарезервированы. Создаются программно.

<Event ID="1010" Name="bit15" Category ="Warning" />

<Event ID="1011" Name="bit16" Category ="Warning"/>

<Event ID="1012" Name="bit17" Category ="Warning" />

<Event ID="1013" Name="bit18" Category ="Warning" />

<Event ID="1014" Name="bit19" Category ="Warning" />

<Event ID="1015" Name="bit20" Category ="Warning" />

-->

<Event ID="1016" Name="Данные не достоверны" Category ="Information" Flags="DisableAck"/>

<!-- /Стандартная часть-->

<Event ID="10" Name="Открыта" Category ="Information" Flags="DisableAck" />

<Event ID="11" Name="Закрыта" Category ="Information" Flags="DisableAck" />

<Event ID="12" Name="Промежуточное положение" Category ="Information" Flags="DisableAck" />

<Event ID="13" Name="Ошибка состояния" Category ="Warning"/>

<Event ID="14" Name="Местное управление" Category ="Information" Flags="DisableAck" />

<Event ID="15" Name="Дистанционное управление" Category ="Information" Flags="DisableAck" />

<Event ID="16" Name="Автоматическое управление" Category ="Information" Flags="DisableAck" />

<Event ID="17" Name="Ручное управление" Category ="Information" Flags="DisableAck" />

<Event ID="18" Name="Перегрузка" Category ="Warning"/>

<Event ID="19" Name="Нет питания" Category ="Warning"/>

<Event ID="20" Name="Застопорена" Category ="Warning"/>

<Event ID="21" Name="Обрыв цепи управления включением" Category ="Warning"/>

<Event ID="22" Name="Обрыв цепи управления отключением" Category ="Warning"/>

</Events>

<Rights>

<Right SystemRight="FBChangeState"/>

<Right SystemRight="FBOpenControlWindow"/>

<Right SystemRight="FBConfirm"/>

</Rights>

Структура элемента <Events> …</ Events > содержит <Event> – описание обычного сообщения. Например: « <Event ID="1" Name="Команда" Category ="Information" Flags="DisableAck" />», где 1 (ID) – идентификатор сообщения, передается в функции FireEvent, SetEventState, «Команда» (Name) – текст сообщения по умолчанию, «Information» (Category) – это категория, «DisableAck» (Flags) – флаг, означающий что сообщение не требует квитирования. В данном случае используется только этот флаг. Остальные сообщения построены по такому же принципу (в некоторых сообщениях используется другая категория – Warning).

Структура элемента <Rights>…</Rights> описание действий, которые могут совершаться пользователем из окна управления или по горячим клавишам. SystemRight задает, что оператор может выполнять определенное системное действие над этим ФБ, в данном случае: FBChangeState - изменить состояние; FBOpenControlWindow – открыть окно управления; FBConfirm – квитировать.

**FlapControl.cs**

ИСПОЛЬЗУЕМЫЕ БИБЛИОТЕКИ И КЛАССЫ

В этом разделе объявляются использованные библиотеки и пространства имен, подключенные файлу FlapControl.cs.

using ecl.commonClasses.BaseIm;

using ecl.commonClasses.BaseIm.ControlWnd;

using ecl.commonClasses.VentVs;

using ecl.StaticFbs.ProjectSettings;

using System;

using System.ComponentModel;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Runtime.InteropServices;

using System.Windows.Forms;

using static ecl.commonClasses.WindowElements;

Помимо стандартной библиотеки System, подключены библиотеки классов, такие как:

ecl.commonClasses.BaseIm (из которой в дальнейшем используется класс stdVisualPins, содержащий константы);

ecl.commonClasses.BaseIm.ControlWnd (из которой в дальнейшем используется класс StandartCommandsBuilder);

ecl.commonClasses.VentVs (благодаря ей в коде используются такие классы и методы, как: ConnectedToServer, OnPaint, ToRuntime, ToDesign, statusBody, updateData, DrawCurtain, orientation);

ecl.StaticFbs.ProjectSettings (из которой в дальнейшем используется класс ProjectSettings).

Пространство имен System.ComponentModel предоставляет классы, используемые для реализации поведения компонентов и элементов управления во время разработки и во время выполнения, в данном случае используется класс DisplayNameAttribute, для отображения имени свойства.

Пространство имен System.Drawing и System.Drawing.Drawing2D предоставляет доступ к основным графическим функциям GDI+, в данном случае к структурам Color, Brush и т.д.

Пространство имен System.Runtime.InteropServices предоставляет разнообразные члены, поддерживающие COM-взаимодействие и службы вызова платформы, в данном случае это пространство имен используется для регистрации FB, как COM-объекта.

Пространство имен System.Windows.Forms содержит классы: PaintEventArgs, MouseEventArgs.

Пространство имен static ecl.commonClasses.WindowElements содержит метод drawString.

КЛАССЫ И МЕТОДЫ

Описываемый класс FlapControl является наследником класса VentVsControl. Методы, содержащиеся в данном классе описаны ниже.

private Color GetConnectColor()

{

if (!ConnectedToServer || FBConnector.GetPinBool(stdVisualPins.OPCQualityGood) == false)

return ProjectSettings.ErrorState;

else

return Color.Black;

}

Метод Color устанавливает цвет отрисовки буквы при разрыве соединения с сервером (при успешном соединении цвет черный), с помощью структуры Color.

protected override void OnPaint(PaintEventArgs e)

{

activeControlToDraw.activeFBOn = activeFBOn;

activeControlToDraw.activeFbBackRectangle = false;

activeControlToDraw.width = Bounds.Width;

activeControlToDraw.height = Bounds.Height;

base.OnPaint(e);

updateData();

e.Graphics.InterpolationMode = InterpolationMode.HighQualityBilinear;

Brush fillBrush = Brushes.Transparent;

e.Graphics.FillRectangle(fillBrush, new Rectangle(0, 0, Bounds.Width, Bounds.Height));

Color brushColor = GetConnectColor();

if (statusLocal || statusDistance)

{

string s;

if (statusDistance)

s = "Д";

else

s = "М";

using (var brush = new SolidBrush(brushColor))

{

drawString(e, s, brush, new Point(Bounds.Width \* 98 / 100, Bounds.Height \* 78 / 100), new SizeF(Bounds.Width, Bounds.Height));

}

}

if (statusAuto || statusManual)

{

string s;

if (statusAuto)

s = "А";

else

s = "Р";

using (var brush = new SolidBrush(brushColor))

{

drawString(e, s, brush, new Point(Bounds.Width \* 98 / 100, Bounds.Height \* 2 / 100), new SizeF(Bounds.Width, Bounds.Height));

}

}

GraphicsContainer container = e.Graphics.BeginContainer();

if (orientation == Orientation.vertical)

{

Matrix transform = e.Graphics.Transform;

transform.RotateAt(270, new Point(0, 0));

transform.Translate(-Bounds.Height, 0);

e.Graphics.Transform = transform;

}

using (Pen rectangleColor = new Pen(Color.Black, 1),

lineColor = new Pen(Color.Black, Bounds.Width / 25))

{

e.Graphics.DrawRectangle(rectangleColor, 1, 1, Bounds.Width - 2, Bounds.Height - 2);

e.Graphics.DrawLine(lineColor, 1, Bounds.Height / 2, Bounds.Width / 10, Bounds.Height / 2);

e.Graphics.DrawLine(lineColor, Bounds.Width - 1, Bounds.Height / 2, Bounds.Width \* 9 / 10, Bounds.Height / 2);

}

drawFlap(e.Graphics, Bounds.Width, Bounds.Height);

e.Graphics.EndContainer(container);

if (Commands?.AllDisabled == true)

{

DrawCurtain(e.Graphics, new SizeF(Bounds.Width, Bounds.Height));

}

}

Перегружаемый метод OnPaint доступен из класса VentVs и предназначен для отрисовки границ, сглаживания линий (e.Graphics.InterpolationMode), заливки прозрачным фоном (e.Graphics.FillRectangl), задания ориентации (вертикальной - Orientation.vertical или горизонтальной), отрисовки элементов поля, отрисовки текста: местный - statusLocal, дистанционный – statusDistance; автоматический – statusAuto, ручной – statusManual.

private void drawFlap(Graphics g, int width, int height)

{

int xInd1 = width \* 45 / 100;

int yInd1 = height \* 45 / 100;

int dInd = width \* 1 / 10;

int xDifferent1 = (width \* 707 / 1000 + width \* 8 / 100);

int xDifferent2 = (width \* 141 / 1000 + width \* 8 / 100);

int yDifferent1 = (height \* 141 / 1000 + height \* 8 / 100);

int yDifferent2 = (height \* 707 / 1000 + height \* 8 / 100);

//Изменение цвета "Индикатора" по состоянию

int thick = width / 20; //толщина линии заслонки

if (!ConnectedToServer || !FBConnector.GetPinBool(stdVisualPins.OPCQualityGood))

{

using (Pen lineColorDef = new Pen(ProjectSettings.Undefined, thick))

g.DrawLine(lineColorDef, xDifferent1, yDifferent1, xDifferent2, yDifferent2);

//Заливка "Оси"

using (SolidBrush fillColor = new SolidBrush(ProjectSettings.Undefined))

g.FillEllipse(fillColor, xInd1, yInd1, dInd, dInd);

return;

}

switch (statusBody)

{

default:

using (Pen lineColorDef = new Pen(ProjectSettings.Undefined, thick))

g.DrawLine(lineColorDef, xDifferent1, yDifferent1, xDifferent2, yDifferent2);

//Заливка "Оси"

using (SolidBrush fillColor = new SolidBrush(ProjectSettings.Undefined))

g.FillEllipse(fillColor, xInd1, yInd1, dInd, dInd);

break;

case OnStatuses.STATUSON:

int xOpen1 = width \* 5 / 10;

int yOpen1 = height \* 1 / 10;

int yOpen2 = height \* 9 / 10;

using (Pen lineColorOn = new Pen(ProjectSettings.Opened, thick))

g.DrawLine(lineColorOn, xOpen1, yOpen1, xOpen1, yOpen2);

//Заливка "Оси"

using (SolidBrush fillColorOn = new SolidBrush(ProjectSettings.Opened))

g.FillEllipse(fillColorOn, xInd1, yInd1, dInd, dInd);

break;

case OnStatuses.STATUSOFF:

int xClose1 = width \* 1 / 10;

int xClose2 = width \* 9 / 10;

int yClose1 = height \* 5 / 10;

using (Pen lineColorOff = new Pen(ProjectSettings.Closed, thick))

g.DrawLine(lineColorOff, xClose1, yClose1, xClose2, yClose1);

//Заливка "Оси"

using (SolidBrush fillColorOff = new SolidBrush(ProjectSettings.Closed))

g.FillEllipse(fillColorOff, xInd1, yInd1, dInd, dInd);

break;

case OnStatuses.STATUSERROR:

using (Pen lineColorErr = new Pen(ProjectSettings.ErrorState, thick))

g.DrawLine(lineColorErr, xDifferent1, yDifferent1, xDifferent2, yDifferent2);

//Заливка "Оси"

using (SolidBrush fillColorErr = new SolidBrush(ProjectSettings.ErrorState))

g.FillEllipse(fillColorErr, xInd1, yInd1, dInd, dInd);

break;

case OnStatuses.STATUSDIFFERENT:

using (Pen lineColorErr = new Pen(ProjectSettings.DifferentState, thick))

g.DrawLine(lineColorErr, xDifferent1, yDifferent1, xDifferent2, yDifferent2);

//Заливка "Оси"

using (SolidBrush fillColorErr = new SolidBrush(ProjectSettings.DifferentState))

g.FillEllipse(fillColorErr, xInd1, yInd1, dInd, dInd);

break;

}

}

Метод drawFlap отвечает за описание графического элемента «Индикатор». Переменные xInd1, yInd1, dInd, xDifferent1, xDifferent2, yDifferent1, yDifferent2 предназначены для расчета координат точек оси относительно размеров поля. Переменная thick задает толщину линии заслонки. Заливка оси происходит с помощью класса SolidBrush.

[NonSerialized]

private BaseControlWnd wnd;

[NonSerialized]

private CommandsController Commands;

С помощью атрибута [NonSerialized]осуществляем вызов панели управления.

protected override void ToRuntime()

{

base.ToRuntime();

Commands = new CommandsController(FBConnector, new StandartCommandsBuilder(FBConnector));

Commands.AllCommandsDisableChanged += Commands\_AllCommandsDisableChanged;

}

protected override void ToDesign()

{

base.ToDesign();

if (Commands != null)

Commands.Dispose();

if (wnd != null && !wnd.IsDisposed)

wnd.Close();

}

Для реализации работы ФБ в Runtime возможно переопределить три функции базовых классов, к ним относятся: ToRuntime: вызывается при переходе в Runtime, ToDesign: вызывается при переходе в Designtime.

public void Paint(PaintEventArgs e, SizeF size)

{

updateData();

using (Pen rectangleColor = new Pen(Color.Black, 1),

lineColor = new Pen(Color.Black, size.Width / 25))

{

e.Graphics.DrawRectangle(rectangleColor, 1, 1, size.Width - 2, size.Height - 2);

e.Graphics.DrawLine(lineColor, 1, size.Height / 2, size.Width / 10, size.Height / 2);

e.Graphics.DrawLine(lineColor, size.Width - 1, size.Height / 2, size.Width \* 9 / 10, size.Height / 2);

}

drawFlap(e.Graphics, (int)size.Width, (int)size.Height);

if (Commands?.AllDisabled == true)

{

DrawCurtain(e.Graphics, size);

}

}

Метод Paint предназначен для работы с классом PaintEventArgs и отрисовки элементов полей с помощью класса Pen.

public float GetAspectRastio()

{

switch (orientation)

{

case Orientation.horizontal:

return Bounds.Height / Bounds.Width;

case Orientation.vertical:

return Bounds.Width / Bounds.Height;

default:

return 1.0f;

}

}

Метод GetAspectRastio предназначен для того, чтобы задать либо горизонтальную ориентацию (Orientation.horizontal), либо вертикальную ориентацию (Orientation.vertical). В любом другом случае метод вернет значение 1.0.

Flap.bmp

C:\Users\Nasti\Downloads\_asue(gitloc)\VisualFbs\Flap\Flap.bmp

Изображение ФБ в формате **bmp-файла должно быть создан ов той же подпапке проекта что и файл класса ФБ**. Имя bmp-файла должно совпадать с именем класса ФБ.