Создаем стрелочный индикатор.

Для отображения изменяющихся величин, таких как напряжение, давление, скорость и т.д., можно обойтись стандартными средствами Visual Basic: Textbox и Label.
Недостаток – постоянно изменяющиеся цифры плохо воспринимаются оператором. Человек имеет образное мышление и, для того чтобы воспринять цифровую информацию, ему нужно вначале создать некоторый мысленный образ. График или стрелочный индикатор являются уже готовым образом, который воспринимается во многом подсознательно. Реакция человека на такой образ является рефлекторной, если этот образ уже “записан” в памяти, и человек имеет какой-то опыт реагирования на него. Именно поэтому стрелочные индикаторы устанавливаются (или эмулируются) на рабочих панелях автомобилей, самолетов и т.п. То есть там, где у человека нет времени на рассуждения о том, “а что эти циферки означают?”.

Наш индикатор мы выполним в виде ActiveX объекта, подобно кнопке или другим элементам управления. В этом случае интерфейс различных программ собирается, словно из кубиков. Мы можем использовать любое количество индикаторов без написания дополнительного кода. Управление индикаторами также упрощается. Для того, чтобы стрелка повернулась в заданное положение, нам необходимо лишь задать новое значение параметра: Gauge.value=xxx.

Итак, приступим.

Поскольку наш индикатор запланирован фотореалистичным, нужно найти фотографию реального прибора с неискаженной перспективой. Эту проблему решают пять минут поиска в сети Интернет. Вот и подходящий индикатор:

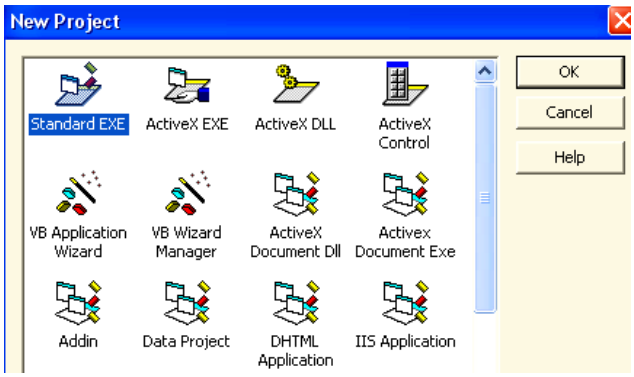
Откроем рисунок в программе обработки изображений (я пользуюсь Corel Photo Paint) и вырежем окружность корпуса прибора. После этого наложим круг серого цвета по центру, и получим круглую рамку, с которой и будем работать в дальнейшем.

Конечно, мы можем создать какие угодно рамки, изменяя цветовую палитру и т.д. Например, такие как эти.

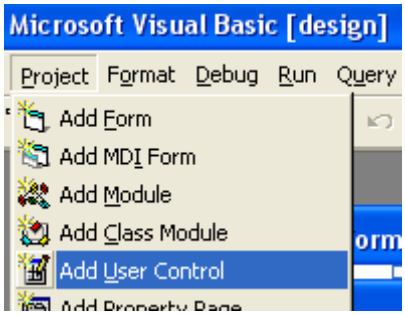
На этом художества заканчиваются, и начинается работа над программным кодом.

Создание объекта ActiveX.

Перед началом работы над пользовательским элементом управления (ПЭУ) удобно вначале создать форму – контейнер, в которой он будет тестироваться. Это обычный проект Visual Basic: Standard EXE.

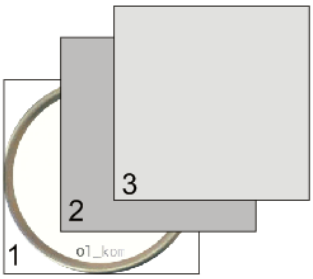


После создания пустой формы добавим в проект ActiveX Control:



Если Вы хотите использовать в своем проекте готовый пользовательский элемент управления, например, рассматриваемый стрелочный индикатор, нужно выбрать пункт меню **Existing**, после чего найти элемент на диске.

Вывод графической информации с двойной буферизацией.



Простейший путь создания индикатора – поместить в форму объект Picture Box, загрузить в него фоновую картинку. После этого сделать нужные надписи, нарисовать шкалы и отобразить стрелку. Как только изменится отображаемая величина, очистить Picture Box и нарисовать все заново. У такого способа есть два больших недостатка: скорость обновления будет низкой, индикатор будет неприятно мигать. Выход из ситуации состоит в использовании двойной буферизации. Для реализации метода создадим не один, а три объекта Picture Box. Первый содержит фоновую картинку. Кроме того, в нем мы прорисуем то, что не нужно постоянно изменять: надписи и шкалы. Второй графический объект предназначен для отображения стрелки. В нем производится “сборка” всего изображения, которое на последнем этапе практически мгновенно “выбрасывается” в третье графическое окно, и пользователь видит уже готовый

результат. В нашем проекте первое окно называется Original_Picture, второе - Assembly_Picture и третье - Visible_Picture.

При инициализации ПЭУ, изменении надписей, перерисовке шкал запускается процедура Draw_First. Алгоритм ее работы следующий (все действия производятся в окне Original_Picture).

- Вычисляется размер шрифта, который будет использоваться для надписей в зависимости от размера ПЭУ.
- Запускается процедура Draw_Labels, которая делает надписи напротив основных делений индикатора (0, 10, 20, …100).
- Запускается процедура Draw_Lines, которая строит основные (длинные) и вспомогательные (короткие) риски шкалы в зависимости от значений переменных m_N_Lines и m_N_Sublines.
- Процедура Print_Labels предназначена для прорисовки верхней (“PRESSURE”) и нижней (“Bar”) надписей.

Как только возникает необходимость изменить показания индикатора, а также в момент инициализации ПЭУ, активизируется функция Show_Value. Эта функция отображает цифровое значение отображаемой индикатором величины и вычисляет требуемый угол поворота стрелки. Далее вызывается процедура Draw_Arrow, в которую передается угол поворота.

Первое действие, которое производится в процедуре Draw_Arrow – перенос в “сборочное” окно Assembly_Picture содержимого Original_Picture. Перенос осуществляется с помощью API – функции BitBlt, обеспечивающей максимальную скорость процесса.

```
BitBlt Assembly_Picture.hDc, 0, 0, Assembly_Picture.ScaleWidth,
Assembly_Picture.ScaleHeight, Original_Picture.hDc, 0, 0, SRCCOPY
```

Список параметров функции BitBlt включает в себя дескриптор окна Assembly_Picture; координаты вершины графической области, в которую будет вставлено изображение; размер этой графической области; дескриптор окна Original_Picture; константу SRCCOPY. Подробно о работе функции BitBlt можно узнать из справочника по WinAPI.

Создание стрелки и окончательная сборка.

После того, как в “сборочное” окно были перенесены фоновая картинка вместе с готовыми шкалами и надписями, определяются координаты четырех точек, по которым строится стрелка индикатора. Эти координаты заносятся в созданный заранее массив:

```
Private Type POINTAPI
  X As Long
  Y As Long
End Type

Dim point(4) As POINTAPI
```

Это делается потому, что рисующая многоугольник API функция Polygon работает именно с таким форматом данных. Сама прорисовка стрелки:

```
temp1 = Polygon(Assembly_Picture.hDc, point(0), 4)
```

Здесь мы даем ссылку на дескриптор окна Assembly_Picture, первый элемент массива point и указываем количество точек в массиве.

Рисуем черный кружок по центру индикатора: Assembly_Picture.Circle (cx, cy), cc, &HFFFFFF, после чего переносим готовое изображение в окно Visible_Picture. Для этого используется функция StretchBlt, позволяющая корректно масштабировать изображение при переносе.

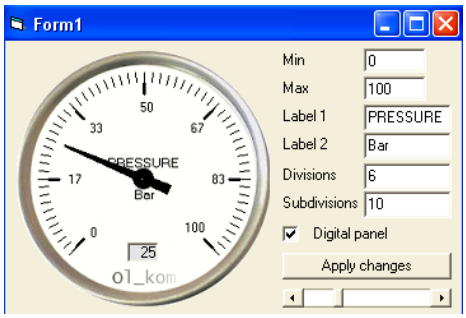
X = **StretchBlt**(Visible_Picture.hDc, 0, 0, W2, H2, Assembly_Picture.hDc, 0, 0, W1, H1, SRCCOPY)

Масштабирование нам нужно потому, что фоновая картинка имеет фиксированный размер, а размер индикатора пользователь может изменять как угодно. Есть и еще одна причина: создаваемый функцией Polygon многоугольник, имеет не очень красивые края. Лучший результат достигается при прорисовке многоугольника в большем окне с последующим уменьшением при переносе. Функция StretchBlt в данном случае выполнит корректное масштабирование, и улучшит прорисовку краев стрелки.

Вот, собственно, и все, что относится к основному алгоритму работы ПЭУ. О назначении функций UserControl_InitProperties, UserControl_ReadProperties, UserControl_WriteProperties и процедур изменения свойств ПЭУ Property Get и Property Let в сети Интернет написано достаточно много.

Как только ПЭУ появляется в форме – контейнере, процедура ReadProperties считывает значения всех свойств ПЭУ из переменной типа PropertyBag. В этот момент в тело процедуры вставлены функции Draw_First и Show_Value, предназначенные для отображения индикатора на экране.

Тестирование ПЭУ.



Для тестирования нашего индикатора, в форме – контейнере предусмотрим возможность изменения основных параметров ПЭУ: минимального и максимального отображаемых значений; двух надписей; количества основных и вспомогательных делений шкалы; переменной, определяющей, будет ли отображаться цифровой индикатор в нижней части. Нажатие кнопки Apply changes вызовет процедуры Property Let объекта ActiveX, за чем последует перерисовка индикатора. Движок в нижней части предназначен для изменения свойства Gauge1.Value. Индикатор имеет и другие параметры, которые отображаются и могут быть изменены в окне Properties.