ИЗМЕНЕНИЕ РАСШИРЕННЫХ НАСТРОЕК ПЕЧАТИ ЧЕРЕЗ API nanoCAD

Расширенные настройки печати в nanoCAD являются дополнением к базовым настройкам.

Через пользовательский интерфейс nanoCAD невозможно увидеть, какая настройка является базовой, а какая – расширенной, однако при работе с API nanoCAD разница видна четко. В качестве примера можно взять настройку выравнивания области печати на листе. Если бы существовала возможность использовать только базовую настройку, то область печати удавалось бы выровнять лишь по центру листа. Расширенная настройка позволяет выравнивать область печати не только по центру, но и по сторонам листа.

В этой статье будут подробно рассмотрены настройки печати, которые относятся к расширенным, а также способы их изменения через API nanoCAD.

В интерфейсе nanoCAD расширенные настройки печати доступны как в диалоговых окнах *Печать* и *Параметры листа* (на рис. 1 выделены цветом), так и в окне настройки параметров встроенных принтеров nanoCAD (рис. 2), которое вызывается кнопкой *Настройка* в блоке *Принтер* окна *Печать* или *Параметры листа*.

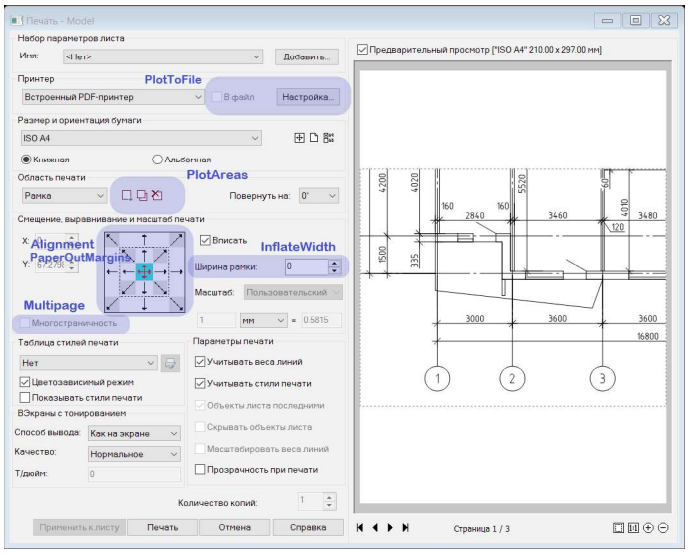

Рис. 1

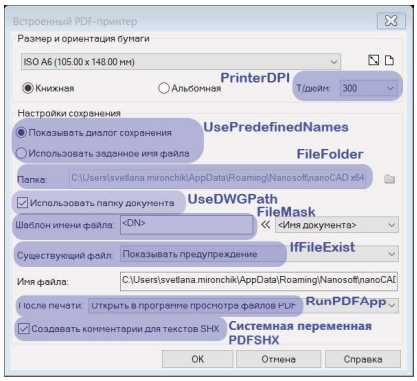

Рис. 2

Окна настройки расширенных параметров для pdf-принтеров и растровых отличаются внешне, однако через API для обоих видов принтеров доступны одни и те же настройки.
Расширенные настройки печати через API nanoCAD изменяются через изменение свойства customPlotSettings нужного листа документа nanoCAD:

```
// Получение ссылки на свойство Plot активного документа
nanoCAD.Plot plot = (nanoCAD.Plot)comDoc.Plot;

// Получение ссылки на расширенные настройки печати активного документа
nanoCAD.InanoCADPlotCustomParams customPlotSettings = plot.CustomPlotSettings[activeLayout];
```

Чтобы подробнее изучить, какие расширенные настройки доступны через API nanoCAD и как их изменять, создадим .NET (C#)-приложение, которое будет устанавливать активному листу документа встроенный в nanoCAD pdf-принтер и изменять расширенные настройки печати.
Рассмотрим поэтапно, как создать такое приложение.

Шаг 1. Создаем в Visual Studio проект типа "Библиотека классов (.NET Standard)". Подключаем к нему библиотеки *hostmgd.dll*, *hostdbmgd.dll* и *ncauto.dll*.

Шаг 2. Создаем класс, в котором будет находиться метод, назначающий активному листу документа nanoCAD pdf-принтер и меняющий расширенные настройки печати.

```
public partial class Commands
{
    [Teigha.Runtime.CommandMethod("CustomPlotSettings")]
    public void CustomPlotSettings()
    {
    }
}
```

В последующих шагах содержится инструкция по реализации метода CustomPlotSettings().

Шаг 3. Получение доступа к документу nanoCAD и его настройкам.

```
// Получение ссылки на активный документ, его базу данных и редактор
HostMgd.ApplicationServices.Document doc =
    HostMgd.ApplicationServices.Application.DocumentManager.
MdiActiveDocument;
nanoCAD.Document comDoc = doc.AcadDocument as nanoCAD.
Document;
Teigha.DatabaseServices.Database db = doc.Database;
HostMgd.EditorInput.Editor editor = doc.Editor;
```

Шаг 4. Получение ссылки на активный лист и расширенные настройки печати.

```
// Получение ссылки на активный лист документа
OdaX.IAcadLayout activeLayout = comDoc.ActiveLayout;

// Получение ссылки на свойство Plot активного документа
nanoCAD.Plot plot = (nanoCAD.Plot)comDoc.Plot;

// Получение ссылки на расширенные настройки печати
nanoCAD.InanoCADPlotCustomParams customPlotSettings =
        plot.CustomPlotSettings[activeLayout];

// Назначение активному листу встроенного pdf-принтера
activeLayout.ConfigName = "Встроенный pdf-принтер";
```

Также сразу назначаем встроенный pdf-принтер активному листу.

Шаг 5. Изменяем расширенные настройки через переменную customPlotSettings и передаем измененные настройки в документ.

```
// Определяем, как выровнять область печати на листе
customPlotSettings.PaperOutMargins = 1;
customPlotSettings.Alignment = 3;

// Определяем толщину рамки вокруг области печати
customPlotSettings.InflateWidth = 10;

// Определяем, в какую папку сохранить напечатанный файл
customPlotSettings.UseDWGPath = false;
customPlotSettings.FileFolder = "C:\\Work";

// Определяем, с каким именем будет сохранен напечатанный файл
customPlotSettings.UsePredefinedNames = true;
customPlotSettings.FileMask = "<DN>_<LN>";

// Определяем разрешение печати
customPlotSettings.PrinterDPI = 1000;

// Определяем действия при наличии в указанной нами папке для сохранения
// файла с таким же именем
customPlotSettings.IfFileExist = 1;

// Определяем, открывать документ сразу после печати или нет
customPlotSettings.RunPDFApp = true;

// Определяем признак создания комментариев SHX
using (Teigha.DatabaseServices.Transaction transaction =
    db.TransactionManager.StartTransaction())
{
  HostMgd.ApplicationServices.Application.
SetSystemVariable("PDFSHX", false);
  transaction.Commit();
}

// Добавляем несколько областей печати "Рамка" в активный лист
```

```
for(int i = 0; i <= 2; i++)
{
   // Очистка коллекции PlotAreas перед добавлением первой новой "Рамки"
   if (i==0) customPlotSettings.PlotAreas.Clear();

   // Запрос точки левого нижнего угла области печати
   var point1 = editor.GetPoint("Укажите левую нижнюю точку области печати:");
   if (point1.Status == HostMgd.EditorInput.PromptStatus.OK)
   {
      // Запрос точки правого верхнего угла области печати
      var point2 = editor.GetCorner("Укажите правую верхнюю точку области печати:", point1.Value);
      if (point2.Status == HostMgd.EditorInput.PromptStatus.OK && point2.Value != point1.Value)
      {
         // Добавление области печати "Рамка" в коллекцию PlotAreas
         customPlotSettings.PlotAreas.Add(new double[] { point1.Value.X, point1.Value.Y },
                                          new double[] { point2.Value.X, point2.Value.Y });
      }
      else { editor.WriteMessage("Введены неверные данные"); return; }
   }
   else { editor.WriteMessage("Введены неверные данные"); return;
}
}

// Проверка, установлен ли активному листу тип области печати "Рамка"
if (activeLayout.PlotType != OdaX.AcPlotType.acWindow)
   activeLayout.PlotType = OdaX.AcPlotType.acWindow;

// Передаем измененные настройки в активный лист
plot.CustomPlotSettings[activeLayout] = customPlotSettings;
```

Измененные настройки обязательно следует передать в документ, без этого действия расширенные настройки не изменятся!

Рассмотрим каждую расширенную настройку печати в том порядке, в котором они расположены в переменной customPlotSettings.

int Alignment – выравнивание области печати по сторонам листа (см. рис. 1).

Может принимать значения, указанные на рис. 3: каждой стороне листа соответствует определенное значение типа int. Свойство применяется совместно со свойством PaperOutMargins.

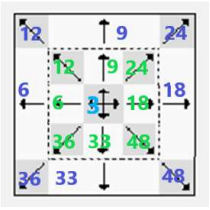

Рис. 3. Значения свойства *Alignment* на панели выравнивания области печати окна *Параметры листа*: значения зеленого цвета – при PaperOutMargins=0, значения синего цвета – при PaperOutMargins=1

- string FileFolder – строка с адресом директории, в которую будет сохранен файл после печати (см. рис. 2).
  Используется при значении свойства UseDWGPath = false, которое означает, что напечатанный документ будет сохранен не в папку расположения файла чертежа, а в папку, указанную в свойстве FileFolder. При значении UseDWGPath = true свойство FileFolder игнорируется.
- string FileMask – строка с именем файла, которое будет присвоено напечатанному документу (см. рис. 2).
  Используется при значении свойства UsePredefinedNames = true и может содержать в себе набор символов, которые будут распознаны в nanoCAD как автозаполняемые поля. Существует несколько вариаций этих полей, которые можно комбинировать друг с другом, используя между ними разделители, допустимые в названиях файла:
  - <DN> – имя документа;
  - <LN> – имя листа;
  - <UN> – имя пользователя;
  - <T> – время;
  - <D> – дата;
  - <C1> – счетчик 1. Счетчик добавляет индекс к названию файлов, начиная с 1;
  - <C2> – счетчик 01. Счетчик добавляет индекс к названию файлов, начиная с 01;
  - <C3> – счетчик 001. Счетчик добавляет индекс к названию файлов, начиная с 001;
  - <C4> – счетчик 0001. Счетчик добавляет индекс к названию файлов, начиная с 0001;
  - <C5> – счетчик 00001. Счетчик добавляет индекс к названию файла, начиная с 00001;
  - <C6> – счетчик 000001. Счетчик добавляет индекс к названию файла, начиная с 000001.

  Например, комбинация <DN>_<LN>_<C2> сформирует имя файла типа "Имя документа_Имя листа_01".
- int IfFileExist – порядок действий на случай, если в папке сохранения файла существует файл с таким же именем (см. рис. 2).
  Возможны следующие значения:
  - 0 – показывать предупреждение;
  - 1 – всегда пересохранять в существующий файл;
  - 2 – автонумерация имени файла;
  - 3 – присоединить страницу к существующему файлу.
- double InflateWidth – отступ от полей бумаги в формате числа double, то есть ширина рамки вокруг области печати (см. рис. 1).
- bool Multipage – управление признаком многостраничности (см. рис. 1).
  Имеет два возможных значения: true и false.
  Автоматически изменяется на true при добавлении в лист нескольких областей печати "Рамка". Через интерфейс доступно для изменения только в листах; предназначено для размещения чертежа на нескольких листах меньшего формата, чем чертеж, если принтер не поддерживает формат чертежа.
- int PaperOutMargins – выравнивание области печати на листе по границам листа или по полям печати (см. рис. 1).
  Возможны два значения (рис. 4):
  - 0 – выравнивание области печати по полям печати листа;
  - 1 – выравнивание области печати по границе листа.

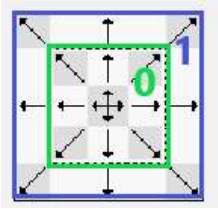

Рис. 4

По умолчанию равно 0. Применяется совместно со свойством Alignment.

- nanoCAD.PlotAreas PlotAreas – коллекция областей печати "Рамка" одного листа (см. рис. 2).

Тип данных nanoCAD.PlotAreas включает в себя следующие методы и свойства:

- метод Add(ptMin, ptMax) – добавляет новую область печати "Рамка" в конец коллекции. В качестве параметров принимает массивы double с координатами точек двух противоположных углов прямоугольной области печати;
- метод Clear() – удаляет все элементы, содержащиеся в коллекции на момент его вызова;
- свойство Count – предназначено только для чтения, возвращает количество элементов в коллекции.

Элементы коллекции имеют тип данных nanoCAD.PlotArea. Доступ к элементам коллекции – по индексу: customPlotSettings.PlotAreas[0]. У каждого элемента есть свойства, через которые можно изменять параметры конкретной области печати "Рамка" из коллекции.

- bool PlotToFile – признак печати в файл (см. рис. 1).
  Возможны два значения: true – печатать в файл, false – не печатать в файл. Во встроенных в nanoCAD принтерах это значение по умолчанию равно true, для остальных принтеров возможны оба варианта.
- int PrinterDPI – разрешение печати, настраивается только во встроенных в nanoCAD принтерах (см. рис. 2).
  Возможные значения DPI для встроенных принтеров: 75, 150, 200, 300, 400, 600, 1000, 1200.
- bool RunPDFApp – просмотр документа после печати. Работает также и для растрового принтера (см. рис. 2).
- bool UseDWGPath – настройка директории сохранения файла (см. рис. 2).
  Возможны два значения: true – сохранить в директорию расположения документа nanoCAD; false – использовать директорию, отличную от директории расположения документа, который будет отправлен на печать.
  Директорию можно указать в свойстве FileFolder либо установить свойству UsePredefinedNames значение false, чтобы перед печатью появилось диалоговое окно сохранения файла.
- bool UsePredefinedNames – управление именем файла (см. рис. 2).
  Возможны два значения: true – использовать шаблон имени файла, false – показать диалоговое окно для ввода имени файла.
  Если свойство имеет значение true, имя файла должно быть заранее определено в свойстве FileMask.
- Системная переменная PDFSHX – управляет признаком создания комментариев SHX при печати на встроенном pdf-принтере (см. рис. 2).
  Принимает два значения: true (или 1) – создавать комментарии SHX в pdf-файле, false (или 0) – не создавать комментарии SHX.

Шаг 5.1. Добавим вывод значений расширенных настроек в консоль до их изменения и после в виде вызова метода DisplayCustomPlotSettings().

```
    private void
DisplayCustomPlotSettings(nanoCAD.InanoCADPlotCustomParams
customPlotSettings, HostMgd.EditorInput.Editor ed)
    {
        ed.WriteMessage("PaperOutMargins is {0}, Alignment is {1}",
customPlotSettings.PaperOutMargins, customPlotSettings.
Alignment);
        ed.WriteMessage("InflateWidth is {0}",
customPlotSettings.InflateWidth);
        ed.WriteMessage("UseDWGPath is {0}, FileFolder is {1}",
customPlotSettings.UseDWGPath, customPlotSettings.FileFolder);
        ed.WriteMessage("UsePredefinedNames is {0}, FileMask is {1}",
customPlotSettings.UsePredefinedNames, customPlotSettings.
FileMask);
        ed.WriteMessage("Printer DPI is {0}", customPlotSettings.
PrinterDPI);
        ed.WriteMessage("If file exist is {0}", customPlotSettings.
IfFileExist);
        ed.WriteMessage("RunPDFApp is {0}", customPlotSettings.
RunPDFApp);
        ed.WriteMessage("SHX comments are {0}", HostMgd.
ApplicationServices.Application.GetSystemVariable("PDFSHX"));
        ed.WriteMessage("{0} plot areas 'Window'", customPlotSettings.
PlotAreas.Count);
    }
}
```

Шаг 6. Компилируем наше приложение, загружаем его в nanoCAD и запускаем.

При успешном завершении работы команды *CustomPlotSettings* в консоли появится сообщение пользователю, показанное на рис. 5.

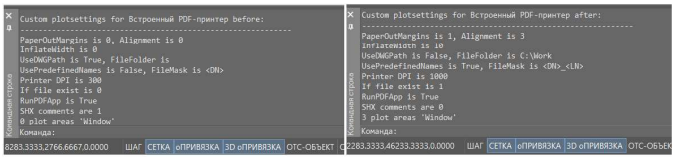

Рис. 5

Итак, мы разработали .NET (C#)-приложение, которое работает только с расширенными настройками печати nanoCAD. В процессе его создания были приведены примеры работы с этими настройками печати, в том числе с выравниванием области печати на листе или добавлением нескольких областей печати "Рамка" на один лист через API nanoCAD.

На этом мы завершаем цикл материалов, посвященных API печати nanoCAD. Подводя итоги, кратко перечислим то, что вы можете узнать, изучив все четыре статьи:

- как назначать документу принтер и отправлять документ на печать;
- как работать с наборами параметров листов и как копировать их из других документов;
- как работать с различными областями печати;
- что такое расширенные настройки печати и как их изменять.

Светлана Мирончик,
Клуб разработчиков nanoCAD
ООО "Нанософт разработка"