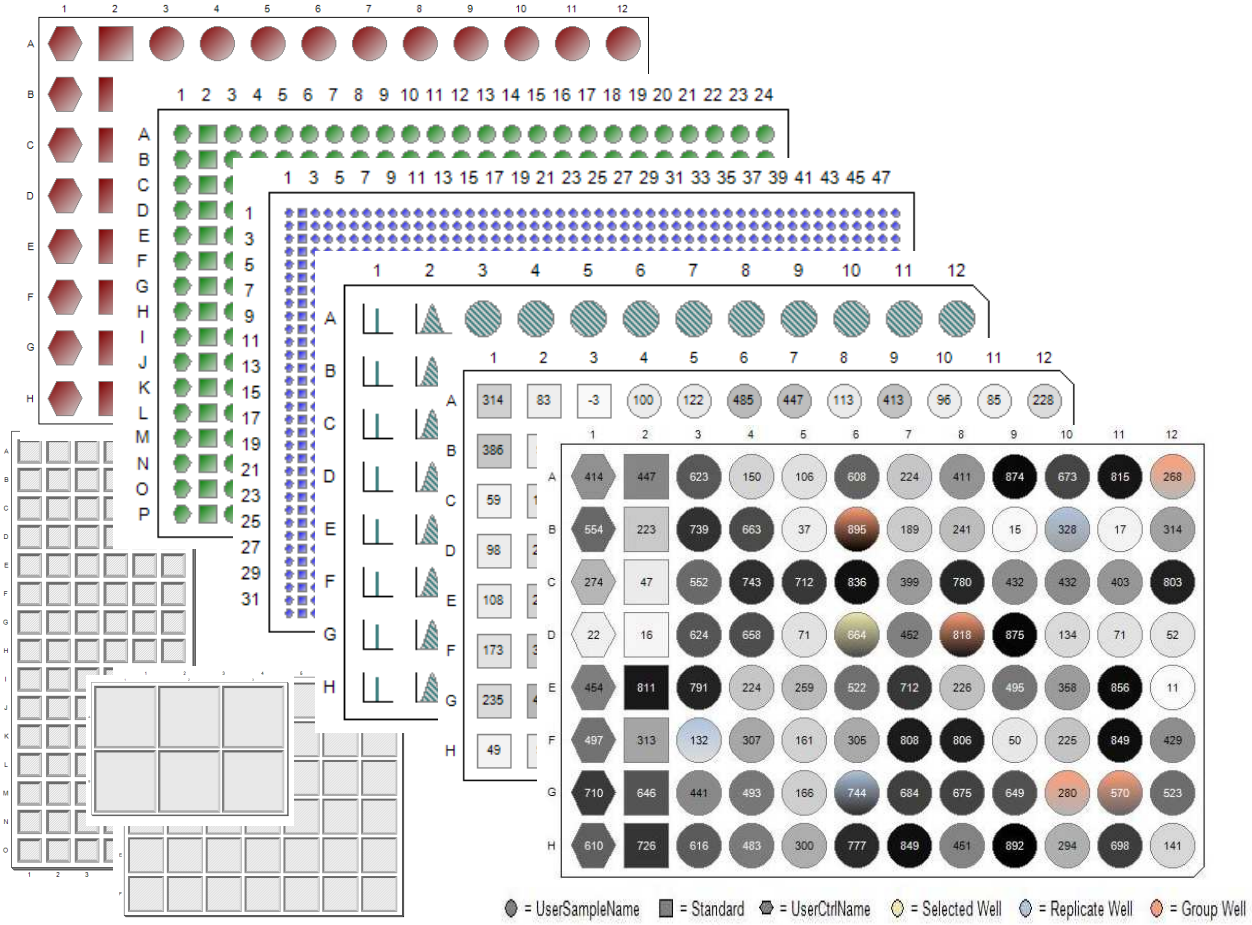
BxMicroTitrePlate

Version 3.4.0.0

Programmer's Manual



July 15, 2009

BioXing

www.bioxing.com

Table of Contents

List of Figures

List of Tables

1 Introduction

Biotechnology and Bioinformatics utilize microtitre plates as a standard format for reference and unknown sample storage and processing, instrumentation, analysis, aggregation and retrieval of data. Since microtitre plates play such a central role, BioXing has developed a highly customizable interactive and enhanced microtitre plate component, called *BxMicroTitrePlate*, which can be easily incorporated into .NET Windows applications. The component is useful for rendering an interactive microtitre plate graphic in applications that perform operations or measurements using microtitre plates. In particular for instrument control, for data acquisition monitoring and analysis summary, multiple plate comparisons, as a navigation tool, and as an interface to facilitate accessing and integrating related well content information.

BxMicroTitrePlate utilizes well shapes, well gray scale coloring and well annotation to concisely convey summary information and individual detail about a well and any related wells. In addition, wells may display images that reflect the sample source or images taken over a time line for comparison when using multiple plates or wells may display two-dimensional graphs (XY Data) that could, for example, be a result of real-time data acquisition and analysis. In addition, a well or set of wells can be blinked and/or colored to reflect well activity during acquisition and analysis. Each well can display a customizable descriptive tooltip and can trigger events that can be used to retrieve, link and add information to the well, thereby facilitating the integration of disparate data. XY data graphs can be annotated with text or image peak labels when displayed in a Well Magnifier View. The underlying structure of the component uses a Data Table and provides change events that can be used for linking to and updating connected persistent data storage such as a database or files. The component provides a UI for defining customization settings and exporting them in an XML data stream that can be saved in a database or file which can then be used for initializing the component within applications.

Due to its flexible design the control can be used to represent generic interactive arrays of 1 x m (single row), n x1 (single column) or n x m where n and m are the number of rows and columns in the array and through the use of the customization properties the array does not have to have the appearance of microtitre plate. This is useful, for example, when representing rectangular fraction collectors or representing collections of data (numbers, graphs or images) for comparison.

Standard 96-well, 384-well and 1536-well plate sizes are selectable and other sizes are easily defined. Important features of this component are multiple shapes to represent different types of well content such as samples, standards, controls, targets, and probes with gray-scale well fill patterns to represent the magnitude of known and measured values respectively. Wells can have replicates and/or designated as belonging to a group. Selectable annotation or images can be placed within wells along with customizable tooltips that appear as the mouse moves over wells. The intent is to provide a view that can be used to convey multiple pieces of information and can be used interactively by

providing event triggers when the mouse is over a well and when the mouse is clicked on a well.

This document will describe the architecture of the component, its API (Application Programming Interface) and present examples to illustrate its use and flexibility. In addition, a collection of forms will be described that can be added to an application to provide the capability for user customization of any aspect of the appearance of a microtitre plate or matrix graphic,

2 BxMicroTitrePlate Overview

BxMicroTitrePlate is contained within the BxMicroTitrePlate.dll dynamic link library which is accessed using the BxTools namespace and the MicroTitrePlate User Control class. All of the capabilities of a User Control are provided such as adding it to a collection of other controls on a form and managing it through standard control properties; however, the design also permits rendering the MicroTitrePlate on any graphics device. For example, it can be printed using a Printer's graphic device or it can be copied to the clipboard or a Web Page by first rendering it on a bitmap using its graphic object.

The intent is for it to be used within applications to programmatically render an interactive graphic customizable through an extensive set of properties that creates a view similar to microtitre plates being used in a laboratory. In addition the wells of the plate can be assigned shapes to represent the type of content within the well and the interior of the well can be used to represent the magnitude of its known value such as concentration or a measured or calculated value such as absorption or concentration calculated from a standard curve. That is, the well interior could contain a gray-scale image that reflects a magnitude and/or annotation text.

For example, in Figure 1 the squares in the 96-Well plate and graph represent the standard curve data where the interior well gray-scale represents the magnitude of absorption or concentration and the annotation displays its numeric value. The dash-lined square rectangle at C8 represents a selected well, which is further colored in a Yellow-Black gray scale for emphasis, and the dash-lined rounded corner rectangles (B7, D9, D10 and H8), which are colored in a Blue-Black gray scale, represent other samples that are replicates of C8. When C8 was selected, an event was triggered that caused the program to automatically annotate the selected wells on the Standard Curve graph using their absorption values. The red squares in the graph represent the values for the square standard wells and the blue tagged circles represent the selected and replicate wells.

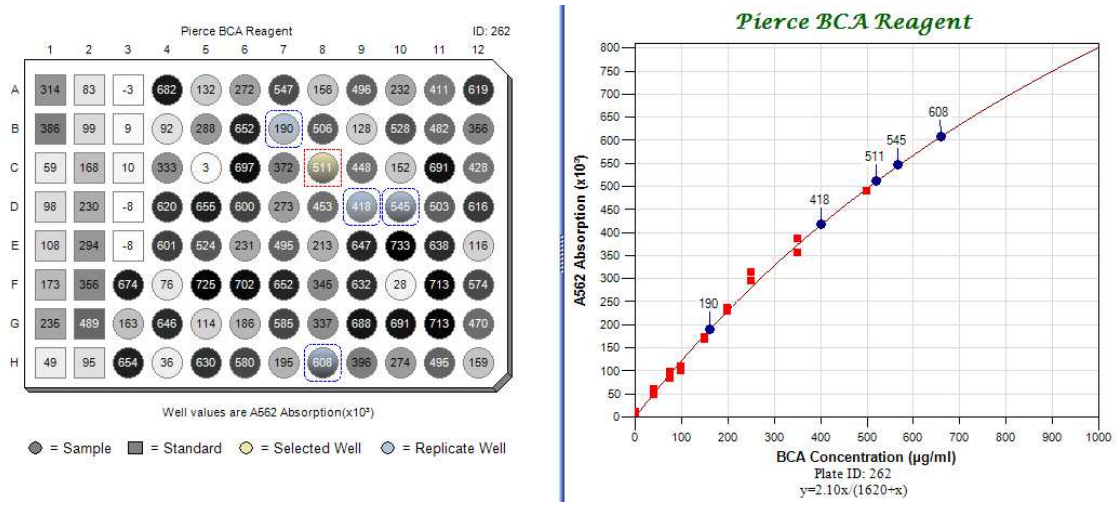


Figure 1: Microtitre plate and a Standard Curve

Figure 2 shows how the BxMicroTitrePlate control might be used in a workflow. The plate is used at the beginning of the workflow to represent the structure/content to be analyzed and then is used to show analyzed results. Subsequently, additional data/information can be retrieved or saved from/to persistent storage. The Magnifier view, a selectable plate control option, is used to compare a set of selected and replicate wells. The plate can be copied to the clipboard and incorporated into printed results and views.

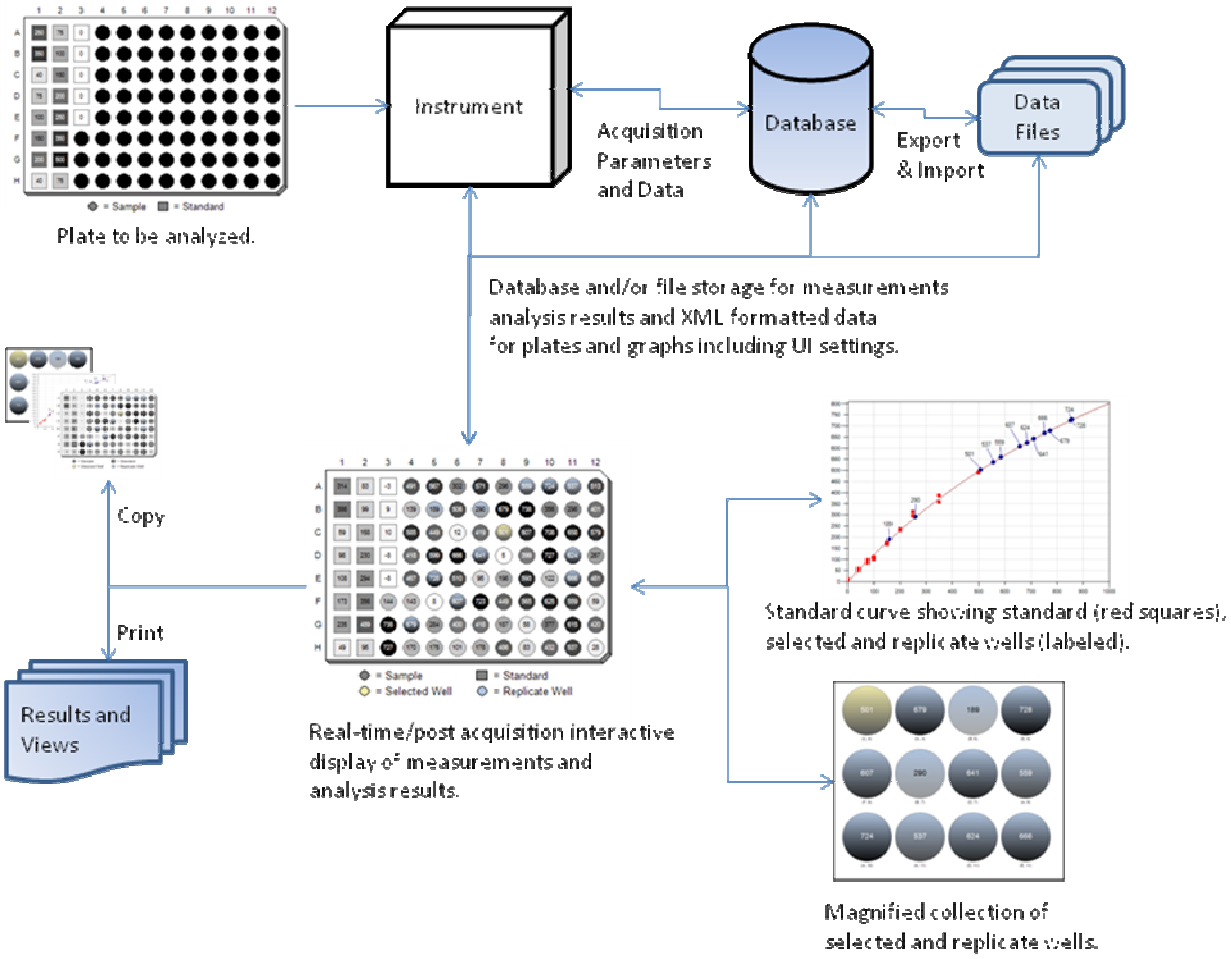


Figure 2: Example workflow showing use of BxMicroTitrePlate

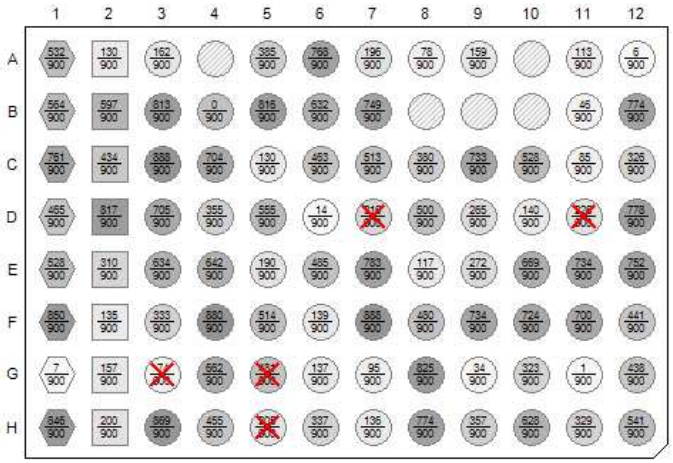


Figure 3 shows a 96-Well microtitre plate view with three different well shapes that could represent Standards, Controls and Samples. It also shows wells that are empty using a cross-hatch pattern and wells that are marked excluded with red X's. Filled wells have a gray-scale pattern showing magnitude and the well interior annotation displays a ratio of a measured magnitude to a maximum value. The plate shape has a Bottom-Right corner cut.

Figure 3: View Showing Excluded and Empty Wells

- The class provides properties for:
    - Selecting standard plate sizes
    - Specifying a custom plate row-column size that could be used for representing rectangular fraction collectors.
    - Specifying well types
        - Samples
        - Controls
        - Standards
        - Tags (fluorescent)
        - Buffers
        - References
        - Probes
        - Targets
        - Other
    - Assigning wells to groups or as replicates which can then be graphically indicated when any member of the group or replicate is selected.
    - Specifying Fonts and Colors for different components of the plate
    - Specifying a Plate ID, Title, Subtitle and Caption
    - Specifying Well Shapes for well types, interior fill patterns, annotation and tooltips
    - OwnerDraw for wells to draw own well shape and/or content such as images or graphs.
    - Specifying overall appearance such as whether an ID, Title, Subtitle, Caption, Row-Column Labels or Legend is to be shown or not.
    - Specifying location of corner cuts.
    - Marking wells as empty, excluded or hidden

  - Rendering on any graphics object.
  - Defining tooltips that are displayed as the mouse is moved over a well
  - Specify Number Formats for different types of well annotation
  - Displaying of graphs and images within wells.
  - Versatile interactive selection of one or more wells
  - Well Magnifier to obtain more detail of an image or graph or to view comparisons of a collection of wells.
  - One or more wells can be blinked and/or individually colored to represent well activity.
  - Drag-Drop of images and text files for 2D graphs.

- The class provides methods for:

  - Copying the plate view to the clipboard
  - Saving the plate view to a file in a specified image format
  - Printing the plate view
  - Getting a bitmap image of the plate view
  - Saving and Loading of plate customizing property settings to/from an XML File
  - Saving the plate DataTable schema to an XML File.

- The class provides events for:

  - Mouse clicks and movement for user interaction.
  - Customization changes
  - Errors.

3 Terminology and Features

Figure 4 illustrates the various parts of the MicroTitrePlate control. The contrasting colors were chosen to accentuate parts and to show customization capability. The annotation of the parts provides the terminology used within the document and in the API. The borders of the Title, Subtitle, Plate and Caption areas are colored white while the Container border is black. The plate and its wells are automatically scaled maintaining a square aspect ratio for wells as the size of the Drawing and Plate areas change. The square aspect ratio means, for example, that circular wells will not have an elliptical shape. In addition, a legend (not shown) can be placed horizontally below the caption or vertically to the right of the plate.

3.1 Terminology

Annotated Well	Each well can have annotation text within it such as the

	magnitude of the X-Value or Y-Value.
BxPlot2D	This is an add-on simple graphing component that is used to render two-dimensional (XY) data in a graph. The graph can be customized and annotated.
Caption Area	The caption area can have multiple lines that span the width of the plate and has these settable properties: • Horizontal Alignment • Font, Font Size and Font Style • Text Color and BackColor
Container or Drawing Area	The container is an area that contains all components of the plate. The background color of the container can be specified where typically it matches the control that it is attached to. The container is also referred to as the drawing area.
Corner Cut	Any and all corners can have a corner cut as illustrated.
Drag-Drop	The control supports drag-dropping of image file and text files that contain XY data for graphs.
ID Area	The ID area is single line whose width is based upon the length of the ID and has these settable properties: • Horizontal Alignment is either at the left-side or right-side within the Title Area. • Font, Font Size and Font Style • Text Color and BackColor
Plate	The plate is the area that contains the wells.
Row-Column Indexes	Row-Column indexes are the indexes for the location of a well and they correspond to the standard plate nomenclature unless it is overridden through a property option.
Subtitle Area	The subtitle area is a single line that spans the width of the plate if in single plate mode or spans half the width of plate if in Two-Plate mode and has these settable properties: • Horizontal Alignment • Font, Font Size and Font Style • Text Color and BackColor
Title Area	The title area is a single line that spans the width of the plate minus the width of the ID area and has these settable properties: • Horizontal Alignment • Font, Font Size and Font Style • Text Color and BackColor
Well Fill Style	The gray-scale well fill style is used to represent the magnitude of the Y-Value or X-Value. Other fill styles can be used to represent other states of the well such as being empty.

Well Magnifier Viewer	This is an embedded control or a Popup Windows Form that is used to magnify one or more selected wells. The embedded control is designed to be used as an integral part of the application while the Popup Windows form is designed to be function outside to the application. The content of the magnifier can be copied to the clipboard.
Well Shapes	The shape of a well can be different from the usually used circle as illustrated in Figure 4 where wells have shapes of Squares. Shapes can be used to represent different types of well content such as Standards, Controls and Samples.
XML Settings File	All of the property values used in defining and customizing the plate can be exported and imported through an XML file. This XML file can be used to initialize plates in multiple applications to provide the same appearance.

Table 1: Description of Terminology.

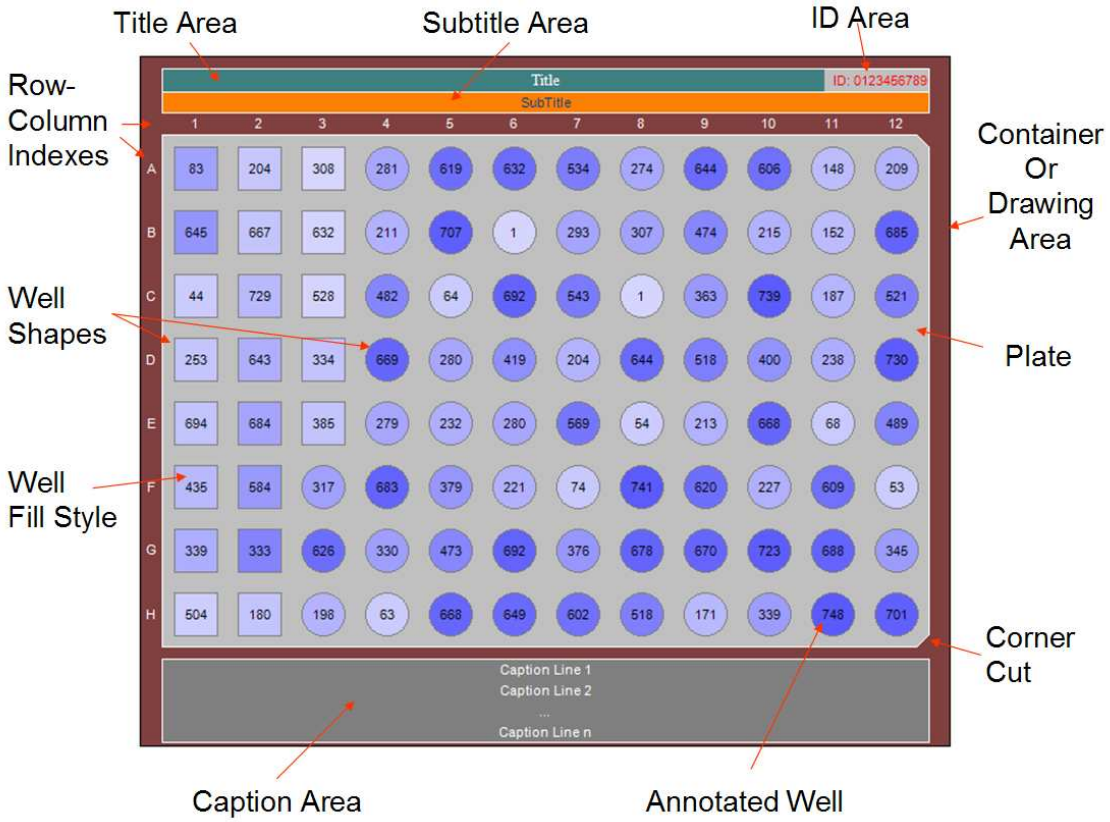


Figure 4: Annotated Figure detailing terminology

4 Programming API

The structure of BxMicroTitrePlate is initialized and controlled by values contained in a .NET DataTable object while the appearance is controlled primarily through Properties and a few Set methods. The DataTable must have a minimum schema (4.2.5.1) which can be obtained through the API methods defined in section 4.2.3. The minimum schema can be expanded, but only the minimum schema will be processed by the MicroTitrePlate control. Internally the control will validate an inputted DataTable for the minimum schema and for population of key fields within the schema. An error event (4.5.1) will be triggered if validation fails.

Figure 5 shows an example application containing the BxMicroTitrePlate control with the options selected for showing embedded and popup magnifier viewers. Two-dimensional data linked to the selected well is rendered as a XY Graph using BxPlot2D in the viewers. A database and a collection of data files are connected to the plate using an internal .NET DataTable object that contains data and/or links associated with each well. The plate image can be copied to the clipboard and printed within reports containing results and views. In addition the graph with its annotation in the embedded viewer can also be copied and printed. Multiple wells can be selected for displaying in the viewers.

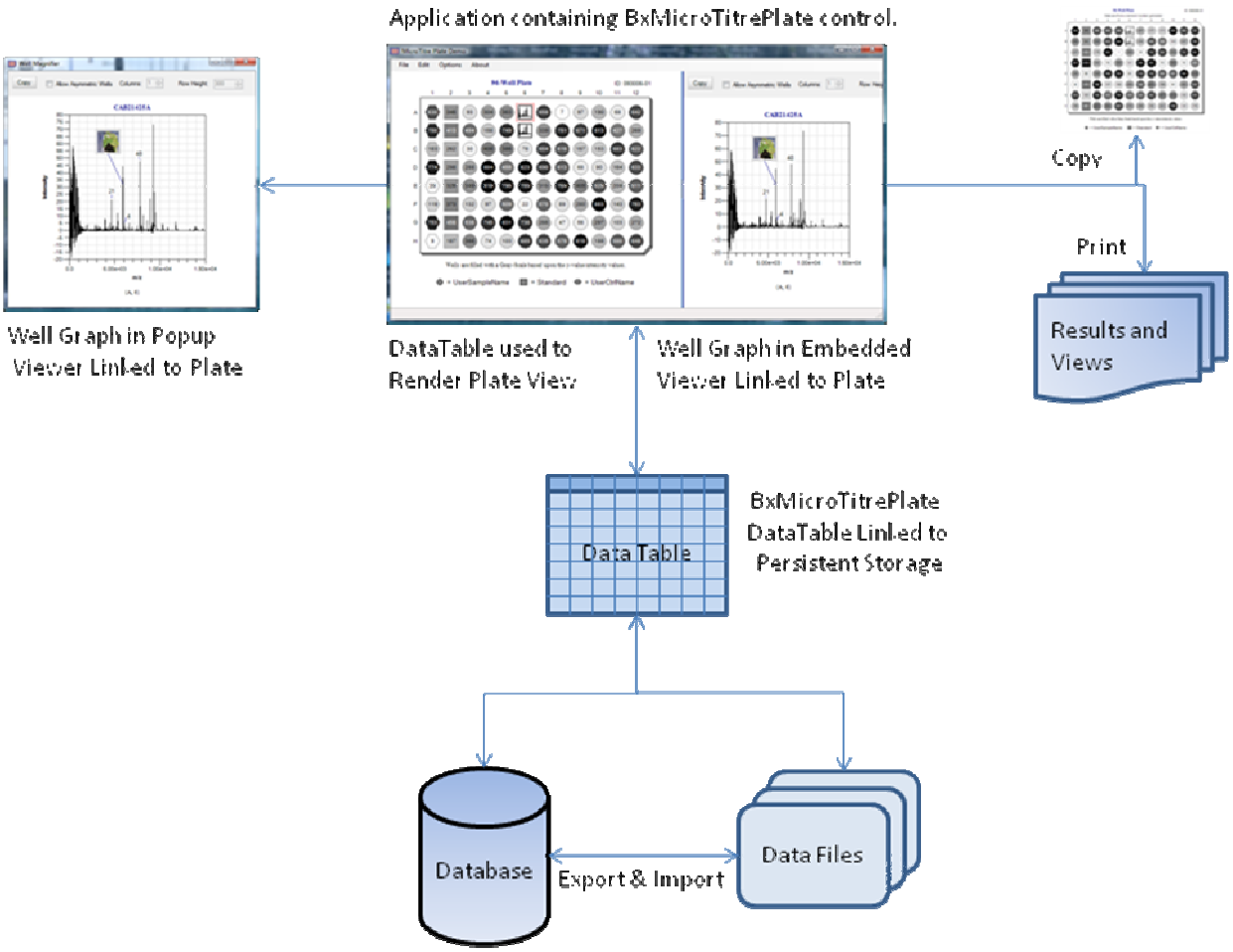


Figure 5: View showing DataTable and Well Magnifier Viewers connected to Plate

4.1 Instantiation of MicroTitrePlate

4.1.1 Required Dynamic Link Libraries

The system requires four dynamic link libraries:

1. BxMicroTitrePlate.dll contains code for plate rendering and management.
2. BxAlignment.dll contains code for graphically retrieving alignment specifications and is used by the Customization Dialog Box
3. BxValidators.dll contains code used for validating entries in the Customization Dialog Box.
4. BxPlot2D.dll contains code for rendering XY data in a graph.

4.1.2 Example Instantiation code

The following code shows an instantiation of an instance of the MicroTitrePlate control and adding it to a Controls collection. It assumes that BxMicroTitrePlate.dll, BxValidators, BxComponents, BxPlotControls, and BxAlignment.dll have been added to the code project which is going to use the control. In Visual Studio this is done by adding it to the list of Reference objects for a project.

Note: The code in this section is contained in BxMicroTitrePlateDemo Visual Studio 2008 project that comes with the system. The BxMicroTitrePlateDemo application shows how to instantiate the plate control, create and populate a DataTable used by the control, Load previously saved property settings, copy to the clipboard, print the plate view and to customize the plate view by using the Customization Dialog Box.

```
// MicroTitrePlate is contained in the BxTools namespace.
BxTools.MicroTitrePlate mtp = new BxTools.MicroTitrePlate();
```

NOTE: Throughout this document 'mtp' will refer to the plate control object

```
// the Dock Style for the plate can be set to fill the entire client
// area to which it is attached as follows
mtp.Dock = DockStyle.Fill;
```

Setting the Dock style to Fill will cause the MicroTitrePlate to fill the entire area of the parent and will be automatically sized as the parent size changes. Other DockStyles can be used and the plate control can be managed using control Size, Location and Anchor properties such as:

```
//mtp.Dock = DockStyle.None;
//mtp.Location = new Point(50, 50);
//mtp.Size = new Size(300, 300);
//mtp.Anchor = AnchorStyles.Top | AnchorStyles.Left |
//             AnchorStyles.Right | AnchorStyles.Bottom;


  Controls.Add(mtp);
```

```
// Create an Empty Plate DataTable, that is, the plate view is
// of an empty plate since no well data has been assigned to any
// DataRow

     mtp.CreateEmptyPlateDataTable(maxRows, maxCols);

// retrieve the plate that was created and assign to program member
     plateTable = mtp.PlateDataTable;
```


Note: More data columns can be added to the plateTable for connecting other information to each well for the application to use. The plate control will not use these columns.

```
// ***************Alternative Snippet ***************************************
//
// note: Instead of doing the above two statements to create an Empty Plate DataTable,
//       the following code snippet could be used.

//plateTable = mtp.GetMicroTitrePlateBaseDataTable();

//// note: More Data columns can be added to the plateTable for connecting other
////       information to each well for the application to use.
////       The plate control will not use these columns but

//int tableRow = 0;

//DataRow dr;

//// note: only the following fields are defined
////       WellNumber, WellLocation, N, Row, Column and Type.
////       The type field has a WellType of Empty.
////
////       The other fields are programmatically defined by calling the
////       FillPlateDataTable() method or
////       they could have been defined here.
//for (int nCols = 1; nCols <= maxCols; nCols++)
//    for (int nRows = 0; nRows < maxRows; nRows++)
//    {

//        // populate a row and add it to the table
//        dr = plateTable.NewRow();
//        dr["WellNumber"] = tableRow + 1; // 1 to total rows
//        dr["WellLocation"] = ((maxRows <= 26) ?
//                    Convert.ToString(Convert.ToChar(Convert.ToInt32('A') + nRows))
//                                + nCols.ToString()
//                                : (nRows + 1).ToString() + ", " + nCols.ToString());
//        dr["N"] = tableRow++; // 0 to total rows minus 1
//        dr["Row"] = nRows + 1;
//        dr["Column"] = nCols;
//        dr["Type"] = mtp.GetNameOfWellType(BxTools.MicroTitrePlate.WellTypes.Empty);

//        // note: The other fields could have been assigned data here
//                  instead of calling FillPlateDataTable()
//
//        plateTable.Rows.Add(dr);


//    }


//// assign Empty plateTable to plate control
//mtp.SetPlateDataTable(plateTable, maxRows, maxCols);

// ***********  end of alternative create empty DataTable snippet.
// ********************************************************************************




// *********** Intitialize plate property values.
 mtp.UseUserWellTypeNames = true;
 mtp.ShowLegend = true;
 mtp.PlateWellColor = Color.Gray;
```



```
 mtp.PlateWellPrimaryColor = Color.Gray;
 mtp.WellFillStyle = BxTools.MicroTitrePlate.WellFillStyles.Gray_Scale;
 mtp.HighlightSelectedWellsWithColors = true;
 mtp.HighlightSelectedWellsWithDashLineRectangles = true;
 mtp.RotatePlate = false;
 mtp.ShowRowColLabels = true;
 mtp.ShowColLabelsAtBottom = false;


 // the plate can display a Title, ID, Subtitle and Caption
 mtp.PlateTitle = "96-Well Plate";
 mtp.ShowTitle = true;

 mtp.PlateId = "xxxx";
 mtp.ShowPlateID = true;

 mtp.PlateSubTitle = string.Empty;
 mtp.ShowSubTitle = false;

 mtp.PlateCaption = "Plate shows all Empty wells.\r\nClick on 'Fill Plate' to populate
it.";
 mtp.CaptionAlignment = ContentAlignment.MiddleCenter;
 mtp.ShowCaption = true;


// Subscribe to events - make sure that the subscription is only done once.
mtp.Error += new BxTools.ErrorEventHandler(mtp_Error);

// subscription to this event is necessary if the controls internal context menu is used.
// that is, if UsePlateContextMenu is set to true which is the default.
//
/ mtp.UsePlateContextMenu = false;  // set to false to not use the control's context
menu.
mtp.CustomizationEvent +=
    new BxTools.MicroTitrePlateCustomizationEventHandler(mtp_CustomizationEvent);


// other events that can be subscribed to
//
//mtp.mtpMouseDown += new MicroTitrePlateMouseDownEventHandler(mtp_mtpMouseDown);
//mtp.mtpMouseUp += new MicroTitrePlateMouseUpEventHandler(mtp_mtpMouseUp);
//mtp.mtpMouseMove += new MicroTitrePlateMouseMoveEventHandler(mtp_mtpMouseMove);
//mtp.mtpMouseEnter += new MicroTitrePlateMouseEnterEventHandler(mtp_mtpMouseEnter);
//mtp.mtpMouseLeave += new MicroTitrePlateMouseLeaveEventHandler(mtp_mtpMouseLeave);
//mtp.mtpBlinkingStarted +=
//new MicroTitreWellBlinkingStartedEventHandler(mtp_mtpBlinkingStarted);
//mtp.mtpBlinkingStopped +=
//        new MicroTitreWellBlinkingStoppedEventHandler(mtp_mtpBlinkingStopped);
//mtp.AddedXYData += new MicroTitreAddXYDataEventHandler(mtp_AddXYData);
//mtp.AddedImage += new MicroTitreAddImageEventHandler(mtp_AddImage);
//mtp.RemovedImage += new MicroTitreRemoveImageEventHandler(mtp_RemoveImage);
//mtp.RemovedXYData += new MicroTitreRemoveXYDataEventHandler(mtp_RemoveXYData);
//mtp.DrawWellItem += new DrawWellItemEventHandler(mtp_DrawWellItem);
```

After the Empty Plate DataTable has been created, then the DataTable can be populated and plate properties specific to the data can be set. As the data changes only the plateTable needs to be updated. If the size of the plate changes, it may be necessary

The following method provides an example.

```
/// <summary>
/// This method simulates filling of a plate DataTable
/// </summary>
/// <param name="plateTable">DataTable to be filled</param>
private void FillPlateDataTable()
{
```



```
mtp.PlateId = "1234";
mtp.PlateTitle = (mtp.Rows * mtp.Columns).ToString() + "-Well Plate";

mtp.PlateSubTitle = "Data are from a random number generator";
mtp.ShowSubTitle = true;

mtp.PlateCaption = "Plate shows Control, Standard and Sample wells using"
       + " different shapes\r\nand intensity using a gray-scale.";



int tableRow = 0;

string strType = "Sample";
string strID = "ID";
string strIndex;
int irValue;

// Random numbers are used to populate YValues and
// for assigning replicate and group numbers.
Random dValue = new Random();
double dblValue = 0.0;
Random rValue = new Random();


DataRow dr;

for (int nCols = 1; nCols <= maxCols; nCols++)
    for (int nRows = 0; nRows < maxRows; nRows++)
    {

        strIndex = ((nRows) * maxCols + nCols).ToString();

        // the switch is used to assign three types of wells: Controls,
        // Standards and Samples.
        // and to specify an ID for its contents.
        if (nCols == 1)
            irValue = (int)BxTools.MicroTitrePlate.WellTypes.Control;
        else
            if (nCols == 2)
                irValue = (int)BxTools.MicroTitrePlate.WellTypes.Standard;
            else
                irValue = (int)BxTools.MicroTitrePlate.WellTypes.Sample;

        switch (irValue)
        {

            case (int)BxTools.MicroTitrePlate.WellTypes.Control:
                strType = "UserCtrlName"; // assign user name for Control well type
              // strType =  mtp.GetNameOfWellType
              //                (BxTools.MicroTitrePlate.WellTypes.Control);
                strID = "Ctrl-" + strIndex;
                break;
            case (int)BxTools.MicroTitrePlate.WellTypes.Standard:
                strType = mtp.GetNameOfWellType
                       (BxTools.MicroTitrePlate.WellTypes.Standard);
                strID = "Std-" + strIndex;
                break;
            case (int)BxTools.MicroTitrePlate.WellTypes.Sample:
                strType = "UserSampleName"; // assign user name for Sample well type
              // strType =  mtp.GetNameOfWellType
              //                (BxTools.MicroTitrePlate.WellTypes.Sample);
                strID = "S-" + strIndex;
                break;
        }

        dblValue = dValue.Next(900);
        dblValue = dValue.Next(900);

        // retrieve a row and populate it with data
```



```
            dr = plateTable.Rows[tableRow++];
            //
            // Note: these fields were already defined in the InitMicroTitrePlate()
            //       or DefineNewPlateDataTable() methods
            //
            //dr["WellNumber"] = tableRow + 1; // 1 to total rows
            //dr["WellLocation"] = ((maxRows <= 26) ?
            //          Convert.ToString(Convert.ToChar(Convert.ToInt32('A') + nRows))
            //                        + nCols.ToString()
            //                         : (nRows + 1).ToString() + ", "
            //                        + nCols.ToString());
            //dr["N"] = tableRow++; // 0 to total rows minus 1
            //dr["Row"] = nRows + 1;
            //dr["Column"] = nCols;


            // populate row fields
            dr["Type"] = strType;
            dr["ID"] = strID;
            dr["Description"] = ((strID == string.Empty) ? string.Empty : "this is a "
                                 + strType.ToLower());
            dr["XValue"] = ((strID == string.Empty) ? Convert.DBNull : tableRow);
            dr["YValue"] = ((strID == string.Empty) ? Convert.DBNull : dblValue);
            dr["WellAnnotation"] = (((strID == string.Empty)) ? Convert.DBNull
                                    : dblValue.ToString("0"));
            dr["Exclude"] = 0; // exclude well (1) or not (0)
            dr["Empty"] = 0; // designate well as being empty (1) or not (0)
            // integer value used to connect replicate wells
            dr["Replicate"] = ((strType == "UserSampleName") ? rValue.Next(25) : 0);
            // integer value used to connect group wells
            dr["Group"] = ((strType == "UserSampleName") ? rValue.Next(25) : 50 + nCols);


        }

    // assign user names for well types
    mtp.SetWellTypeName(BxTools.MicroTitrePlate.WellTypes.Control, "UserCtrlName");
    mtp.SetWellTypeName(BxTools.MicroTitrePlate.WellTypes.Sample, "UserSampleName");

    // assign the Diamond shape to the Control well type
    mtp.ShapeControl = BxTools.MicroTitrePlate.WellShapeStyles.Diamond;
    // assign the Hexagon shape to the Standard well type
    mtp.ShapeStandard = BxTools.MicroTitrePlate.WellShapeStyles.Hexagon;
    // assign the SquareFrame to the Sample Well type
    //mtp.ShapeSample = BxTools.MicroTitrePlate.WellShapeStyles.SquareFrame;

}
```

4.1.3 Creating a Very Large View of a Plate

A technique for creating a very large view of a plate, one that uses horizontal and vertical scrollbars, is shown in the following code snippet. This can be useful, for example, to increase the well size for making well annotation readable.

```
// Create panel object that will contain the plate and
// have scroll bars.
System.Windows.Forms.Panel panelMTP = new System.Windows.Forms.Panel();
panelMTP.Size = new Size(300,300);   // initial size of panel
panelMTP.AutoScroll = true;
paneLMTP.Dock = DockStyle.Fill;
```



```
// add panel to application form
Controls.Add(panelMTP);


// MicroTitrePlate is contained in the BxTools namespace.
BxTools.MicroTitrePlate mtp = new BxTools.MicroTitrePlate();
mtp.Dock = DockStyle.None;
mtp.Location = new Point(10, 10); // small offset if desired.

// plate is much larger than panel forces scrollbars as long as panel is smaller
mtp.Size = new Size(600, 600);
// only anchor the Top, Left
mtp.Anchor = AnchorStyles.Top | AnchorStyles.Left ;

// Add plate to panel object.
panelMTP.Controls.Add(mtp);
```

4.2 Methods

4.2.1 General Methods

The follow methods are provided for drawing the plate on any graphics device, copying to the clipboard, saving it to a file in a specified image format and obtaining a plate image for other application uses such as displaying on a web page. Each of these methods is overloaded which means that they accept different sets of arguments such as the graphics object and size of the drawing area or image.

In methods where the size is not specified, the size of the image will be roughly 800x650 or 650x800 depending upon whether the plate is larger in the horizontal or vertical direction respectively and it will have whitespace minimized around the plate. In addition methods where the size is specified, if the allowResize parameter is true, then the returned image may be smaller due minimizing whitespace around the plate.

Method	Return	Description
Clone()	MicroTitrePlate	This method is used to clone the current MicroTitrePlate control including all of its data and property settings.
CopyToClipboard(…)	void	Copy a redrawn microtitre plate with all of its current property settings fitted within the image Size parameters to the clipboard. If a size is not provided then 800 x 650 pixels will be used.

CopyDataTableToClipboard(…)	string	This method copies tab delimited text containing the column caption names and the column data for each row in the currently selected plate DataTable to the Clipboard. An overloaded method accepts a columnList for specifying which columns and their order to copyt to the clipboard.
CopyViewToClipboard()	Void	Copy the current plate GUI view that has the current size of the control to the clipboard. The result is an image on the clipboard that matches the current view including clipped legends and text. Other methods that specify sizes are available to create larger images that will contain a redrawn plate view that do not contained clipped regions. The CopyToClipboard() method has an inherent size of 800 x 650 and the other overloaded methods permit size specifications.
DrawLegend()	Size	Draws a legend containing each WellShapeType in the plate. The legend can be drawn in the vertical or horizontal direction with or without a border.
DrawTable(…)	void	Draws the plate on a graphics device using the drawing area and positioning parameters. In addition a Boolean argument is available for simulation.
ExportPlateDataTable	Void	This method exports a tab delimited text file containing the column caption names and the column data for each row in the currently selected plate DataTable. An ArrayList specifies the columns to be printed and their order.
GetMicroTitrePlateImage(…)	Image	Get an Image of a redrawn

		microtitre plate with all of its current property settings fitted within the image Size parameters. If a size is not provided then 800 x 650 pixels will be used.
GetMicroTitrePlateImageView(…)	`Image`	Get an Image of the current GUI view that has the current size of the control. The result is an image that matches the current view including clipped legends and text. NOTE: Since this image can be quite large, it may not be suitable for printing and one of the other methods where a size is specified should be used to obtain an image that fits within the print drawing area. The GetMicroTitrePlateImage() method has an inherent size of 800 x 650 and the other overloaded methods permit size specifications.
GetPlateDataTableAsText	`String`	This method Gets tab delimited text containing the column caption names and the column data for each row in the currently selected plate DataTable.The ArrayList specifies the columns to be retrieved and their order.
SaveMicroTitrePlateImage (string filename, ImageFormat imageFormat)	`bool`	This method is used to save the current plate image to the specified file in the specified image format. If the filename extension does not match the image format extension or contain an extension, then the file name extension will be added. The size of the image will be roughly 800x650 or 650x800 depending upon whether the plate is larger in the horizontal or vertical direction respectively and it will have whitespace minimized around

		the plate. The supported formats are: .Bmp, .Emf, .Gif, .Jpeg, .Png, .Tiff, .Wmf

Table 2: General Purpose Methods

4.2.2 Draw Legend

4.2.2.1 Automatic Drawing of Legend

The system contains properties that initiate automatic drawing of the legend within the plate drawing area. The legend can be drawn vertically on the right side of the plate or horizontally below the plate with and without a border drawn around it. In addition the alignment of the legend keys within the region containing the legend can be set. These property values are also used when copying to the clipboard and obtaining an image that can be used in printing or rendering on any graphics device.

In addition, when UseIndividualWellColors is set to true and Replicates and Group wells are shown then the plate legend will display the color key for them if the property IncludeReplicateAndGroupWellsInLegend is set to true.

Property	DataType	Description
LegendKeyAlignment	ContentAlignment	Set or Get ContentAlignment for rendering of the legend keys within its drawing rectangle. Default is MiddleCenter.
ShowLegend	bool	Set or Get boolean value for whether to show plate legend (true) or not (false). Default is false.
ShowLegendBorder	bool	Set or Get boolean value for whether to show plate legend Border(true) or not (false). Default is false.
ShowLegendVertical	bool	Set or Get boolean value for whether to show plate legend Vertical(true) or not

		(false). Default is false.
IncludeReplicateAndGroupWellsInLegend	Bool	Set or Get boolean value for whether to show in the plate legend Replicate and Group Well color keys.

Table 3: Automatic Draw Legend Properties

4.2.2.2 Examples of Drawing Legend

The following figures illustrate drawing of horizontal and vertical legends respectively. In the horizontal case, the legend shows how items are wrapped into multiple rows. The last figure shows Replicate and Group keys in the legend.

Settings for Figure 6:

```
mtp.ShowLegend = true;
mtp.ShowLegendVertical = false;  // draw horizontal
mtp.ShowLegendBorder = false;
```

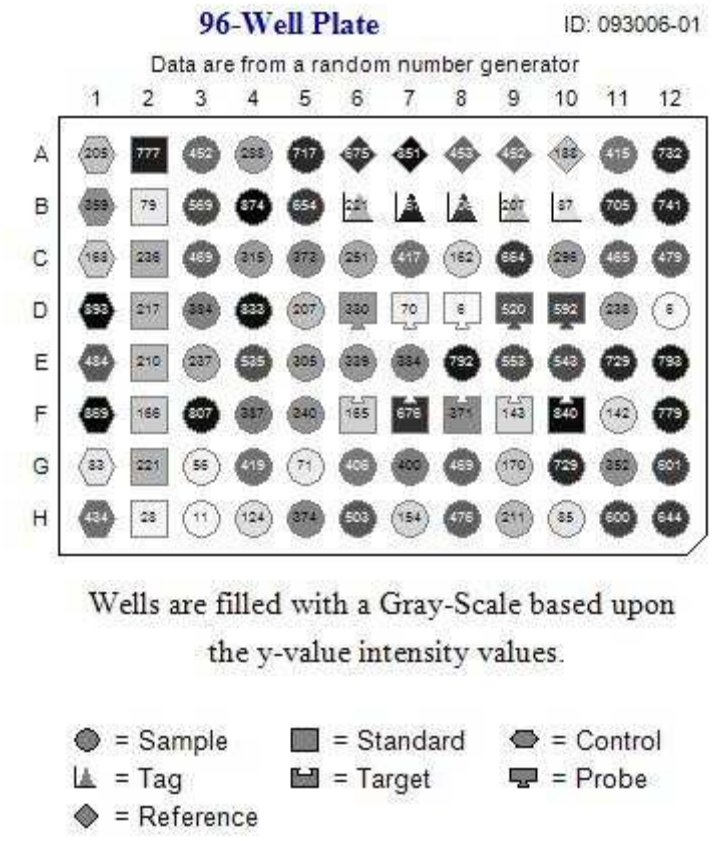


Figure 6: Figure shows horizontal legend with multiple rows.

Settings for Figure 7:

```
mtp.ShowLegend = true;
mtp.ShowLegendVertical = true;  // draw vertical
mtp.ShowLegendBorder = false;
```

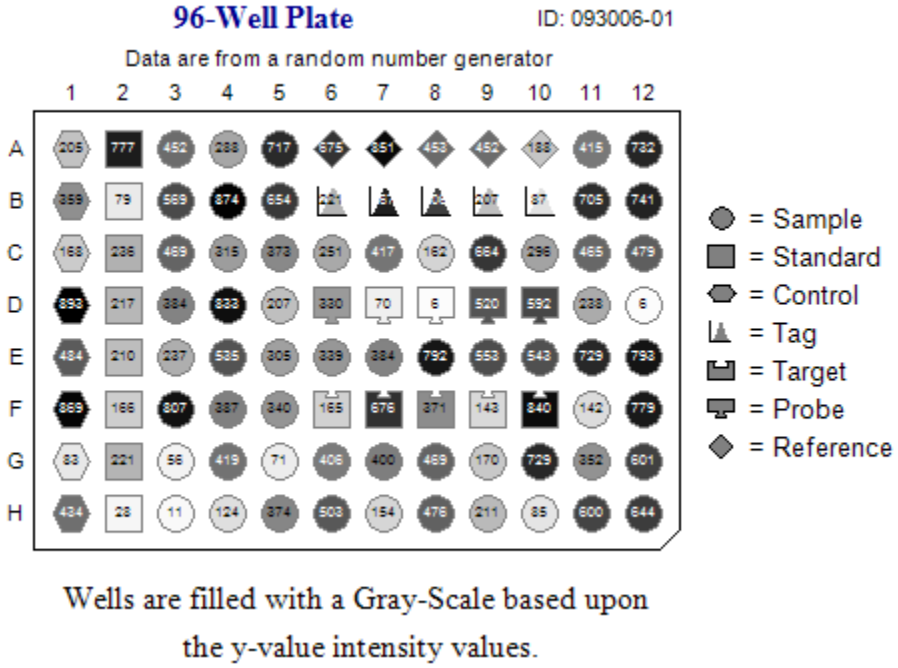


Figure 7: Figure shows Vertical Legend on the right side of plate.

Settings for Figure 8:

```
mtp.ShowLegend = true;
mtp.ShowLegendVertical = false;  // draw horizontal
mtp.ShowLegendBorder = false;
mtp.IncludeReplicateAndGroupWellsInLegend = true;
mtp.ShowGroup = true;
mtp.ShowReplicates = true;
mtp.UseIndividualWellColors = true;
// specify mapping of user well type names to internal well type names.
mtp.SetWellTypeName(MicroTitrePlate.WellTypes.Sample, "UserSampleName");
```


mtp.SetWellTypeName(MicroTitrePlate.WellTypes.Control, "UserCtrlName");

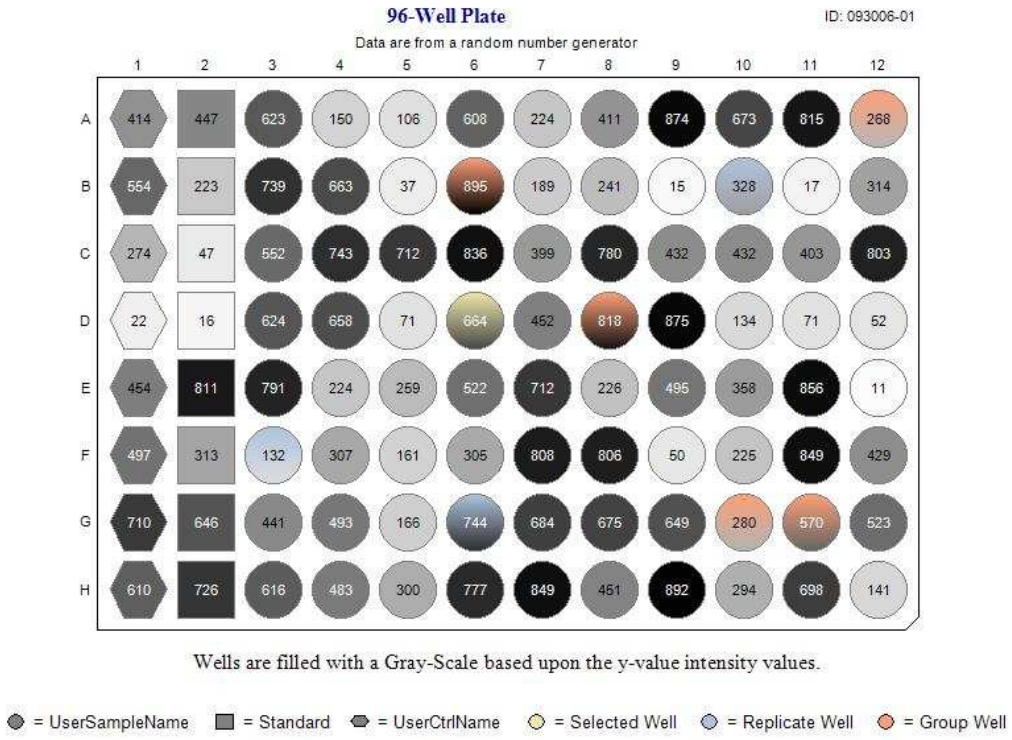


Figure 8: Figure shows legend with Replicate and Group colored keys

4.2.2.3 Manually Drawing Legend

The manual approach to drawing the legend is typically used when the plate needs to be maximized and/or when it needs to be drawn in a specific location or in a separate window. The ShowLegend property should be set to false and the developer code should explicitly call the DrawLegend() methods as illustrated below.

Figure 9 shows an example of a horizontally drawn Legend for a plate containing 5 different well types. The DrawLegend method returns a size structure for the legend and has a parameter for whether to simulate drawing of the Legend or rendering it. If the legend is to be centered either vertically or horizontally with respect to the plate, then it could first be simulated to obtain its Size and then rendered specifying its centered position.

```
// call the DrawLegend method with the last argument set to false to get the
// size structure used to position the legend n the next DrawLegend call that actually
// renders it. mtp is a BxMicroTitrePlate object.
Size legendSize = mtp.DrawLegend(e.Graphics,
      new Point(e.MarginBounds.X
        , e.MarginBounds.Y + mtp.PlateContainerHeight + 20)
```



```
        , false, true, false);
legendSize = mtp.DrawLegend(e.Graphics,
      new Point(mtp.PlateLocation.X + (mtp.PlateWidth -
        legendSize.Width) / 2
        , e.MarginBounds.Y + mtp.PlateContainerHeight + 20)
        , false, true, true);
```

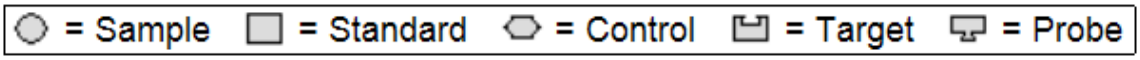


Figure 9: Example of a Legend drawn horizontally.

4.2.3 DataTable Schema

The input to the MicroTitrePlate control is a DataTable that must contain a minimum schema. Additional columns can be added and will be ignored by the control; however, none of the minimum schema columns can be omitted. There are two methods built into the control that can be used to obtain the minimum schema.

Method	Description
GetMicroTitrePlateBaseDataTable()	This method can be called directly to create and obtain a DataTable containing the minimum schema. The Return value is a DataTable.

WriteMicroTitrePlateData BaseSchema(string filename)	This is an indirect method in that it will write an XML file containing the schema to the directory and name specified by the filename argument. That is, the filename argument can be a full path name. This XML file could then be used as follows:
	
	`string fn = "MTPSchema.xml";`
	`// make sure that it does not exist before`
	`// writing schema to disk storage.`
	`if (!File.Exists(fn))`
	`mtp.WriteMicroTitrePlateDataTableBaseSchema(fn);`
	
	`// Initialize a DataTable with previously`
	`// written schema.`
	`DataTable MTPTable = new DataTable();`
	`MTPTable.ReadXmlSchema(fn);`

CreateEmptyPlateDataTable (int NbrOfRows, int NbrOfColumns)	This method creates an empty DataTable with a DataRow for each well in the plate. The plate view is of an empty plate since no well data has been assigned to any row. Only WellNumber, WellLocation, N, Row, Colmmn and Type within a DataRow are defined. The type field has a WellType of Empty. The other fields must be programmatically defined. After creating the DataTable it calls SetPlateDataTable(…) which sets the following properties: PlateDataTable, Rows and Columns. The calling application should create a DataTable object and initialize it with the control's PlateDataTable property. For example: DataTable plateTable = mtp.PlateDataTable;

Table 4: Methods for obtaining the minimum DataTable schema

4.2.4 Assigning DataTable to BxMicroTitrePlate User Control

Once a DataTable is created and populated then it can be assigned to the User Control using the PlateDataTable properties. For example:

```
// Create a DataTable with the minimum schema
DataTable MTPTable = GetMicroTitrePlateBaseDataTable();


// then call the application's fill method which adds a DataRow for each well in the plate
// and sets the DataRow Field values for each well.
FillDataTable(MTPTable);


// assign the DataTable to the control.
mtp.SetPlateDataTable (MTPTable, <Number of Rows>, <Number of Columns>);

mtp.Refresh();
```

or

```
// Create a DataTable for an Empty Plate.
//  The returned plate has already been assigned to the control
MTPTable = CreateEmptyPlateDataTable(<Number of Rows>, <Number of Columns>);

// Fill each well's DataRow with data
FillDataTable(MTPTable);
```



```
// Refresh the control to show the result of populating the DataTable.
mtp.Refresh();
```

NOTE:

Once the DataTable is populated it is better to use the SetDataTable…() methods described in the next section to assign the DataTable.

4.2.5 Set Plate DataTable Methods

The methods in Table 5 enable setting of the DataTable, Rows and Columns. The methods return true if all values were set; otherwise false is returned. An error event is also triggered with error message 24:

```
Plate DataTable was not set which could be due to an empty
DataTable or a Row-Column specification error.
```

The following code snippet illustrates the use of the different methods for assigning DataTable MTPTable. It assumes that the 96, 384 and 1536 well plates have standard plate dimensions; otherwise, use the method where the rows and columns are specified.

```
bool success;

if (MTPTable.Rows.Count == 96)
   success = mtp.SetDataTable96Wells(MTPTable);
else
    if (MTPTable.Rows.Count == 384)
       success = mtp.SetDataTable384Wells(MTPTable);
    else
        if (MTPTable.Rows.Count == 1536)
            success = mtp.SetDataTable1536Wells(MTPTable);
        else // custom plate.
            success = mtp.SetDataTable(MTPTable, rows, cols);


if (!success)
{
    // perform error recovery.
    // note that there is also an error event that can be used.
}
else
{
    // continue with normal code.
}
```

These methods should be used when dynamically switching from one size of plate to another since it ensures that property settings are performed in the proper order for the DataTable validation methods

If subsequent plate sizes do not change then only the mtp. PlateDataTable row values need to be changed; otherwise, the above sequence of steps needs to be used each time.

Method	Return	Description
SetPlateDataTable(DataTable, Rows, Columns);	bool	This method sets the DataTable and the number of rows and columns of a ***custom*** plate for Plate A. It can also be used for the standard well size plates. The number of rows and columns are used in the validation methods that determine DataTable integrity.
SetPlateDataTable96Wells(DataTable);	bool	This method sets the DataTable for Plate A using a 96-well structure. This method also sets the plate Rows and Columns values to 8 and 12 respectively.
SetPlateDataTable384Wells(DataTable);	bool	This method sets the DataTable for Plate A using a 384-well structure. This method also sets the plate Rows and Columns values to 16 and 24 respectively.
SetPlateDataTable1536Wells(DataTable);	bool	This method sets the DataTable for Plate A using a 1536-well structure. This method also sets the plate Rows and Columns values to 32 and 48 respectively.
SetPlateRowColumnSize(Rows, Columns)	Void	This method is used to set the row-column size of the plate or rectangular array of wells.

Table 5: List of Methods used to set a plates DataTable, Rows and Columns

4.2.5.1 Description of Columns in Minimum DataTable Schema.

Column	Data Type	Description
N	`int`	Number that spans the range from 0 to total number of

		wells minus 1. That is, it is a zero-based index value.
Row	int	Row index value for well
Column	int	Column index value for well

Type	string	The string can contain the internal name of a Well Type obtained from a WellTypes enum value or one defined programmatically through the SetWellTypeName(). This string is used to determine the shape of the well and can be used for programmatic functions. If string is null or empty then the Well Type Shape is automatically defined as the shape of a sample.
		
		The control provides a method that can be used to get the name of a WellTypes enum value. For example, if mtp is a MicroTitrePlate control and the string name for a Sample well type is needed, then the following code can be used.
		
		`MicroTitrePlate.WellTypes wt = MicroTitrePlate.WellTypes.Sample;`
		
		`mtp.GetWellTypeEnumName(wt);`
		
		The GetWellTypeEnumName() method returns the user defined name for the well type when property UseUserWellTypeNames is set to true or predefined internal names if false.

ID	string	ID that is used to identify content of well
Description	string	Description of well content.
XValue	double	X Value associated with the well that may be used when linking a well to a graph. For example, it may contain Values for Standards of known concentration and calculated values for samples.
YValue	double	Y Value associated with the well that may be used when linking a well to a graph. For example, it may contain measured values of fluorescence for wells.
WellAnnotation	string	Annotation that is used to appear on each well. It should be short enough to fit within the well borders.
Exclude	bool	True if well is to be excluded; otherwise false.
Empty	bool	True if well is empty (Not Filled); otherwise false
Replicate	int	This is a number that connects replicate wells together. That is, a specific well and all of its replicates will have the same number. This permits the GUI to highlight replicates for a selected well and populates a public ArrayList *ReplicateWells* property with a Well structure for each replicate well.
Group	int	This is a number that groups wells together. That is, a

		specific well and all members of the group will have the same number. This permits the GUI to highlight grouped wells for a selected well and populates a public ArrayList *GroupedWells* with a Well structure for each well in a group.

Table 6: Description of columns for minimum DataTable Schema.

4.2.5.2 DataTable from GetMicroTitrePlateDataTableBaseSchema()

The GetMicroTitrePlateDataTableBaseSchema() method returns a DataTable that can be used as an application's DataTable for the MicroTitrePlate control. For reference and creating a customized method the internal code for the method is shown below:

```
public DataTable GetMicroTitrePlateBaseDataTable()
{

        DataTable MTPTable = new DataTable("MicroTitrePlate");


        MTPTable.Columns.Add("WellNumber", typeof(int));
        MTPTable.Columns.Add("WellLocation", typeof(string));
        MTPTable.Columns.Add("N", typeof(int));
        MTPTable.Columns.Add("Row", typeof(int));
        MTPTable.Columns.Add("Column", typeof(int));
        MTPTable.Columns.Add("Type", typeof(string));
        MTPTable.Columns.Add("ID", typeof(string));
        MTPTable.Columns.Add("Description", typeof(string));
        MTPTable.Columns.Add("XValue", typeof(double));
        MTPTable.Columns["XValue"].Caption = "X Value";

        MTPTable.Columns.Add("YValue", typeof(double));
        MTPTable.Columns["YValue"].Caption = "Y Value";

        MTPTable.Columns.Add("WellAnnotation", typeof(string));
        MTPTable.Columns["WellAnnotation"].Caption = "Well Annotation";

        MTPTable.Columns.Add("Exclude", typeof(bool));
        MTPTable.Columns.Add("Empty", typeof(bool));
        MTPTable.Columns.Add("Replicate", typeof(int));
        MTPTable.Columns.Add("Group", typeof(int));

        return MTPTable.Clone();

}
```


Additional Columns can be added to the returned DataTable as shown below where a column with mapping name of 'NewCol' and data type of 'string' is added.

```
DataTable dtPlate = mtp.GetMicroTitrePlateBaseDataTable ();
dtPlate.Columns.Add(new DataColumn("NewCol", typeof(string));
```

Note that all additional columns will be ignored by the MicroTitrePlate control except when specifying columns for annotation and tooltips.

4.2.5.3 Listing of XML File Written by WriteMicroTitrePlateDataBaseSchema()

The following is a listing of the contents of the XML file that is written by the WriteMicroTitrePlateDataBaseSchema() method and that can be read in to create an application DataTable for the MicroTitrePlate Control. Additional Columns can be added to the DataTable; however, they will be ignored by the control.

```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:MainDataTable="MicroTitrePlate"
                      msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="MicroTitrePlate">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="WellNumber" msdata:Caption="Well&#xD;&#xA; Number"
                          type="xs:int" minOccurs="0" />
              <xs:element name="WellLocation" msdata:Caption="Well&#xD;&#xA;Location"
                          type="xs:string" minOccurs="0" />
              <xs:element name="N" type="xs:int" minOccurs="0" />
              <xs:element name="Row" type="xs:int" minOccurs="0" />
              <xs:element name="Column" type="xs:int" minOccurs="0" />
              <xs:element name="Type" msdata:Caption="Well&#xD;&#xA;Type"
                          type="xs:string" minOccurs="0" />
              <xs:element name="ID" msdata:Caption="Sample&#xD;&#xA;ID"
                          type="xs:string" minOccurs="0" />
              <xs:element name="Description" type="xs:string" minOccurs="0" />
              <xs:element name="XValue" msdata:Caption="Number of&#xD;&#xA;Pixels"
                          type="xs:double" minOccurs="0" />
              <xs:element name="YValue" msdata:Caption="Corrected&#xD;&#xA;Intensity"
                          type="xs:double" minOccurs="0" />
              <xs:element name="WellAnnotation" msdata:Caption=
                          "Well Annotation&#xD;&#xA;[Min,Avg,σ,Max]"
                          type="xs:string" minOccurs="0" />
              <xs:element name="Exclude" type="xs:boolean" minOccurs="0" />
              <xs:element name="Empty" type="xs:boolean" minOccurs="0" />
              <xs:element name="Replicate" type="xs:int" minOccurs="0" />
              <xs:element name="Group" type="xs:int" minOccurs="0" />
```



```
            <xs:element name="RowStatus" type="xs:int" minOccurs="0" />
            <xs:element name="WellRectX" msdata:Caption="Well Rect&#xD;&#xA;X"
                        type="xs:int" minOccurs="0" />
            <xs:element name="WellRectY" msdata:Caption="Well Rect&#xD;&#xA;Y"
                        type="xs:int" minOccurs="0" />
            <xs:element name="WellRectWidth" msdata:Caption="Well Rect&#xD;&#xA;Width"
                        type="xs:int" minOccurs="0" />
            <xs:element name="WellRectHeight" msdata:Caption="Well Rect&#xD;&#xA;Height"
                        type="xs:int" minOccurs="0" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

4.2.6 Set and Get Well DataTable Values

There are set methods for each of the data fields in the minimum DataTable. The methods take either a Plate Row-Column index pair or a Table-Row index for determining the row in the DataTable where the set values will be applied. In addition, the methods assume that the values will be assigned to the currently selected DataTable which can be set by using:

```
mtp.SelectedPlate = MicroTitrePlate.PlateSelection.Plate
```

An application can use the Set methods for dynamically updating the plate view. For example, first create a DataTable containing a row for each well where the Table-Row, Plate-Row and Plate-Column fields are defined, the WellType is "Empty" and all other fields are null. Then, as well data is defined (e.g. wells are filled), these methods can be used to set the values there by updating the plate's DataTable.

Note: mtp.Refresh() should be called after all of the SetWell methods have been called or whenever the GUI is to reflect the changes. This approach was chosen to minimize refresh flickering if the GUI was to change after each set method was called. Also, mtp.SuspendRefresh = true; can be used to minimize flicker when many property values are being set.

Method	Description
RemoveWellColors() or ClearWellColors()	Either of these methods removes all individually defined well colors and re-renders all wells using the globally defined color scheme.
GetWellGroupNumber(…)	This method returns the group number for the specified row in the DataTable or well in the plate
GetWellReplicateNumber(…)	This method returns the replicate number for the specified row in the DataTable or well in the plate
SetDisplayNumberFormat(…)	This method sets the format to use when displaying/rendering numbers contained in the DataTable.
SetWell(…)	This method can be used to set a well's data values. Internally it simply calls the individual set method

	for each argument.
SetWellAnnotation(…)	This method sets the annotation that appears inside the well.
SetSelectedWellColor()	This method will define the well fill color for the current set of selected wells including any replicate or group wells. If property UseIndividualWellColors is set to true then defined colors will be used when rendering the wells; otherwise, the globally defined color scheme will be used.
SetWellColor()	This method is used to define the Fill Color for an individual well and overloaded methods are used to define the fill color for a range of wells. If property UseIndividualWellColors is set to true then defined colors will be used when rendering the wells; otherwise, the globally defined color scheme will be used.
SetWellGroupNumber(…)	This method sets the group number for the specified well.
SetWellDescription(…)	This method sets the description for the specified well.
SetWellReplicateNumber(…)	This method sets the replicate or group number for the specified well.
SetWellID(…)	This method sets the ID for the specified well.
SetWellType(…)	This method sets the type of Well using the string for one of the WellTypes enum values. Use GetWellTypeEnumName(WellTypes.???) to obtain the string.
SetWellXYValues(…)	This method sets the X and Y values for the specified well.
SetSelectedWellsAsReplicates()	This method will assign the current set of selected wells either the next available replicate group number or a replicate number that one of the wells already has. If the selected wells already contain conflicting replicate numbers no replicate number is assigned and an Error event is triggered. An overloaded method permits specifying the replicate number.
SetSelectedWellsAsNotReplicates()	This method will set the replicate number of the selected wells to zero. That is, it will unassign the current set of selected wells as replicates.
SetSelectedWellsAsGroup()	This method will assign the current set of selected wells either the next available group number or a group number that one of the wells already has. If the selected wells already contain conflicting group numbers no group number is assigned and an Error

	event is triggered. An overloaded method permits specifying the group number.
SetSelectedWellsAsNotGroup()	This method will set the group number of the selected wells to zero. That is, it will unassign the current set of selected wells to a group.

Table 7: Description of Set Well DataTable value methods.

4.2.7 Rotate Plate and Row-Column Labels

The row-column size of a plate or rectangular array of wells or positions for placing test tubes such as in a fraction collector is specified through the SetDataTable….() methods where the number of rows and columns are arguments for the methods. The array is, by default, rendered with the rows being labeled from top to bottom on the left side of the array and the columns from left to right above the array as shown in Figure 10.

If it is necessary to rotate the array, then the property RotatePlate is set to true. This causes the array to be rotated 90 degrees counterclockwise and the rows now represent the columns and are labeled from bottom to top. The corner cuts will be automatically rotated if the properties dialog is used, but if properties are set programmatically, then the corner cut properties must also be set programmatically. The columns now represent the rows and they are labeled left to right at the top of the array. If it is necessary to label the columns at the bottom or below the array, then set ShowColLabelsBottom to true.
shows a rotated plate with column labels below the plate

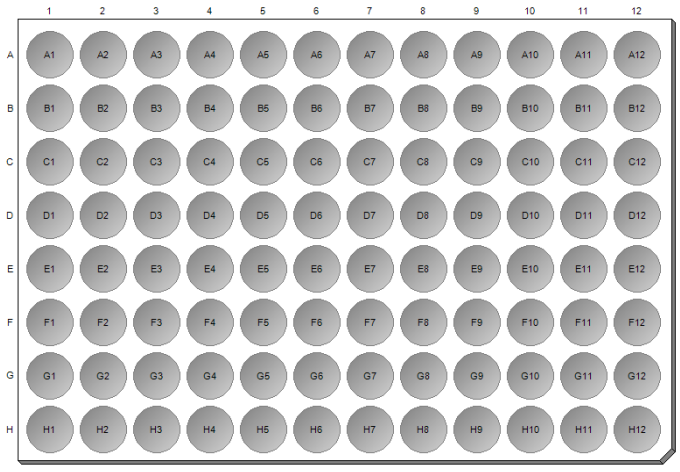


Figure 10: Default Orientation of Plate

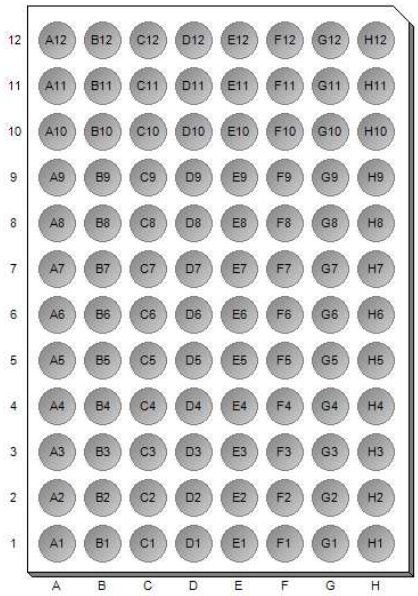


Figure 11: Rotated Plate with Column

Labels below Plate

4.2.8 IsWellxxx State Methods

There are a few methods that can be used to determine the state of a well. There are two overloaded forms of each method, one accepts plate Row and Column indexes and the other accepts a Table Row index.

Method	Returns
IsWellEmpty	true if well is empty; otherwise false
IsWellExcluded	true if well is excluded; otherwise false
IsWellHidden	true if well is hidden; otherwise false
IsWellSelected	true if well is selected; otherwise false

Table 8: List of IsWellxxx State Methods

4.2.9 Set State Methods

There are a few methods that can be used to set the state of a well. There are two overloaded forms of each method, one accepts plate Row and Column indexes and the other accepts a Table Row index.

Method	Description
EmptyWell	Marks the specified well as empty. This does not change the WellType, it is intended to reflect that the well is not filled. That is, it could be of type 'Sample' without any sample in the well. The WellType of empty is meant to

	reflect that the well is not being used.
ExcludeWell	Exclude the specified well – the intent is that maybe the well content was not valid as determined by a measurement
HideWell	Hide the specified well. Used to create different visual GUI.
SelectWell	Select the specified well or a range of wells. When a single well is selected then related wells are also selected such as its replicates if the ShowReplicate option is set to true and/or members of its group if the ShowGroup option is set to true.
UnSelectWell	This method is used to unselect a well indexed by its DataTable row value. The unselected well is removed from the SelectedWells ArrayList and the selected Well Rectangle is erased. This method uses the default Graphics object created from the control's CreateGraphics() method.
UnSelectWells	This method clears all selected wells and any wells that were colored including replicates and groups.

Table 9: List of Set Well State Methods

4.2.10 Well Blinking Methods

There are methods to start and stop blinking of selected wells and if a single well is selected then replicates and/or group wells. Blinking is displayed by alternately drawing the dashed-line rectangles and if UseIndividualWellColors is set to true, the fill color of the wells is also alternately displayed. The following figure shows the different types of rectangles and color wells that are blinked.

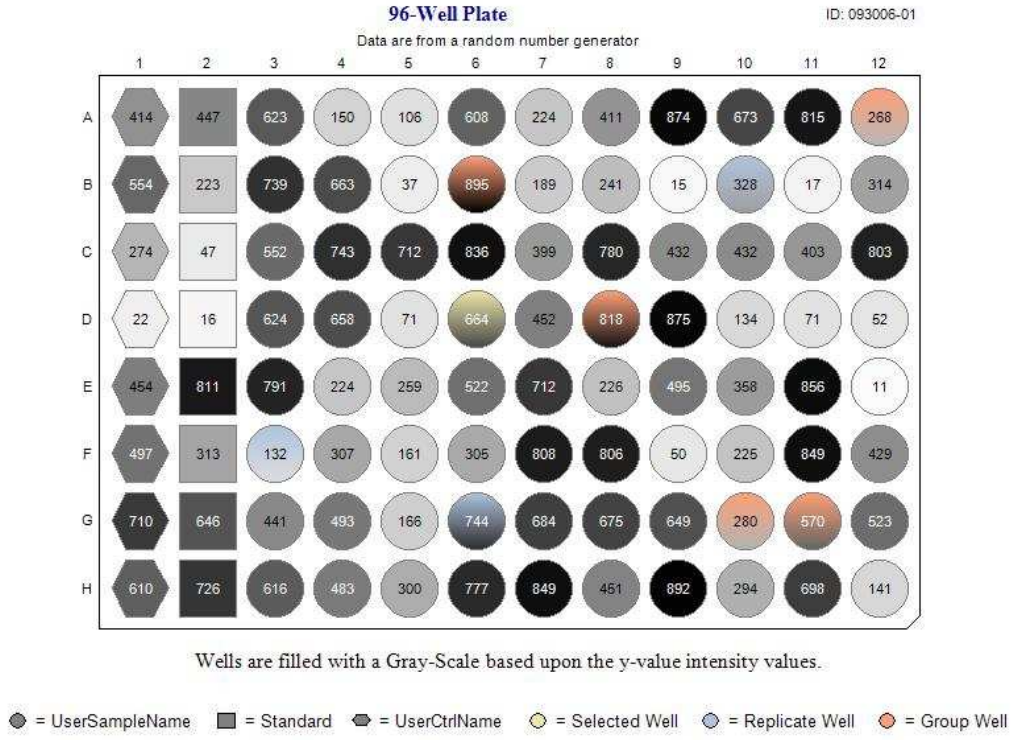


Figure 12: View showing different types of rectangles and colored wells.

In addition, there are two events that can be used for notification of starting and stopping of blinking. Refer to the section on Events.

There are two different scenarios for implementing blinking:

1. Blinking can be managed by the calling program by calling the StartBlinking() and StopBlinking() methods coupled with using a Timer object such as one created using Windows.Forms.Timer. The duration and blink interval are set by the calling program through the local Timer object.

2. Blinking can be managed by the MicroTitrePlate Control by calling the blinking methods which permit setting of the duration and interval. Internally the Control will start a Timer object and define the duration and interval using these values. Blinking will automatically stop when the duration is reached.

   NOTE: IF THE DURATION IS SET TO -1 THEN THE CALLING PROGRAM MUST CALL StopBlinking() TO END BLINKING.

Method	Description
StartBlinking()	This method is used to start blinking all selected wells including replicate and group wells if ShowReplicates and ShowGroups are set to true. Blinking is indicating by drawing and hiding their dashed line rectangles. In addition, if UseIndividualWellColors is set to true then the well fill color will also blink. The method StopBlinking() must be called to stop the well blinking. Use other methods to specify an interval and a duration of blinking. Default interval is 1000 milliseconds.
StartBlinkingWell(int Row, int Column)	This method is used to start blinking the selected well dashed line rectangle In addition, if UseIndividualWellColors is set to true then the well fill color will also blink.The method StopBlinking() must be called to stop the well blinking. Use other methods to specify an blinking Interval and a Duration of blinking. Default interval is 1000 milliseconds.
StartBlinking(int Interval, int Duration)	This method is used to start blinking all selected wells including replicate and group wells if ShowReplicates and ShowGroups are set to true. Blinking is indicating by drawing and hiding their dashed line rectangles. In addition, if UseIndividualWellColors is set to true then the well fill color will also blink. Call method StopBlinking() and set Duration to -1 to manually stop well blinking; otherwise, set Duration to the number of milliseconds that blinking is to occur for automatic stopping of blinking.
StartBlinkingWell(int Row, int Column, int Interval, int Duration)	This method is used to start blinking the selected well and dashed line rectangle In addition, if UseIndividualWellColors is set to true then the well fill color will also blink. If ShowReplicates and/or ShowGroup is set to true then their rectangles and well fill colors will also blink. Call method StopBlinking() and set Duration to -1 to manually stop well blinking; otherwise, set Duration to the number of milliseconds that

	blinking is to occur for automatic stopping of blinking.
StopBlinking()	This method manually stops well blinking.

Table 10: List of well blinking methods

4.2.11 OwnerDraw Option

The ***DrawWellMode*** option provides the developer with the capability to render content at well sites. That is, just before a well site is to be rendered by the BxMicroTitrePlate control it checks the ***DrawWellMode*** property to see if it is set to ***OwnerDraw***. If it is set to ***OwnerDraw*** and the ***DrawWellItem*** event is not null then the ***DrawWellItem*** event is triggered, its ***DrawWellItemEventArgs*** structure is populated and rendering at the well site is then carried out by the ***DrawWellItem*** event handler. In conjunction with this, there is another property that is available to pass back rendering of the well site to the control. The name of this property is ***DrawWellNormal*** which is set to **`false`** just prior to triggering the ***DrawWellItem*** event handler and if the handler wants to defer drawing to the control, then the handler should set the property to **`true`** and return. This is useful when it is necessary to override the control's rendering for specific wells and at all other times the control's rendering is acceptable.

The intent of this option was to enable applications that manage their own images and/or graphs to render them at well sites without passing the images or graph data to the BxMicroTitrePlate control. This ensures that there is only one instance of an image or graph data which minimizes memory requirements. In addition this feature enables the application to render a unique design for the well.

All other functionality such as displaying of tooltips and mouse interactive selection of wells is still preserved.

4.2.11.1 DrawWellItemEventArgs

The DrawWellItemEventArgs structure is passed as one of the arguments in the DrawWellItem event handler.

Column	Data Type	Description
Bounds	`Rectangle`	Rectangle including location of well to be drawn.
Column	`int`	Column index value for well
font	`Font`	Font for rendering well text.
graphics	`Graphics`	Graphics object used for drawing the well
Row	`int`	Row index value for well
State	`WellState`	WellState enum value of well to be drawn (Unselected, Selected, Replicate, Group, Empty, Hidden or Excluded)
TableRow	`Int`	Index of well within the plate DataTable.

WellType	`WellType`	The string can contain the internal name of a Well Type obtained from a WellTypes enum value or one defined programmatically through the SetWellTypeName(). This string is used to determine the shape of the well and can be used for programmatic functions. If string is null or empty then the Well Type Shape is automatically defined as the shape of a sample.
		
		The control provides a method that can be used to get the name of a WellTypes enum value. For example, if mtp is a MicroTitrePlate control and the string name for a Sample well type is needed, then the following code can be used.
		
		`MicroTitrePlate.WellTypes wt = MicroTitrePlate.WellTypes.Sample;`
		
		`mtp.GetWellTypeEnumName(wt);`
		
		The GetWellTypeEnumName() method returns the user defined name for the well type when property UseUserWellTypeNames is set to true or predefined internal names if false.

Table 11: DrawWellItemArgs structure definitions.

4.2.11.2 WellState enum

This enum is used in the DrawWellItemEventArgs returned in the DrawWellItem event handler. Where it makes sense the values can be bit-wise OR'd together. For example, an unselected well that is marked as empty is WellState.Unselected | WellState.Empty.

The following code snippet can be used to check whether the state of the well is empty or not.

```
WellState ws = GetWellState(row, column);

if((ws & WellState.Empty) != 0)
  // Well is marked as empty
else
  // Well is not marked as empty.
```

Member	Description
Unselected	The well is unselected.
Selected	The well is selected.

Replicate	The well is a replicate of the selected well and the property ShowReplicates is set to true.
Group	The well belongs to the same group as the selected well and the property ShowGroup is set to true.
Empty	The well is marked as empty using the boolean Empty plate DataTable field or it has a well type of Empty
Excluded	The well is excluded.
Hidden	The well is hidden.

4.2.11.3 OwnerDraw Example Code Snippets

Common code for examples:

```
// Set Well Draw mode to Owner Draw
mtp.DrawWellMode = MicroTitrePlate.DrawMode.OwnerDraw;

// Subscribe to the DrawWellItem event
mtp.DrawWellItem += new DrawWellItemEventHandler(mtp_DrawWellItem);
```

Example 1 – This example renders all wells in the owner draw event handler. Wells are render with different fill colors for the well states, a black border and an interior frame.

NOTE: Even though the State member can have bits on that designate Empty, Excluded or Hidden, they are not checked for in this code example. Refer to the next code snippet for an example of checking those bits.

```
// Event handler for the DrawWellItem event.
void mtp_DrawWellItem(object sender, MicroTitrePlate.DrawWellItemEventArgs e)
{
  // fill the well site rectangle with a color based upon its state
    if ((e.State & MicroTitrePlate.WellState.Selected) != 0)
        e.graphics.FillRectangle(new SolidBrush(mtp.SelectedWellFillColor), e.Bounds);
    else
        if ((e.State & MicroTitrePlate.WellState.Replicate) != 0)
            e.graphics.FillRectangle(new SolidBrush(mtp.ReplicateWellFillColor),
               e.Bounds);
        else
            if ((e.State & MicroTitrePlate.WellState.Group) != 0)
                e.graphics.FillRectangle(new SolidBrush(mtp.GroupWellFillColor),
               e.Bounds);
            else
            {
                e.graphics.FillRectangle(new SolidBrush(mtp.PlateWellColor), e.Bounds);
            }
       // Draw a black border
        e.graphics.DrawRectangle(Pens.Black, e.Bounds);

       // Draw the interior frame of the well site.
        for (int i = 1; i <= 4; i++)
        {
            e.graphics.DrawLine(Pens.DimGray, new Point(e.Bounds.X + i, e.Bounds.Y + i),
                new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i));
            e.graphics.DrawLine(Pens.Gray, new Point(e.Bounds.X + i, e.Bounds.Y + i),
                new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i));
            e.graphics.DrawLine(Pens.DarkGray,
```



```
                new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i),
                new Point(e.Bounds.X + e.Bounds.Width - i,
                    e.Bounds.Y + e.Bounds.Height - i));
            e.graphics.DrawLine(Pens.LightGray,
                    new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i),
                    new Point(e.Bounds.X + e.Bounds.Width - i,
                        e.Bounds.Y + e.Bounds.Height - i));
        }

}
```

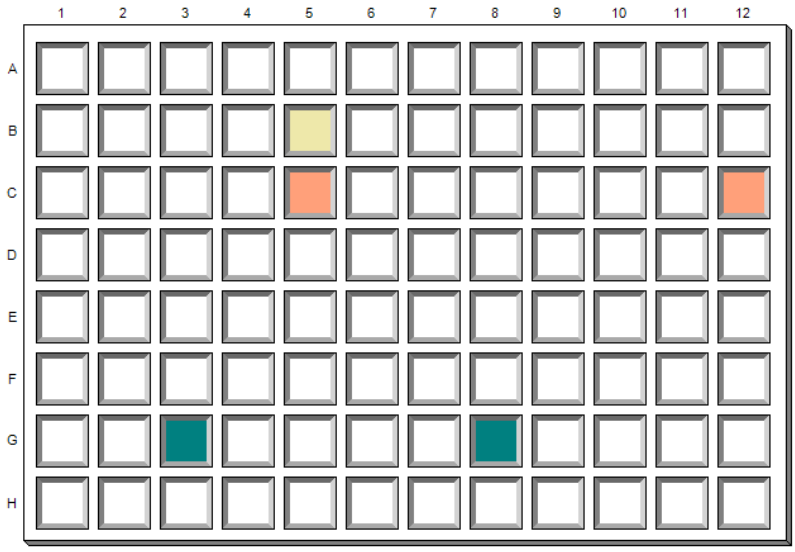


Figure 13: All wells being rendered using DrawWellItemEvent handler.

Example 2 – This example renders only Selected, Replicate and Group wells in the owner draw event handler and all other wells are rendered using the control. Owner Draw Wells are rendered with different fill colors for the well states, with a letter designating the well state, a black border and an interior frame which are easily seen as being different from the rendering of the unselected wells by the control.

NOTE: The State member can contain a bit designating that the well is also Empty, Excluded or Hidden, so a bitwise check is used.

The DrawWellNormal property is set to true to force the control to render unselected wells.

```
void mtp_DrawWellItem(object sender, MicroTitrePlate.DrawWellItemEventArgs e)
{
    // Fill bounds with color representing state of the well.
    StringFormat sfmt = new StringFormat();
    sfmt.Alignment = StringAlignment.Center;
    sfmt.LineAlignment = StringAlignment.Center;
```



```
if ((e.State & MicroTitrePlate.WellState.Hidden) == 0)
{
    if ((e.State & MicroTitrePlate.WellState.Selected) != 0)
    {
        if ((e.State & MicroTitrePlate.WellState.Empty) != 0)
            e.graphics.FillRectangle(new HatchBrush(HatchStyle.LightUpwardDiagonal,
                 mtp.WellEmptyColor, Color.White), e.Bounds);
        else
            e.graphics.FillRectangle(new SolidBrush(mtp.SelectedWellFillColor),
                 e.Bounds);

        if ((e.State & MicroTitrePlate.WellState.Excluded) != 0)
        {
            e.graphics.DrawLine(Pens.Red, new Point(e.Bounds.X, e.Bounds.Y),
                new Point(e.Bounds.X + e.Bounds.Width,
                     e.Bounds.Y + e.Bounds.Height));
            e.graphics.DrawLine(Pens.Red, new Point(e.Bounds.X+e.Bounds.Width,
                e.Bounds.Y), new Point(e.Bounds.X, e.Bounds.Y + e.Bounds.Height));
        }
        e.graphics.DrawString("S", e.font, Brushes.Black, e.Bounds, sfmt);
    }
    else
        if ((e.State & MicroTitrePlate.WellState.Replicate) != 0)
        {
            if ((e.State & MicroTitrePlate.WellState.Empty) != 0)
                e.graphics.FillRectangle(
                  new HatchBrush(HatchStyle.LightUpwardDiagonal,
                       mtp.WellEmptyColor, Color.White), e.Bounds);
            else
            e.graphics.FillRectangle(
                  new SolidBrush(mtp.ReplicateWellFillColor), e.Bounds);
            if ((e.State & MicroTitrePlate.WellState.Excluded) != 0)
            {
                e.graphics.DrawLine(Pens.Red, new Point(e.Bounds.X, e.Bounds.Y),
                    new Point(e.Bounds.X + e.Bounds.Width,
                        e.Bounds.Y + e.Bounds.Height));
                e.graphics.DrawLine(Pens.Red,
                    new Point(e.Bounds.X+e.Bounds.Width, e.Bounds.Y),
                    new Point(e.Bounds.X, e.Bounds.Y + e.Bounds.Height));
            }
            e.graphics.DrawString("R", e.font, Brushes.Black, e.Bounds, sfmt);
        }
        else
            if ((e.State & MicroTitrePlate.WellState.Group) != 0)
            {
            if ((e.State & MicroTitrePlate.WellState.Empty) != 0)
                e.graphics.FillRectangle(
                   new HatchBrush(HatchStyle.LightUpwardDiagonal,
                       mtp.WellEmptyColor, Color.White), e.Bounds);
            else
                e.graphics.FillRectangle(
                    new SolidBrush(mtp.GroupWellFillColor), e.Bounds);
                if ((e.State & MicroTitrePlate.WellState.Excluded) != 0)
                {
                    e.graphics.DrawLine(Pens.Red, new Point(e.Bounds.X, e.Bounds.Y),
                          new Point(e.Bounds.X + e.Bounds.Width,
                               e.Bounds.Y + e.Bounds.Height));
                    e.graphics.DrawLine(Pens.Red,
                          new Point(e.Bounds.X+e.Bounds.Width, e.Bounds.Y),
                          new Point(e.Bounds.X, e.Bounds.Y + e.Bounds.Height));
                }
                e.graphics.DrawString("G", e.font, Brushes.Black, e.Bounds, sfmt);
            }
            else
            {
                // Control should render unselected wells.
                mtp.DrawWellNormal = true;
            }

    // Draw border and interior frame if control is not rendering well
    if (!mtp.DrawWellNormal)
```



```
        {
            // draw black border
            e.graphics.DrawRectangle(Pens.Black, e.Bounds);

            // draw interior frame within bounds
            for (int i = 1; i <= 4; i++)
            {

                if (e.State == MicroTitrePlate.WellState.Unselected)
                {
                    e.graphics.DrawLine(Pens.LightGray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i));
                    e.graphics.DrawLine(Pens.DarkGray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i));
                    e.graphics.DrawLine(Pens.Gray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i),
                        new Point(e.Bounds.X + e.Bounds.Width - i,
                                  e.Bounds.Y + e.Bounds.Height - i));
                    e.graphics.DrawLine(Pens.DimGray,
                        new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + e.Bounds.Width - i,
                                  e.Bounds.Y + e.Bounds.Height - i));

                }
                else
                {
                    e.graphics.DrawLine(Pens.DimGray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i));
                    e.graphics.DrawLine(Pens.Gray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i));
                    e.graphics.DrawLine(Pens.DarkGray,
                        new Point(e.Bounds.X + i, e.Bounds.Y + e.Bounds.Height - i),
                        new Point(e.Bounds.X + e.Bounds.Width - i,
                                  e.Bounds.Y + e.Bounds.Height - i));
                    e.graphics.DrawLine(Pens.LightGray,
                        new Point(e.Bounds.X + e.Bounds.Width - i, e.Bounds.Y + i),
                        new Point(e.Bounds.X + e.Bounds.Width - i,
                                  e.Bounds.Y + e.Bounds.Height - i));
                }
            }
        }
    }

}
```

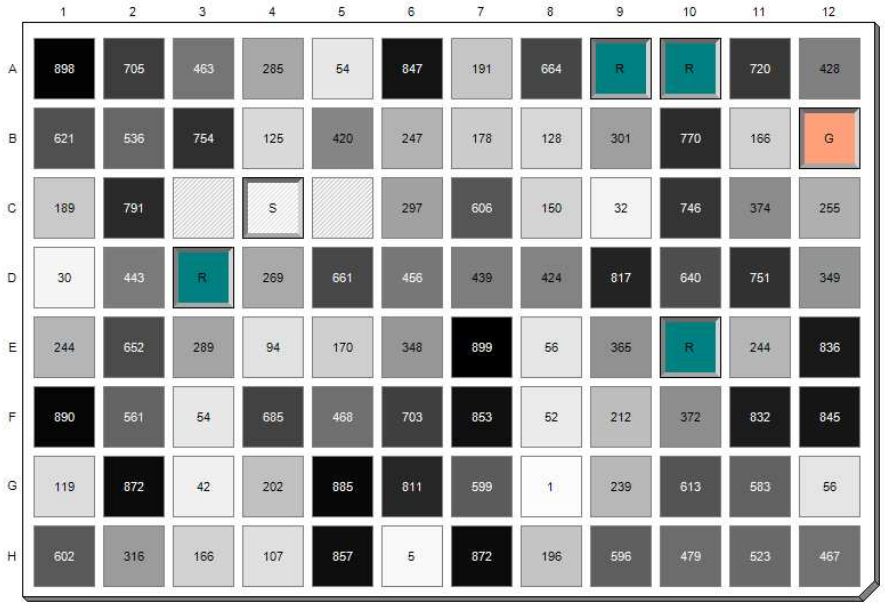


Figure 14: Selected(S), Replicate (R) and Group(G) wells are rendered in DrawWellItemEvent handler.

4.2.12 Provided Menu Item and Button Images

The MicroTitrePlate control has an assortment of embedded images that can be retrieved for use in other objects or controls such as Menu Items and Buttons. The transparent color is Silver for all images which can also be obtained from the ImageTransparentColor property as:

```
Color transparentColor = mtp.ImageTransparentColor;
```

Image	Name	Transparent Color	Get Property
	Copy Plate	Silver	ImageCopyPlate
	Change Well Type	Silver	ImageChangeWellType
	Exclude Well	Silver	ImageExcludeWell
	Include Well	Silver	ImageIncludeWell
	Plate	Silver	ImagePlate
	Mark as Empty Well	Silver	ImageEmptyWell
	Mark Selected Wells as Replicates	Silver	ImageReplicateWell
	Unmark	Silver	ImageUnreplicateWell

	Selected Wells as Replicates		
	Group Wells	Silver	ImageGroupWell
	Ungroup Wells	Silver	ImageUngroupWell

Table 12: List of Images for well and plate actions

Image image = mtp.ImageWellShape(WellShapesStyles.Circle);

The SquareFrame shape is designed when representing fraction collectors with square locations for test tubes.

Image	Name	Transparent Color	WellShapesStyles Enum Value
	Circle	Silver	Circle
	Bar Graph	Silver	Bar_Graph
	Diamond	Silver	Diamond
	GaussianGraph	Silver	Gaussian_Graph
	Hexagon	Silver	Hexagon
	Probe	Silver	Probe
	Square	Silver	Square
	SquareFrame	Silver	SquareFrame
	Shading Under Curve	Silver	ShadingUnderCurve
	Target	Silver	Target
	Triangle	Silver	Triangle

Table 13: List of Images for Well Shapes

Figure 15 illustrates the use of images in menu items. The menu on the left uses the following code to obtain the image for ChangeWellType and will later be assigned to its corresponding menu item:

Image imageMenuItem = mtp.ImageChangeWellType;

The menu on the right uses the following code` to get the image associated with a well that has content of type Control and will later be assigned to its corresponding menu item:

Image imageMenuItem = mtp.ImageShapeControl;

For example, using a ToolStripMenuItem object the code may look like the following:


```
private System.Windows.Forms.ToolStripMenuItem miCopyToClipboard;
miCopyToClipboard = new System.Windows.Forms.ToolStripMenuItem();

miCopyToClipboard.Text = "Copy To Clipboard";
miCopyToClipboard.Image = mtp.ImageCopyPlate;
miCopyToClipboard.ImageTransparentColor = mtp.ImageTransparentColor;
```

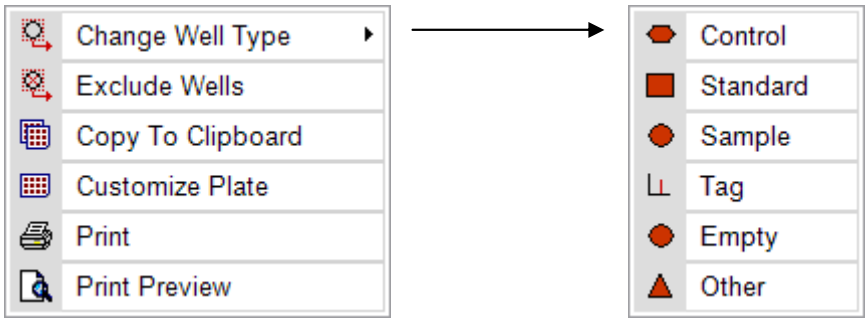


Figure 15: Example of Context Menu Items using BxMicroTitrePlate Images

4.3 Well Selection

Built into the control is the capability to select a single well using the Left Mouse Button and selecting multiple wells using the Left Mouse Button while dragging the mouse or in combination with Ctrl, Shift or Alt keys. Selected wells are indicated by a dashed-line rectangle or a dashed-line rounded corner rectangle where the color of the different style dashed lines can be specified. This section describes the different selection results. The property *SelectedWells* returns an ArrayList of Well structures, one for each of the selected Wells and the properties *ReplicateWells* and *GroupedWells* returns an ArrayList of Well structures, one for each of the replicate or grouped Wells respectively.

4.3.1 Left Mouse Button

4.3.1.1 Using a Rubber Band Rectangle to Select Sets of Wells

Dragging the mouse while the Left Mouse Button is down will draw a rubber band rectangle and when the Left Mouse Button is released to the up position, all wells within the final rectangle including those on the edges of the rectangle will be selected.

More sets of wells can be selected by holding down the Ctrl-Key while using the above rubber band procedure.

4.3.1.2 Single Click Left Mouse Button

A Left Mouse Button click is used to select a single well that is highlighted with a dashed-line rectangle and if the *Replicates* option is set to true or Checked in the Properties Configuration Dialog Box Form then all replicates for the selected well are also selected and highlighted with a dashed-line-rounded corner rectangle.

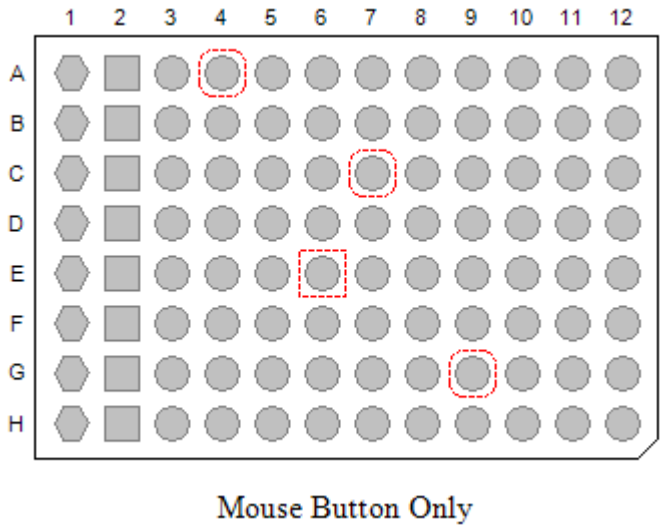


Figure 16: Left Mouse Only - Showing Selected Wells and Replicates

4.3.2 Left Mouse Button + Ctrl Key

The image in Figure 17 was obtained by holding down the Ctrl Key and clicking the Left Mouse button over different wells where each selected well is highlighted with a red dashed-line rectangle.

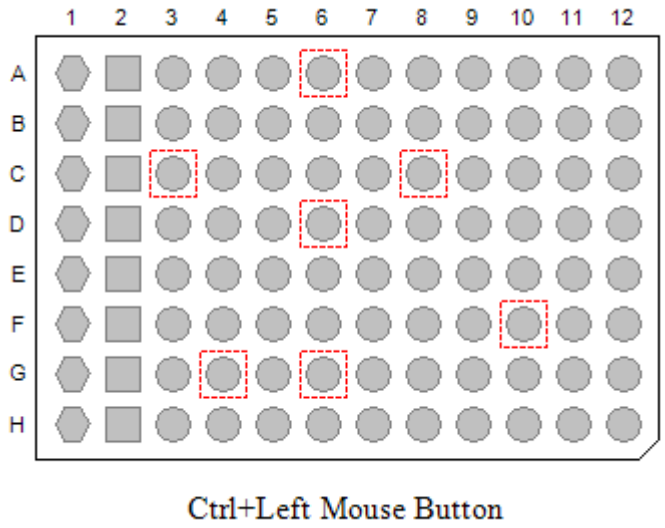


Figure 17: Selected Wells using Left Mouse Button + Ctrl Key

NOTE: Holding down the Ctrl-Key while performing

4.3.3 Left Mouse Button + Shift Key

The image in Figure 18 was obtained by

- Setting the *Replicates* option to false or Unchecked in the Properties Configuration Dialog Box Form.
- Clicking the left Mouse button on D6
- Holding down the Shift Key and clicking the Left Mouse button on E8. An array of wells in the range [D6 : E8] is selected and each well is highlighted with a red dashed-line rectangle.

If range is in the same Row or same column then all wells in the row range or column range will be selected

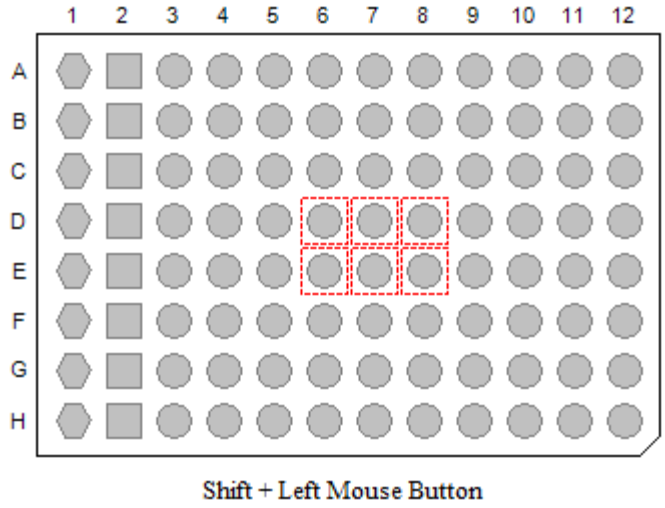


Figure 18: Rectangular array of wells is selected

4.3.4 Left Mouse Button + Alt Key

The image in Figure 19 was obtained by

- Setting the *Replicates* option to false or unchecked in the Properties Configuration Dialog Box Form.
- Clicking the left Mouse button on D6
- Holding down the Alt Key and clicking the Left Mouse button on E8. All of wells in the range [D6 : E8] is selected and each well is highlighted with a red dashed-line rectangle.

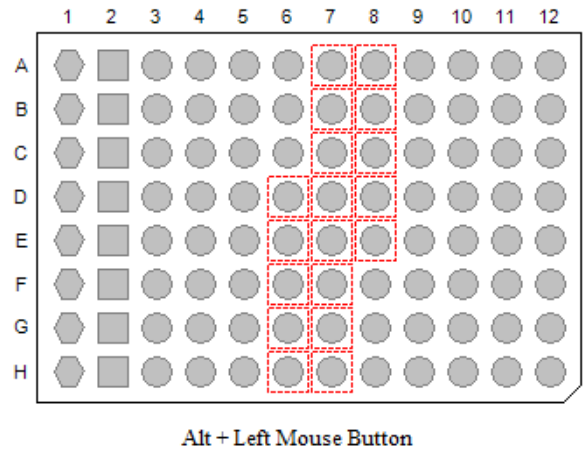


Figure 19: Wells selected using Alt Key + Left Mouse Button

4.4 Properties

This section will describe all of the public properties available through the API and most can be set using the Properties Dialog box that is described in section 6. The first sections contain descriptions of the properties divided into categories and a complete alphabetical listing of all properties with their default values is in section 4.4.11.

4.4.1 Colors

This section contains the properties for setting colors related to the appearance of the plate and its container. The Fonts section provides additional properties for setting the ForeColor and BackColor for Fonts.

Property	DataType	Description
CaptionBorderColor	Color	Get or Set the border color for the microtitre plate Caption rectangle
ExcludeColor	Color	Set or Get Color used for drawing the exclude mark (X) onto wells.
FocusRectangleColor	Color	Get or Set the Focus Rectangle Color of the container containing the microtitre plate, titles and captions
GroupWellFillColor	Color	Set or Get the color to use to fill group wells when the UseIndividualWellColors is set to true. Default color is LightSalmon.
GroupWellLineColor	Color	Set or Get the color for the border of the well group rectangle.
OutLineShapeColor	Color	Set or Get Color used for outlining the shape of the well icon when the property OutLineShape is true.
PlateBackgroundImage	Image	Set the image to use for the background of the plate. If ShowPlateBackgroundImage is also set to true then the image will be used in stead of the PlateBackcolor.
PlateBorderColor	Color	Get or Set the border color of the microtitre plate
PlateColor	Color	Get or Set the color for the microtitre plate
PlateContainerBackColor	Color	Get or Set the background color for the container containing the microtitre

		plate, titles and captions
PlateContainerBorderColor	Color	Get or Set the border color of the container containing the microtitre plate, titles and captions
PlateWellColor	Color	Set or Get the color that is used for the color of the plate wells.
PlateWellPrimaryColor	Color	Set or Get the primary color for Plate wells that is used when the Gray_Scale is selected for the well fill style.
ReplicateWellFillColor	Color	Set or Get the color to use to fill replicate wells when the UseIndividualWellColors is set to true. Default color is LightSteelBlue.
ReplicateWellLineColor	Color	Set or Get the color for the border of the well replicate rectangle.
SelectedWellFillColor	Color	Set or Get the color to use to fill selected wells when the UseIndividualWellColors is set to true. Default color is PaleGoldenrod.
SelectedWellLineColor	Color	Set or Get the color for the border of the well selection rectangle.
TitleBorderColor	Color	Get or Set the color for the microtitre plate Title and Subtitle border color.
WellEmptyColor	Color	Set or Get the color to use for an empty well.
WellFillStyle	WellFillStyles	Set or Get the fill style for wells (e.g. solid, hatch, gradient, gray_scale) Note: When Gray_Scale is used and the Color.Gray is selected and the Annotation Font text color is Black, then the annotation color will be adjusted to White when the well color is determined to be too dark to show text with a black color.

Table 14: Description of Properties for Colors.

4.4.2 Configuration

Property	DataType	Description
HighlightSelectedWellsWithColors	bool	Set or Get boolean value which enables highlighting (true) or not highlighting (false) of selected, replicate and group wells with

		colors.
HighlightSelectedWellsWithDashLineRectangles	bool	Set or Get boolean value which enables highlighting (true) or not highlighting (false) of selected, replicate and group wells with dashed-lined rectangles.
IncludeReplicateAndGroupWellsInLegend	bool	Set or Get boolean value for whether to show in the plate legend Replicate and Group Well color keys.
LabelRowsWithNumbers	bool	Get or Set boolean for whether the microtitre plate will label rows with numbers instead of letters. This is used to override the standard notation used certain for microtitre plate sizes.
OutLineShape	bool	Set or Get boolean value for whether the shape of the well icon is to be outlined (true) or not(false).
PlateAlignment	ContentAlignment	Set or Get the ContentAlignment value for aligning the plate within the drawing area.
RotatePlate	bool	Set to true to rotate the plate CounterClockwise by 90 degrees. The row and column labeling will be rendered accordingly.
ShowCaption	bool	Get or Set boolean for whether the microtitre plate Caption is to be shown (true) or not (false).
ShowCaptionBorder	bool	Get or Set boolean for whether the microtitre plate Caption is to have a frame (true) or not (false).
ShowColLabelsBottom	bool	Get or Set boolean for whether column labels will be rendered at the bottom of the plate when ShowRowColLabels is set to true.
ShowCornerCutBottomLeft	bool	Get or Set boolean for whether the microtitre plate will have a bottom left corner cut (true) or not (false).
ShowCornerCutBottomRight	bool	Get or Set boolean for whether the microtitre plate will have a bottom right corner cut (true) or

		not (false).
ShowCornerCutTopLeft	bool	Get or Set boolean for whether the microtitre plate will have a top left corner cut (true) or not (false).
ShowCornerCutTopRight	bool	Get or Set boolean for whether the microtitre plate will have a top right corner cut (true) or not (false).
ShowFocusRectangle	bool	Get or Set boolean for whether the microtitre plate will show a focus rectangle (true) or not (false).
ShowGroup	bool	Set or Get boolean value for whether to show wells in the same Group (true) as the selected well or not (false).
ShowLegend	bool	Set or Get boolean value for whether to show plate legend (true) or not (false). Default is false.
ShowLegendBorder	bool	Set or Get boolean value for whether to show plate legend Border(true) or not (false). Default is false.
ShowLegendVertical	bool	Set or Get boolean value for whether to show plate legend Vertical(true) or not (false). Default is false.
ShowMagnifierWindow	bool	Set or Get boolean that Shows(true) or Closes(false) the Magnifier Window.
ShowPlateBackgroundImage	Bool	Set or Get flag that will use the PlateBackgroundImage as the background for the plate if true otherwise the PlateBackColor will be used.
ShowPlateBorder	bool	Get or Set boolean for whether the microtitre plate will show a border(true) or not (false).
ShowPlateContainerBorder	bool	Get or Set boolean for whether the microtitre plate will show a frame rectangle (true) or not (false).
ShowPlateID	bool	Set or Get boolean for showing(true) or not

		showing(false) a plate ID, if defined, in the Title Bar on the far right
ShowPlate3D	bool	Get or Set boolean for whether the microtitre plate will show a border in 3D(true) or not (false). The bottom and right borders will have a thickness filled in with Color.Gainsboro and a border line color the same as the plate border.
ShowRowColLabels	bool	Get or Set boolean for whether the microtitre plate will have row and column labels (true) or not (false).
ShowReplicates	bool	Set or Get boolean value for whether to show Replicates (true) for the selected well or not (false).
ShowSelectedWells	bool	Set or Get boolean value for whether to show selected well rectangles (true) or not (false).
ShowSubTitle	bool	Get or Set boolean for whether the microtitre plate SubTitle is to be shown (true) or not (false).
ShowSubTitleBorder	bool	Get or Set boolean for whether the microtitre plate SubTitles are to have frames (true) or not (false).
ShowTitle	bool	Get or Set boolean for whether the microtitre plate Title is to be shown (true) or not (false).
ShowTitleBorder	bool	Get or Set boolean for whether the microtitre plate Title is to have a frame (true) or not (false).
ShowWellAnnotation	bool	Set or Get boolean value for whether the well annotation is to be rendered (true) or not (false).
ShowWellAnnotationLineSeparator	bool	Set or Get boolean value for whether well annotation text that contains \r\n should be displayed with a horizontal line between the two segments or not.

Table 15: Description of Properties for Configuration

4.4.3 Fonts

Property	DataType	Description
CaptionBackColor	Color	Get or Set the background color for the microtitre plate Caption rectangle
CaptionFont	Font	Get or Set the Font for the microtitre plate Caption.
CaptionForeColor	Color	Get or Set the color for the microtitre plate Caption text
IdBackColor	Color	Get or Set the background color for the microtitre plate ID
IdFont	Font	Get or Set the Font for the microtitre plate ID.
IdForeColor	Font	Get or Set the color for the microtitre plate ID text
ScaleAnnotationFontLarger	bool	Set or Get boolean value for whether the well annotation font is to be scaled larger(true) or not (false).
ScaleAnnotationFontSmaller	bool	Set or Get boolean value for whether the well annotation font is to be scaled smaller(true) or not (false).
TitleBackColor	Color	Get or Set the background color for the microtitre plate Title rectangle
TitleFont	Font	Get or Set the Font for the microtitre plate Title.
TitleForeColor	Color	Get or Set the color for the microtitre plate Title text
SubTitleBackColor	Color	Get or Set the color for the microtitre plate SubTitle backcolor for Plate A or B
SubTitleFont	Font	Get or Set the Font for the microtitre plate SubTitle.
SubTitleForeColor	Color	Get or Set the color for the microtitre plate SubTitle text for Plate A or B
SubTitleLeftBackColor	Color	Get or Set the background color for the microtitre plate SubTitleLeft rectangle
SubTitleLeftForeColor	Color	Get or Set the color for the microtitre plate SubTitleLeft text
SubTitleRightBackColor	Color	Get or Set the background color for the microtitre plate SubTitleRight rectangle
SubTitleRightForeColor	Color	Get or Set the color for the microtitre plate SubTitleRight text
WellAnnotationColor	Color	Set or Get the font text color used for annotation within wells.
WellFont	Font	Set or Get the Font used for annotation

		within wells.

Table 16: Description of Properties for Fonts

4.4.4 Data for Text Fields

Property	DataType	Description
CaptionAlignment	ContentAlignment	Get or set the alignment for the microtitre plate Caption
PlateCaption	string	Get or Set the Caption for the microtitre plate A
PlateId	string	Get or Set the ID of Plate A
IdAlignment	ContentAlignment	Get or set the alignment for the microtitre plate ID for plate A or B. *Note: If the plate contains text for a Title then the ID will be rendered at the left side for Alignment of Left or Center and at the right side for Alignment of Right.*
SubTitleAlignment	ContentAlignment	Get or set the alignment for the microtitre plate subTitle for plate A or B
PlateSubTitle	string	Get or Set the SubTitle for microtitre plate A
PlateTitle	string	Get or Set the Title for the microtitre plate A
TitleAlignment	ContentAlignment	Get or set the alignment for the microtitre plate Title

Table 17: Description of Properties for Text Fields.

4.4.5 Well Images

Images can be displayed within wells in place of annotation or coloring and supported image types are .bmp, .tif, .jpg, .gif and .png. They are added to the plate using Well Image methods or drag-dropping one or more image files directly onto the wells.

The Boolean property ShowWellImage has to be set to true to activate rendering of the images within the wells. When images are smaller than the well size and it is desirable to

render them as large as the well size, then the property ScaleImageLargerToWellSize should be set to true.

4.4.5.1 WellImageStruct

Member	DataType	Description
Row	int	Row index of well within the plate.
Column	int	Column index of well within the plate.
ImageSetName	string	A name for the set of images to which the image belongs.
ImageCaption	string	Text to use when displaying a tooltip or caption for the image.
Filename	string	Filename including path of image.
WellImage	Image	Image to be displayed.

Table 18: Description of WellImage structure.

4.4.5.2 Well Image Properties

Property	DataType	Description
DragDropDirection	WellFillDirection	Set or Get direction which will be used to fill successive wells when more than one file is drag-dropped onto the plate. Down specification is for filling down a column from the well drop row and when the last row is reached the column number is incremented and filling continues from the first row. Across specification for filling in the well drop row across columns from the well drop column and when the last column is reached the row number is incremented and filling continues from the first column.
ScaleImageLargerToWellSize	bool	Set or Get boolean value that specifies whether images that are smaller than the well size are to be

		scaled larger to the well size (true) or not (false).
ShowWellImages	bool	Set or Get boolean value that specifies whether images are to be placed inside (true) of wells (true) or not (false). When images are used well annotation and coloring of wells are not rendered.
WellImageColumn	string	Set or Get the name of the column containing well images that is to be used when the ShowWellImages property is set to true.

Table 19: Description of Properties for Showing Images in Wells

4.4.5.3 Well Image Methods

Method	Description
AddWellImage(int Row, int Column, Image WellImage)	This method is used for adding Well Images where it sets the ImageSetName to 'WellImages' and DisplayName and FullName to an empty string.
AddWellImage(int Row, int Column, string ImageSetName, string ImageCaption, string FileName, Image WellImage)	This method is used for adding Well Images and setting members of the WellImage struct using the respective argument values.

Table 20: Description of WellImage Methods.

4.4.5.4 Well Image File Drag-Drop

One or more supported image file types can be selected and drag-dropped onto a well. When more than one image file is selected, images are automatically assigned to successive wells in the direction specified using the DragDropDirection property. The WellImage struct members are populated as follows:

- Row and Column index of well where image was dropped. Indexes are automatically adjusted when more than one image is dropped.
- ImageSetName is “DragDrop”
- ImageCaption is FileName of the image without extension

- FileName is the full FileName including path of the image.
- WellImage is the image.

4.4.6 Well Graphs - XY Data

Graphs of two dimensions (XY) can be displayed within wells. The data can be added to the system through methods or a data file containing formatted XY data drag-dropped onto a well.

4.4.6.1 Well XYDataStruct used for Graphs

Member	DataType	Description
Row	int	Row index of well within the plate.
Column	int	Column index of well within the plate.
XYPts	PointF[]	Collection of PointF objects containing X-axis and Y-axis values
XDisplaySpecification	NumberDisplaySpecification	Number Display Format specification for X-axis values
YDisplaySpecification	NumberDisplaySpecification	Number Display Format specification for Y-axis values
LineColor	Color	Color to use for the plotted line.
PlotTitle	string	Title of plot data
PlotCaption	string	Caption for plot data
FileName	string	Filename including path of xydata.

Table 21: Description of XYDataStruct used for graphs.

4.4.6.2 Well XY Graph Properties

Property	DataType	Description
DragDropDirection	WellFillDirection	Set or Get direction which will be used to fill successive wells when more than one file is drag-dropped onto the plate. Down specification is for filling down a column from the well drop row and when the last row is reached the column number is

		incremented and filling continues from the first row. Across specification for filling in the well drop row across columns from the well drop column and when the last column is reached the row number is incremented and filling continues from the first column.
ShowWellGraphs	bool	Get or Set value for whether to show well graphs (true) or not (false).

Table 22: Description of Properties for XY Graphs

4.4.6.3 Well XYGraph Methods

Method	Description
AddWellXYData(int Row, int Column, float[] X, float[] Y)	Method for adding XY data for a Well where X and Y display format specifications have the display name set to X and Y respectively with other parameters set to string.Empty, 0, false and color black.
AddWellXYData(int Row, int Column, float[] X, float[] Y, string PlotTitle, string PlotCaption, string FileName, Color LineColor, NumberDisplaySpecification XDisplaySpecification , NumberDisplaySpecification YDisplaySpecification)	Method for adding XY data for a well. The display format specifications need to be set as well as the graphing line color.

Table 23: Description of XY Methods.

4.4.6.4 Well XY Data File Drag-Drop

Text files that have one of the following formats can be drag-dropped onto a well.

1. Text file containing one line for each XY data pair with the X value being first and separated from the Y value with a tab character.
2. Same as the previous description with the addition that the first line contains a label or title for the X and Y values separated by a tab character.

When more than one XY Data file is selected, data are automatically assigned to successive wells in the direction specified using the DragDropDirection property. The XYDataStruct members are populated as follows:

- Row and Column index of well where image was dropped. Indexes are automatically adjusted when more than one image is dropped.
- LineColor is Color.Black
- PlotTitle is the FileName of the XYData without extension.
- PlotCaption is string.empty
- FileName is the full FileName including path of the XYData
- X and Y DataDescription will use the inputted titles otherwise “X” and “Y” respectively for the DisplayName, no units, 0 decimal places, 0 scalefactor, no scientific notation and no thousands separator.

4.4.7 Well Tooltips

Well Tooltips can be customized by setting to true or false specific ShowWell<column> property values and in addition other columns can be specified by populating the TooltipWellOtherColumns ArrayList with the column names.

Property	DataType	Description
OmitColumnHeadingsInTooltip	bool	Set or Get value for whether to show column names of selected 'Other Columns' in the tooltip. If it is true then the tooltip will have a format of 'ColumName = Value'; otherwise, 'Value'.
ShowTooltips	bool	Set or Get boolean value for whether to show tooltips(true) or not(false).
ShowTooltipWellDescription	bool	Set or Get boolean value for whether the Well Description is to be included in the tooltip.
ShowTooltipWellLocation	bool	Set or Get boolean value for whether the Well Location is to be included in the tooltip.
ShowTooltipWellID	bool	Set or Get boolean value for whether the Well ID is to be included in the tooltip.
ShowTooltipWellType	bool	Set or Get boolean value for whether the Well Type is to be included in the tooltip.
ShowTooltipWellOtherColumns	bool	Set or Get value for whether to show selected other columns than the ones in the standard set.
ShowTooltipWellXValue	bool	Set or Get boolean value for whether the Well X-Value is to be included in the tooltip.

ShowTooltipWellYValue	bool	Set or Get boolean value for whether the Well Intensity Value is to be included in the tooltip.
TooltipWellOtherColumns	ArrayList	The ArrayList contains columns that are to be displayed in the Well tooltip that are not included in the standard set (XValue, YValue, Row, Column, Type, ID, Description, N and WellLocation) of columns. Any specified columns that are in the standard set will be removed.
WellInfo	string	Set or Get the tooltip information for the well the mouse is currently over.

Table 24: Description of Properties for Well Tooltips

4.4.8 Well Type Shape Mapping

Property	DataType	Description
ImageShapeBuffer	Image	Get the image for the ShapeBuffer
ImageShapeControl	Image	Get the image for the ShapeControl
ImageShapeEmpty	Image	Get the image for the ShapeEmpty
ImageShapeOther	Image	Get the image for the ShapeOther
ImageShapeReagent	Image	Get the image for the ShapeReagent
ImageShapeReference	Image	Get the image for the ShapeReference
ImageShapeProbe	Image	Get the image for the ShapeProbe
ImageShapeSample	Image	Get the image for the ShapeSample
ImageShapeStandard	Image	Get the image for the ShapeStandard
ImageShapeTag	Image	Get the image for the ShapeTag
ImageShapeTarget	Image	Get the image for the ShapeTarget
ShapeBuffer	WellShapeStyles	Set or get the shape of a well designated as a Buffer.
ShapeControl	WellShapeStyles	Set or get the shape of a well designated as a Control.
ShapeEmpty	WellShapeStyles	Set or get the shape of a well designated as a Empty.
ShapeOther	WellShapeStyles	Set or get the shape of a well designated as a Other.
ShapeReagent	WellShapeStyles	Set or get the shape of a well designated as a Reagent
ShapeReference	WellShapeStyles	Set or get the shape of a well designated as a Reference.

ShapeProbe	WellShapeStyles	Set or get the shape of a well designated as a Probe.
ShapeSample	WellShapeStyles	Set or get the shape of a well designated as a Sample.
ShapeStandard	WellShapeStyles	Set or get the shape of a well designated as a Standard.
ShapeTag	WellShapeStyles	Set or get the shape of a well designated as a Tag.
ShapeTarget	WellShapeStyles	Set or get the shape of a well designated as a Target
WellShapeImages	ImageList	Get an ImageList that contains the allowed well shapes. These images are used in conjuction with WellShapeStyles values and WellTypes enum values.

Table 25: Description of properties for Well Content Type to Well Shapes

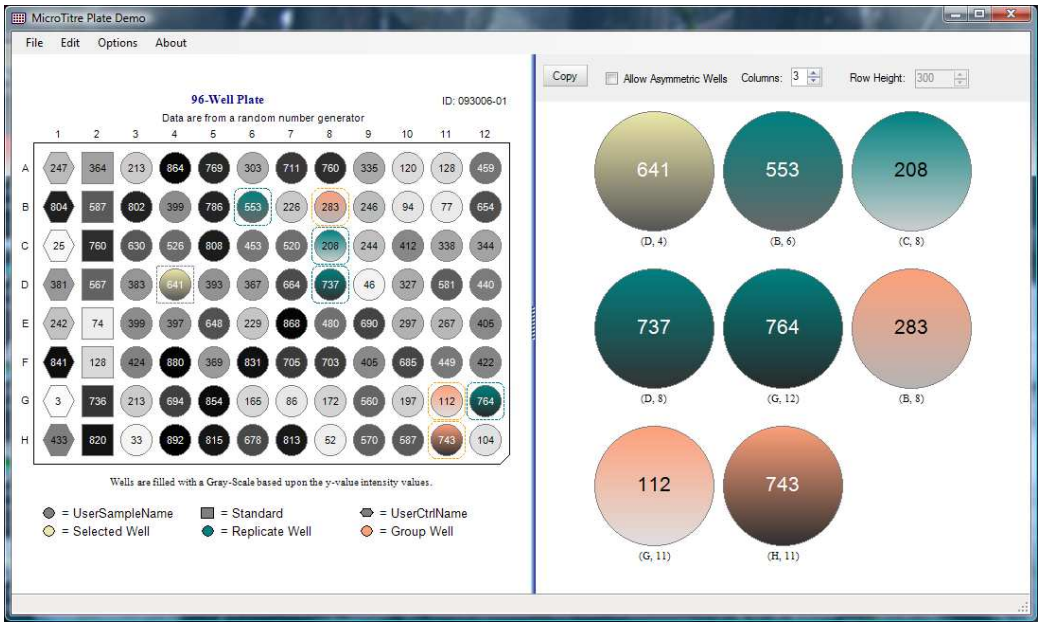
4.4.9 Well Magnifier

BxMicroTitrePlate has options for creating an embedded magnifier viewer control or a popup magnifier viewer window which can be used to obtain larger views of well annotation, graphs and images and can be used to compare a collection of wells. Both viewers can be displayed independently or simultaneously, can be independently sized and are automatically synchronized which faciltates showing on different monitors. When a Well Magnifier viewer is activated, then all selected wells including replicates and group wells, if they are to be shown, will be magnified in the viewer(s) where the well size is based upon the size of the individual viewer. In this example there are 8 selected wells arranged in three columns as indicated by the column Numeric_Up_Down selector. The asymmetric option relaxes the restriction that well height must equal well width that is useful when showing graphs or asymmetric images as seen in views presented below. The Copy button copies the content of the magnifier view to the clipboard.

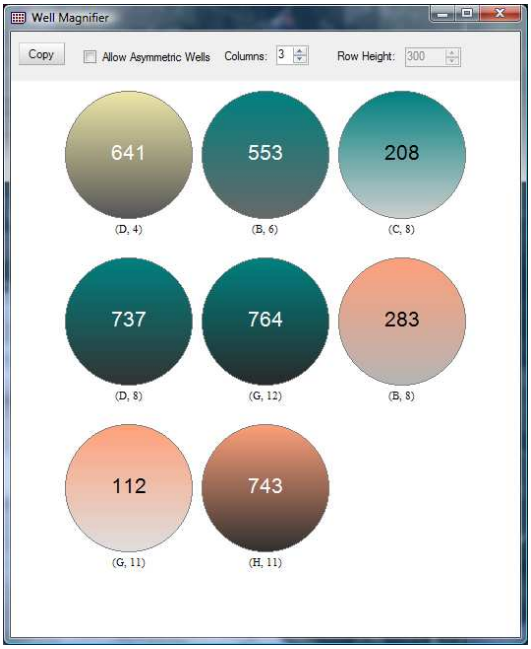
Note: If there are no selected wells, then as the Mouse moves over a well, it will automatically be shown in the viewer where its well shape and content will automatically be magnified to fit the viewer. This permits a quick detail view of a well's content as the mouse moves over it which is especially useful for plates with more than 96 wells or when it has an image or graph.

4.4.9.1 Embedded Control

The embedded viewer, shown in the right panel of the demo application, can be programmatically added to any Windows object that supports adding of controls.



4.4.9.2 Window Form



Setting the property ShowMagnifierWindow = true will show/launch a movable Well Magnifying Window where the well that the cursor is over will be redrawn to a size based upon the resizable client area of the Window. The window is closed by setting the ShowMagnifierWindow to false or clicking on the Red close button in the upper right corner of the window.

As the cursor moves from one well to another the new well is displayed. When images are displayed within wells, they will be scaled to the larger well size only if the property ScaleImageLargerToWellSize is true. This could be used for plates with many wells where well sizes are quite small and when images are displayed within wells to enable viewing of well annotation or larger images.

Property	DataType	Description
ShowMagnifierWindow	bool	Set or Get boolean that Shows(true) or Closes(false) the Magnifier Window.

Table 26: Description of properties for the Well Magnifier Window

4.4.10 Miscellaneous Properties

Property	DataType	Description
Columns	int	Get the number of columns in the microtitre plate. Use SetPlateRowColumnSize() method to set the number of Columns or one of the SetPlateDataTable…() methods.
GrayScaleYValue	bool	Set to true to use YValue for color intensity shading of wells; otherwise false to use XValue.
PlateHeight	int	Get height of the microtitre plate in pixels.
PlateLocation	Point	Get the location (upper left corner) of the Microtitre plate.
PlateWidth	int	Get width of the microtitre plate in pixels.
PlateContainerHeight	int	Get the height of the rectangle containing the microtitre plate
PlateContainerLocation	Point	Get the point location of the rectangle containing the microtitre plate
PlateContainerWidth	int	Get the width of the rectangle containing the microtitre plate
Rows	int	Get the number of rows in the microtitre plate. Use SetPlateRowColumnSize() method to set the number of Rows or one of the SetPlateDataTable…() methods.
SuspendRefresh	Bool	Set or Get value for controlling whether Refresh() should be called after a property value is set. It should be set to false when setting many property values to eliminate screen flicker. After the property values are set then Refresh() should be called to update the view with the changes.
TotalWells	int	Get total number of Wells in the microtitre plate.

XValueDecimalPlaces	int	Set or Get number of decimal places to display for X-Value
XValueFormat	string	Get format string used when displaying X-Values. The format is based upon the current settings of XValueDecimalPlaces, XValueUseThousandSeparator and XValueScientificNotation.
XValueName	string	Set or Get Name to use for X-Value
XValueScalingFactor	int	Set or Get power of 10 scale factor to apply to X-Value
XValueScientificNotation	bool	Set or Get boolean value for whether to express X-Value in scientific notation or not.
XValueUnits	string	Set or Get Units to use with X-Value
XValueUseThousandSeparator	bool	Set or Get boolean value for whether to express X-Value with a thousand separator or not.
YValueDecimalPlaces	int	Set or Get number of decimal places to display for Y-Value
YValueFormat	string	Get format string used when displaying Y-Values. The format is based upon the current settings of YValueDecimalPlaces, YValueUseThousandSeparator and YValueScientificNotation.
YValueName	string	Set or Get Name to use for Y-Value
YValueScalingFactor	int	Set or Get power of 10 scale factor to apply to Y-Value
YValueScientificNotation	bool	Set or Get boolean value for whether to express Y-Value in scientific notation or not.
UseIndividualWellColors	bool	Get or Set value that indicates whether individual well colors are to be used(true) where specified or to use (false) the global well fill color.
UseXValueForAnnotation	bool	Set or Get boolean value for whether the X-Value is to be used for annotation (true) or not (false). The annotation will use XValueFormat and if UseYValueAnnotation is also set to true then the X-Value will be on top of the Y-Value. Default is

		false.
UseYValueForAnnotaiton	bool	Set or Get boolean value for whether the Y-Value is to be used for annotation (true) or not (false). The annotation will use YValueFormat and if the UseXValueAnnotation is also set to true then the X-Value will be on top of the Y-Value. Default is false.
YValueUnits	string	Set or Get Units to use with Y-Value
YValueUseThousandSeparator	bool	Set or Get boolean value for whether to express Y-Value with a thousand separator or not.
WellLayout	ArrayList	Get an ArrayList containing a Well struct for each well in the plate. The Well structure contains the location and size of a well through the WellRect member along with other indexing information.

Table 27: Description of miscellaneous properties

4.4.11 Alphabetical List of Properties with Default Values

This section provides an alphabetical list of all property values

Property	DataType	Default Value
CaptionAlignment	ContentAlignment	ContentAlignment. MiddleLeft
CaptionBackColor	Color	Color.Transparent
CaptionBorderColor	Color	Color.Black
CaptionFont	Font	Font("Arial", 8.25f)
CaptionForeColor	Color	Color.Black
Columns	int	Read Only – returns 12 initially
DragDropDirection	WellFillDirection	WellFillDirection.Down
DrawWellMode	DrawMode	DrawMode.Normal
DrawWellNormal	bool	False if DrawMode.OwnerDraw
ExcludeColor	Color	Color.Red
FocusRectangleColor	Color	Color.Red
GrayScaleYValue	bool	true
GroupWellFillColor	Color	Color.LightSalmon
GroupWellLineColor	Color	Color.DarkGray
HighlightSelectedWellsWithColors	bool	true
HighlightSelectedWellsWithDashLine Rectangles	bool	true
IdBackColor	Color	Color.Transparent
IdFont	Font	Font("Arial", 8.25f)
IdForeColor	Font	Color.Black

IdPlateAlignment	ContentAlignment	ContentAlignment. MiddleRight
ImageShapeBuffer	Image	Circle
ImageShapeControl	Image	Hexagon
ImageShapeEmpty	Image	Circle
ImageShapeOther	Image	Triangle
ImageShapeProbe	Image	Square with protrusion (Key)
ImageShapeReagent	Image	Circle
ImageShapeReference	Image	Diamond
ImageShapeSample	Image	Circle
ImageShapeStandard	Image	Square
ImageShapeTag	Image	Bar_Graph
ImageShapeTarget	Image	Square with insertion region (Lock)
IncludeReplicateAndGroupWellsInLegend	bool	true
LabelRowsWithNumbers	bool	false
LegendKeyAlignment	ContentAlignment	ContentAlignment .MiddleCenter
OutLineShape	bool	true
OutLineShapeColor	Color	Color.Gray
OmitColumnHeadingsInTooltip	bool	false
PlateAlignment	ContentAlignment	ContentAlignment .MiddleCenter
PlateCaption	string	string.Empty
PlateColor	Color	Color.White
PlateDataTable	DataTable	Contains the DataTable for Plate A
PlateBorderColor	Color	Color.Black
PlateContainerBackColor	Color	Color.White
PlateContainerBorderColor	Color	Color.Black
PlateContainerHeight	int	Calculated Value based on selected GUI
PlateContainerLocation	Point	Calculated Value based on selected GUI options as function of the graphics device
PlateContainerWidth	int	Calculated Value based on selected GUI options as function of the graphics device
PlateHeight	int	Calculation based on Rows and Well Height
PlateId	string	string.Empty
PlateLocation	Point	Dynamically determined based upon settings such as PlateAlignment and whether Title, Subtitles, ID and Caption are selected for display.
PlateSubTitle	string	string.Empty
PlateTitle	string	string.Empty
PlateWellColor	Color	Color.Gray
PlateWellPrimaryColor	Color	Color.Gray
PlateWidth	int	Calculation based on Columns and Well Width
ReplicateWellFillColor	Color	Color.LightSteelBlue
ReplicateWellLineColor	Color	Color.DarkGray
RotatePlate	bool	false
Rows	int	Read Only – returns 8
SaveSettingsAutomatically	bool	true

SaveSettingsFileName	string	"BxMicroTitrePlatePropertySettings.xml"
ScaleAnnotationFontLarger	bool	false
ScaleAnnotationFontSmaller	bool	true
ScaleImageLargerToWellSize	bool	false
SelectedWellFillColor	Color	Color.PaleGoldenRod
SelectedWellLineColor	Color	Color.DarkGray
ShapeControl	WellShapeStyles	WellShapeStyles.Hexagon
ShapeEmpty	WellShapeStyles	WellShapeStyles.Circle
ShapeOther	WellShapeStyles	WellShapeStyles.Triangle
ShapeSample	WellShapeStyles	WellShapeStyles.Circle
ShapeStandard	WellShapeStyles	WellShapeStyles.Square
ShapeTag	WellShapeStyles	WellShapeStyles.Bar_Graph
ShowCaption	bool	false
ShowCaptionBorder	bool	false
ShowColLabelsBottom	bool	false
ShowCornerCutBottomLeft	bool	false
ShowCornerCutBottomRight	bool	true
ShowCornerCutTopLeft	bool	false
ShowCornerCutTopRight	bool	false
ShowFocusRectangle	bool	false
ShowGroup	bool	true
ShowLegend	bool	false
ShowLegendBorder	bool	false
ShowLegendVertical	bool	false
ShowMagnifierWindow	bool	false
ShowPlateBorder	bool	true
ShowPlate3D	bool	true
ShowPlateContainerBorder	bool	false
ShowPlateID	bool	true
ShowRowColLabels	bool	true
ShowReplicates	bool	true
ShowSelectedWells	bool	true
ShowSubTitle	bool	false
ShowSubTitleBorder	bool	false
ShowTitle	bool	true
ShowTitleBorder	bool	false
ShowTooltips	bool	true
ShowTooltipWellLocation	bool	true
ShowTooltipWellType	bool	false
ShowTooltipWellID	bool	true
ShowTooltipWellDescription	bool	false
ShowTooltipWellOtherColumns	bool	false
ShowTooltipWellXValue	bool	false
ShowTooltipWellYValue	bool	false
ShowWellAnnotation	bool	true
ShowWellAnnotationLineSeparator	bool	false
ShowWellGraphs	bool	false
ShowWellImages	bool	false
ShowWellMagnifierControlViewer	bool	false
ShowWellMagnifierPopupViewer	bool	false
SubTitleAlignment	ContentAlignment	ContentAlignment .MiddleCenter
SubTitleBackColor	Color	Color.Transparent

SubTitleFont	Font	Font("Arial", 8.25f)
SubTitleForeColor	Color	Color.Black
SuspendRefresh	bool	false
TitleAlignment	ContentAlignment	ContentAlignment .MiddleCenter
TitleBackColor	Color	Color.Transparent
TitleBorderColor	Color	Color.Black
TitleFont	Font	Font("Arial", 8.25f)
TitleForeColor	Color	Color.Black
TooltipWellOtherColumns	ArrayList	No entries.
TotalWells	int	Calculated value from Rows and Columns
UseIndividualWellColors	bool	false
UsePlateContextMenu	bool	true
UseXValueForAnnotation	bool	false
UseYValueForAnnotaiton	bool	false
WellAnnotationColor	Color	Color.Black
WellEmptyColor	Color	Color.LightGray
WellFillStyle	WellFillStyles	WellFillStyles.Gradient
WellFont	Font	Font("Arial", 8.25f)
WellInfo	string	string.Empty
WellLayout	ArrayList	ArrayList contains one Well structure for each Well in the plate. The Well.WellRect member contains the location of the well and is updated as needed.
WellImageColumn	String	string.Empty
WellShapeImages	ImageList	Contains the shape images in the order synchronized with the WellShapeStyles enum values.
XValueDecimalPlaces	int	2
XValueFormat	string	"#,##0.00"
XValueName	string	"X-Value"
XValueScalingFactor	int	0
XValueScientificNotation	bool	false
XValueUnits	string	string.Empty
XValueUseThousandSeparator	bool	true
YValueDecimalPlaces	int	2
YValueFormat	string	"#,##0.00"
YValueName	string	"Y-Value"
YValueScalingFactor	int	0
YValueScientificNotation	bool	false
YValueUnits	string	string.Empty
YValueUseThousandSeparator	bool	true

Table 28: Alphabetical List of Properties with their Default Values

4.5 Events

4.5.1 Error Event

The error event is triggered whenever any of the following errors are encountered. The Error event passes back the Error Code and Error Message. These errors should be handled and in most cases a plate with empty wells will be rendered.

```
mtp.Error += new BxTools.ErrorEventHandler(mtp_Error);
```

Error Code	Error Message
E1	DataTable was null.
E2	DataTable did not contain any rows.
E3	DataTable contained less columns than the minimum number of columns.
E4	DataTable was missing column: 'Column Name'.
E5	DataTable Column 'Column Name' has DataType: 'DataType value' but DataType should be: 'DataType value'.
E6	DataTable is not in the correct Row - Column order.
E7	Primary color was not changed due to new primary color not being one of the supported primary colors.
E8	DataTable row count (Count) does not match the number of wells (Rows x Columns) in the plate.
E9	DataTable is missing required data in columns N, Row or Column.
E10	DataTable contained an invalid Row number.
E11	DataTable contained an invalid Column number.
E12	DataTable contained an invalid number in column N.
E13	Invalid Row-Column or Table Row index was entered.
E14	Cannot assign a replicate number to selected wells since wells already contain conflicting replicate numbers.
E15	Cannot assign selected wells as replicates since wells have conflicting Well Types.
E16	Exception Error in exporting plate DataTable.
E17	There was no data in Plate DataTable to export.
E18	Cannot assign a group number to selected wells since wells already contain conflicting group numbers.
E19	Cannot assign selected wells to a group since wells have conflicting Well Types.
E20	Well color was not set due to an invalid Row or Column Value.
E21	Well color was not removed due to an invalid Row or Column Value.
E22	Wells were not selected set due to an invalid Row or Column Value.
E23	Wells were not unselected set due to an invalid Row or Column Value.
E24	Plate DataTable was not set which could be due to an empty

	DataTable or a Row-Column specification error.
XE1	Incorrect XML file.\r\n The first XML tag must be: 'BxMicroTitrePlateSettings ...' Processing of file was terminated.
XE2	An exception error occurred while loading property settings from an XML file.
XE3	An exception error occurred while saving the current property settings to an XML file.
XE4	Invalid XML property setting tag: (tag that is invalid).
XE5	XML file does not exist. (Name of File).

Table 29: List of Error Codes and Messages

4.5.2 Mouse Events

```
mtp.mtpMouseDown += new
      MicroTitrePlateMouseDownEventHandler(mtp_mtpMouseDown);
mtp.mtpMouseUp += new
      MicroTitrePlateMouseUpEventHandler(mtp_mtpMouseUp);
mtp.mtpMouseMove += new
      MicroTitrePlateMouseMoveEventHandler(mtp_mtpMouseMove);
mtp.mtpMouseEnter += new
      MicroTitrePlateMouseEnterEventHandler(mtp_mtpMouseEnter);
mtp.mtpMouseLeave += new
      MicroTitrePlateMouseLeaveEventHandler(mtp_mtpMouseLeave);
```

4.5.3 WellLayout Event

The WellLayout event is triggered whenever well sizes are calculated such as when the size of the plate changes.

```
mtp.mtpWellLayout += new
      MicroTitreWellLayoutEventHandler (mtp_mtpWellLayout);
```

4.5.4 Blinking Well Events

Two blinking well events are available to notify the calling application when wells have started blinking and when they have stopped.

```
mtp.mtpBlinkingStarted += new
       MicroTitreWellBlinkingStartedEventHandler(mtp_mtpBlinkingStarted);
mtp.mtpBlinkingStopped += new
       MicroTitreWellBlinkingStoppedEventHandler(mtp_mtpBlinkingStopped);
```


4.6 Structs and Enums

This section contains descriptions of the public structs and enums available to the developer.

4.6.1 Well struct

A Well structure contains the layout data for a well within a plate. There is one Well structure for each well in the public property WellLayout ArrayList. This information can be used for adding specialized GUI feedback or annotation. For example to draw a blue rectangle around the well with zero-based index 'N":

```
// Get the well object of well N
BxTools.MicroTitrePlate.Well w =
     (BxTools.MicroTitrePlate.Well)mtp.WellLayout[N];

// Graphics Object for User Control
Graphics gw = mtp.CreateGraphics();

// draw a blue rectangle around the well
gw.DrawRectangle(Pens.Blue, w.WellRect);
```

Member	Data Type	Description
TableRow	int	Contains the zero-based row number for the well relative to the original data source table
Row	int	Contains the row number of the well in the plate
Column	int	Contains the column number of the well in the plate
WellRect	Rectangle	Contains the X, Y location relative to the origin of the drawing rectangle, width and height of the rendered well.

Table 30: Description of Well struct members

4.6.2 WellShapeStyle enum

This enum is used to designate the shape to be used for a well type.

Member	Description
Circle	Used to designate the Well shape as a circle.
Square	Used to designate the Well shape as a square.
Hexagon	Used to designate the Well shape as a Hexagon.
Gaussian_Graph	Used to designate the Well shape to have the appearance of a gaussian graph. Intended for wells that contain a Tag.
Triangle	Used to designate the Well shape as a Triangle.
ShadingUnderCurve	Used to designate the Well shape to have the appearance of

	representing a percentage by the amount of shading under a curve.
Bar_Graph	Used to designate the Well shape to have the appearance of a bar graph. Intended for wells that contain a Tag.
Hidden	Used to designate the Well that is hidden.

Table 31: Description of WellShapeStyle enum members.

4.6.3 WellTypes

This enum is used to designate the type of content contained within a well. A WellShapeStyle can be assigned to a WellType so that all wells with the same type will have the same shape style.

Member	Description
Buffer	A well that contains a buffer solution
Control	A well that contains a control
Empty	An empty well
Hidden	A well that is to hidden, that is not displayed
Other	A well that contains something that does not fit any of the other enum values
Probe	A well that contains a probe
Reagent	A well that contains reagent
Reference	A Well that is to be used as a reference
Sample	A well that contains a sample
Standard	A well that contains a standard
Tag	A well that contains a flourescent tag or dye color
Target	A well that contains a target

Table 32: Description of WellTypes enum members.

4.6.4 PlateSelection

This enum is used to select which plate is to be displayed. For example:

```
mtp.SelectedPlate = BxTools.MicroTitrePlate.PlateSelection.Plate;
```

Member	Description
Empty	Used when there is no plate data (a DataTable has not been assigned or populated) or to designate an empty or new plate.
Plate	Used for Plate

Table 33: Description of PlateSelection enum members.

5 XML Settings File

All of the properties and some of the data structures can be saved to an XML file and loaded to initialize the plate. The following methods are used to save and load the XML file from a hard drive. In addition, two methods are used to write and read a string containing the property values in XML format. The intent of these are to store and retrieve the settings from a database.

Method	Description
SavePropertySettings()	This method writes the current property settings to the BxMicroTitrePlatePropertySettings.xml file.
SavePropertySettings(string xmlPropertyFileName)	This method writes the current property settings to the destination filename with directory path. If the filename is empty then the following default file name is used: BxMicroTitrePlatePropertySettins.xml file.
SavePropertySettingsToString()	This method writes the current property settings to a string that contains the settings in an XML format.
LoadPropertySettings()	This method loads property settings from the BxMicroTitrePlatePropertySettins.xml file.
LoadPropertySettings(string xmlPropertyFileName)	This method loads property settings from the input filename with directory path. If the filename is empty then the following default file name is used: BxMicroTitrePlatePropertySettins.xml file.
LoadPropertySettingsFromString(string xmlStringData)	This method is used to load property settings from a string that contains settings in an XML format. It first converts the string to a memory stream and then creates an XmlTextReader from the stream.

Table 34: XML Settings Save and Load Method descriptions.

6 Plate Properties Dialog Box

BxMicroTitrePlate provides a Properties Dialog Box that can be added to an application for allowing a user to customize the plate graphic by selecting and inputting property values.

NOTE: All properties can be set programmatically; therefore, using the Properties Dialog Box is simply a design issue of whether to provide the user with customization capability or not.

The dialog box has two sections - the left section contains names of property forms and the right section contains the selected form. The intent of the property form name is to represent a logical grouping of settable properties although sometimes this is not always accomplished.

Clicking on a property form name will cause its form to be displayed with its current values and to permit changing of the values. Once all values have been changed for each of the forms, clicking on the Apply button will cause the Property Dialog Box to close and apply all changes. Clicking on the cancel button will close the form and cancel all changes made to any of the forms.

6.1 Methods for managing forms

The following methods are available for managing forms that are to be made available to the user.

Method	Return Value	Description
`HidePropertyForm( PropertyForms pfEnum)`	True if form hidden; otherwise false.	This method will hide the specified form unless it is the only available form. That is, the name of the form will not be shown in list of available forms.
`ResetPropertyFormList()`	void	This method resets the list of available forms to the original full list.
`ShowOnlyPropertyForm( PropertyForms pfEnum)`	void	This method removes all forms and adds the specified form.
`ShowPropertyForm( PropertyForms pfEnum)`	void	This method can be used to show a form by adding it to the available list and then selecting it or if it is already in the list, then select it.

Table 35: List of methods used to manage forms in Properties Dialog Box.

PropertyForms is a public enum that has a member for each available form. The following code example shows its use in the HidePropertyForm() method to hide the Configuration form.

```
bool fhidden =
   mtpProperties.HidePropertyForm(
      BxMicroTitrePlate.MTPProperties.PropertyForms.Configuration);
```

6.2 Code Snippet for using Properties Dialog Box

The following code could be used to instantiate and show the Properties Dialog Box.

```
// create an instance of the BxMicroTitrePlate properties Dialog Box
BxMicroTitrePlate.MTPProperties mtpProperties = new
        BxMicroTitrePlate.MTPProperties();

// if the size of the plate is permitted to change, then the
// PlateSizeChangeEvent should be subscribed to. This event is
// triggered anytime that a new plate size is selected or if
// the rows or columns are changed when the SpecifySize option is
// selected.
mtpProperties.PlateSizeChanged += new
BxMicroTitrePlate.PlateSizeChangeEvent(mtpProperties_PlateSizeChanged);

// if the plate size is permitted to change then set AllowPlateSizes to
// true.
mtpProperties.AllowPlateSizes = true;

// Calling the CustomizeMicrotitrePlate() method and passing it an
// instance of MicroTitrePlate will initialize all of the Property
// Forms with the current values and then open the DialogBox
// The DialogResult will be returned.
DialogResult res = mtpProperties.CustomizeMicrotitrePlate(mtp);

// If the apply button is clicked the return value is DialogResult.OK
// and the customize method automatically updates the BxMicroTitrePlate
// object
if (res == DialogResult.OK)
{
      // perform any additional tasks
}
```

6.3 Properties Dialog Box Properties

The Dialog Box has properties that control whether or not additional options are made available in the forms.

Property	Data Type	Default Value	Description
AllowPlateAlignment	`bool`	`false`	This shows and enables options in the Configuration Dialog where the alignment of the plate in the drawing area can be set.
AllowPlateDrawingSizes	`bool`	`false`	This shows and enables options in the Configuration Dialog where the different drawing sizes of the plate can be selected.
AllowPlateSizes	`bool`	`true`	This shows and enables options permitting the setting of the plate size by selecting the number of wells on the plate or specifying a custom size.

Table 36: Description of Properties directly related to the Dialog Box

6.4 Dialog Box Forms

This section contains a description of the property forms in the Property DialogBox. The intent of the forms is to provide a graphical user interface for customizing the appearance and text labeling of the microtitre plate where each form represents, as close as possible, a category of settable properties. When the mouse pointer is moved over a button, checkbox, textbox, radio button or combo box an informative tooltip is displayed describing the associated property. The forms will have additional properties if two plates are being managed at the same time by the control. Two plates

6.4.1 Colors

This form is used to set colors of different components of BxMicroTitrePlate such as the background colors of the plate and its container. The color within the rectangle to the left of the 'Color' buttons shows the currently selected color and the rectangle to the left of the Well Fill Style combo box will provide a view of the selected style.

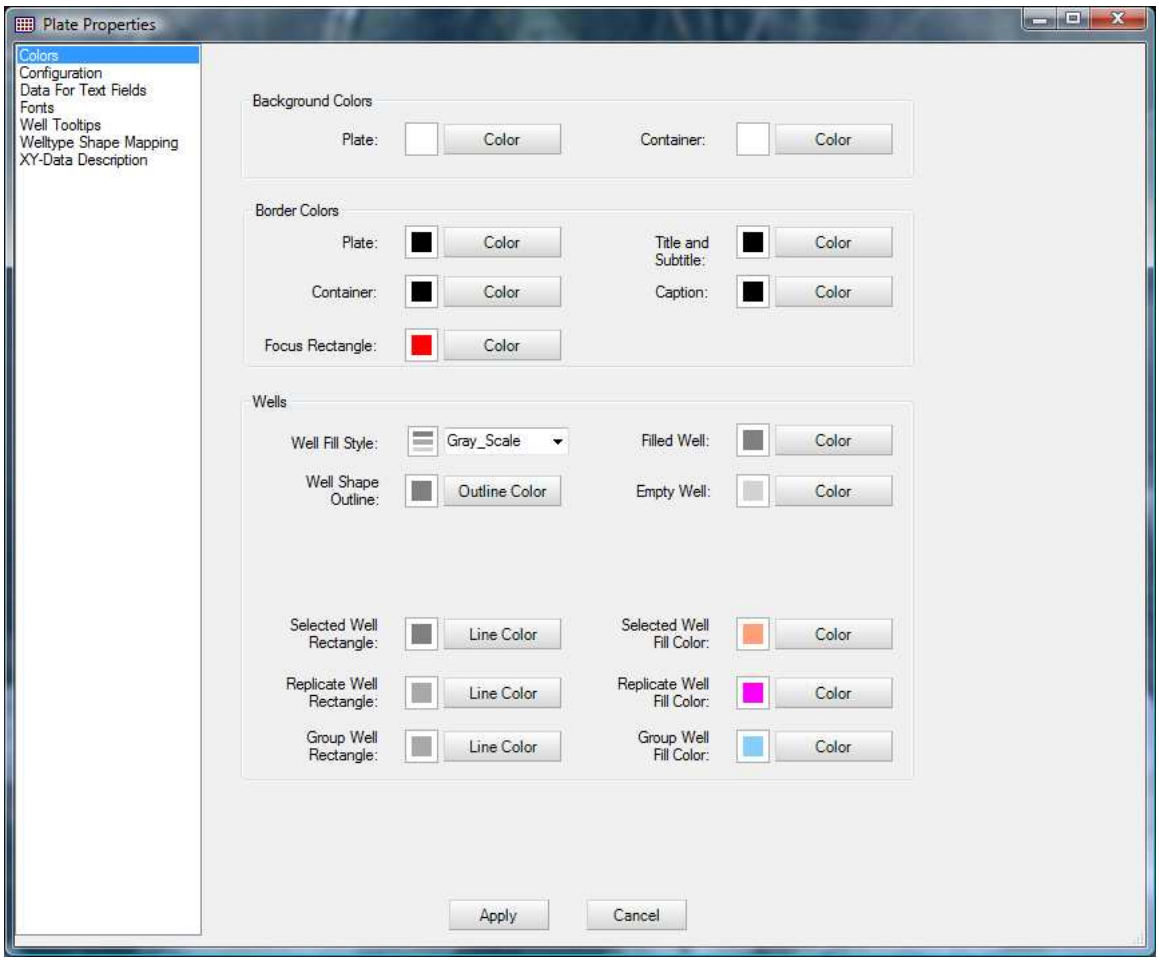


Figure 20: Form for selecting plate colors.

6.4.2 Configuration

This form is used configure the plate as to the number of wells and their layout. It also allows for setting whether certain components of the control are to be displayed such as the Plate ID, title, subtitle, borders and well annotation. The Replicates checkbox is used for specifying whether replicate wells for a selected well be highlighted or not.

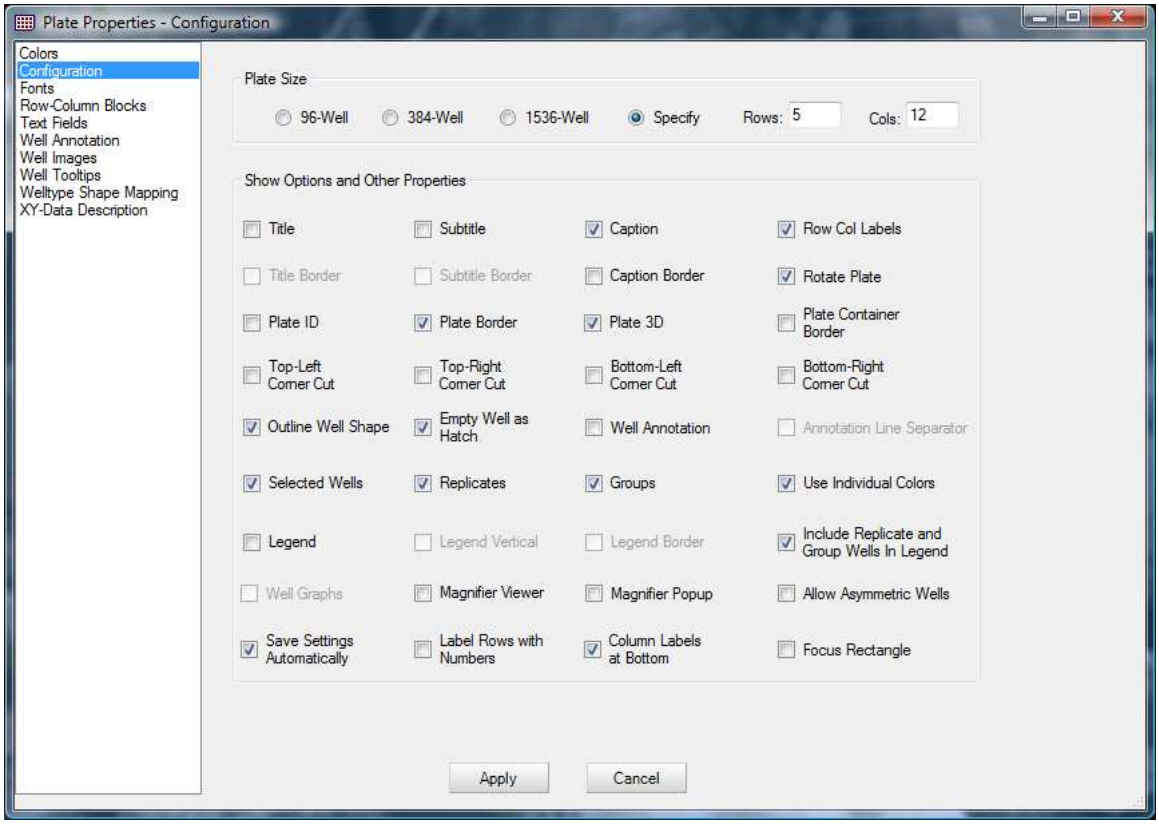


Figure 21: Form for selecting configuration options

6.4.3 Fonts

This form is used to define the font style, color of text and background for each of the different text fields available in the microtitre plate control. The content of the TextBoxes on the dialog box provides immediate feedback by showing the name of the selected font in the selected style and text color on the specified background color. An example is the Font Style of Arial with white text colors on a Teal background for the ID.

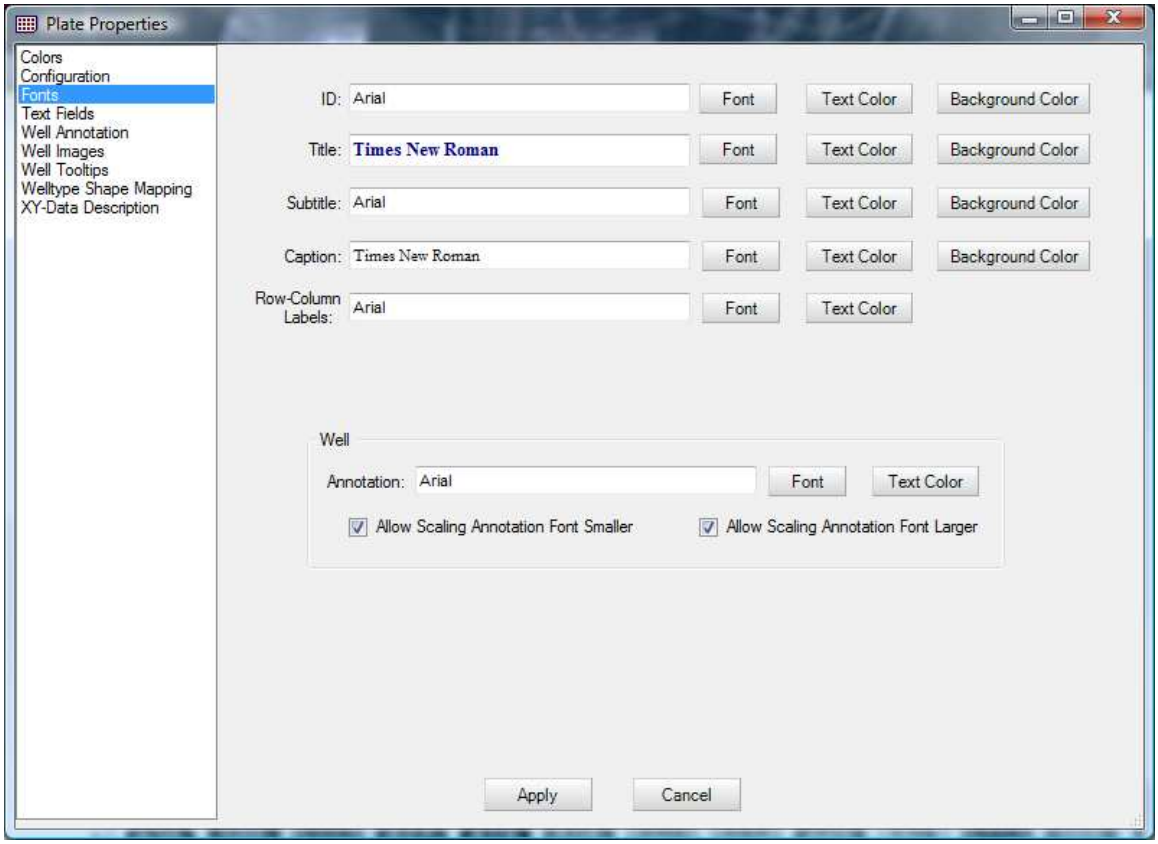


Figure 22: Form for selecting fonts and their text and background colors.

In the Well section two check boxes are provided, that when checked, will perform automatic scaling of the font used for annotation within the wells as the size of wells are changed as the size of the control is changed. Figure 23 shows a composite of a plate that contains annotation in three sizes where the annotation within each well has been automatically scaled as the well size changes.

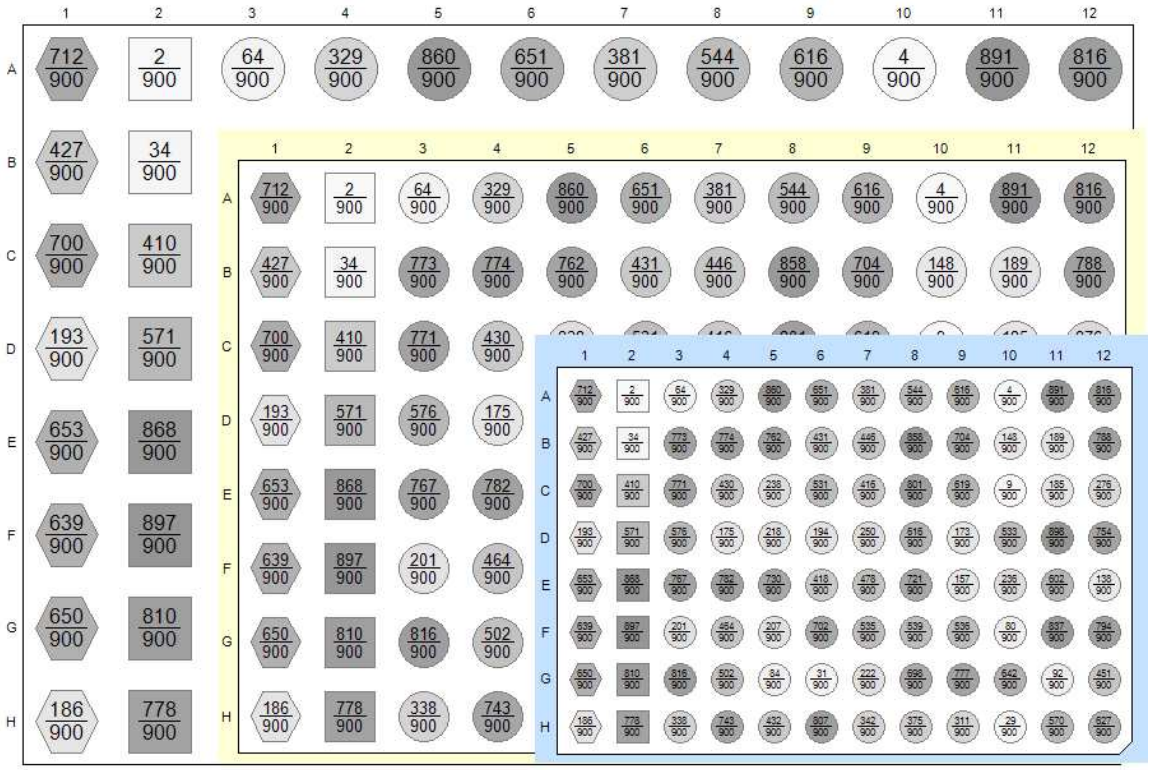


Figure 23: Composite of the same plate in three sizes showing annotation size scaling

6.4.4 Data for Text Fields

This form provides a set of text boxes for entering data and specifying alignment of data to be displayed in their respective text areas. Multiple lines can only be entered for the Caption.

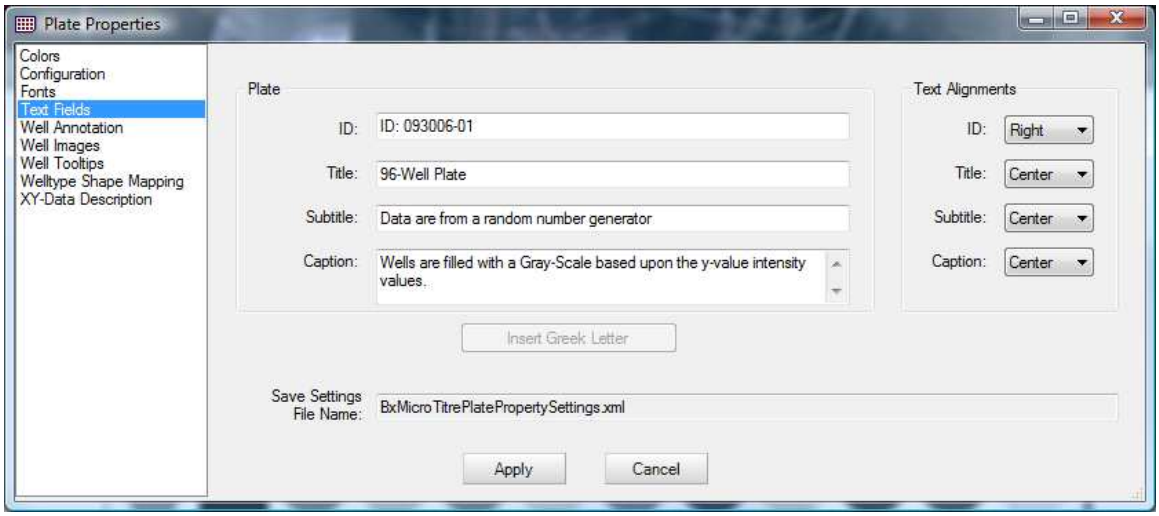


Figure 24: Form for entering text field data.

The 'Insert Greek Letter' button displays a form (Figure 25 and Figure 26) that can be used to select a Greek Letter that can be inserted into one of the Text Fields. In Figure 25 the Greek Letter Sigma is highlighted in Gray, which happens as the Mouse Cursor is moved over a letter, and the OK button is not enabled indicating that it is not selected. In Figure 26 the Greek Letter Sigma is highlighted in Orange and the OK button is enabled indicating that Sigma (σ) is selected. Clicking on the OK button will close the form, save the selected letter in a GreekLetters Property (GreekLetterSelected) and it will also copy it to the clipboard. If a TextBox is selected prior to clicking the 'Insert Greek Letter' button then the selected letter will automatically be inserted; otherwise, a notification message box will be displayed informing the user that after a TextBox is selected, the letter can be pasted using the Ctrl+V key combination.

Programmatically (Refer to Section 7) a Greek Letter can be obtained by instantiating the object and then using its methods and enums such as the following code snippet illustrates:

```
// Instantiate GreekLetters object
BxComponents.GreekLetters bxGL = new
              BxComponents.GreekLetters();

// Get the Greek Letter mu as a string
string mu =
    bxGL.GreekLetter(BxComponents.GreekLetters.GreekSmallLetters.μ);

// Get a description as: <Greek Letter name> <hex code>
// For example: Mu 0x03BC
string Description =
  Enum.GetName(typeof(BxComponents.GreekLetters.GreekSmallLetter),
    BxComponents.GreekLetters.GreekSmallLetter.Mu)
    + "  0x"
    + (((int) BxComponents.GreekLetters.GreekSmallLetter.Mu)
      .ToString("X"))
      .PadLeft(4,'0');
```

Figure 25: Greek Letter Selector Form showing an unselected highlighted letter.

Figure 26: Greek Letter Selector Form showing a selected letter in orange.

6.4.5 Well Annotation

This form is used to specify the column to use for Well Annotation. In the XYDataDescription form the X and Y values can be selected, but their selection can be overridden in this form. The when both X and Y values are unselected the ComboBox is enabled and used to select the annotation column from the list of all string and number columns in the plate DataTable. When a number column is selected a Number Format specification data entry form is displayed.

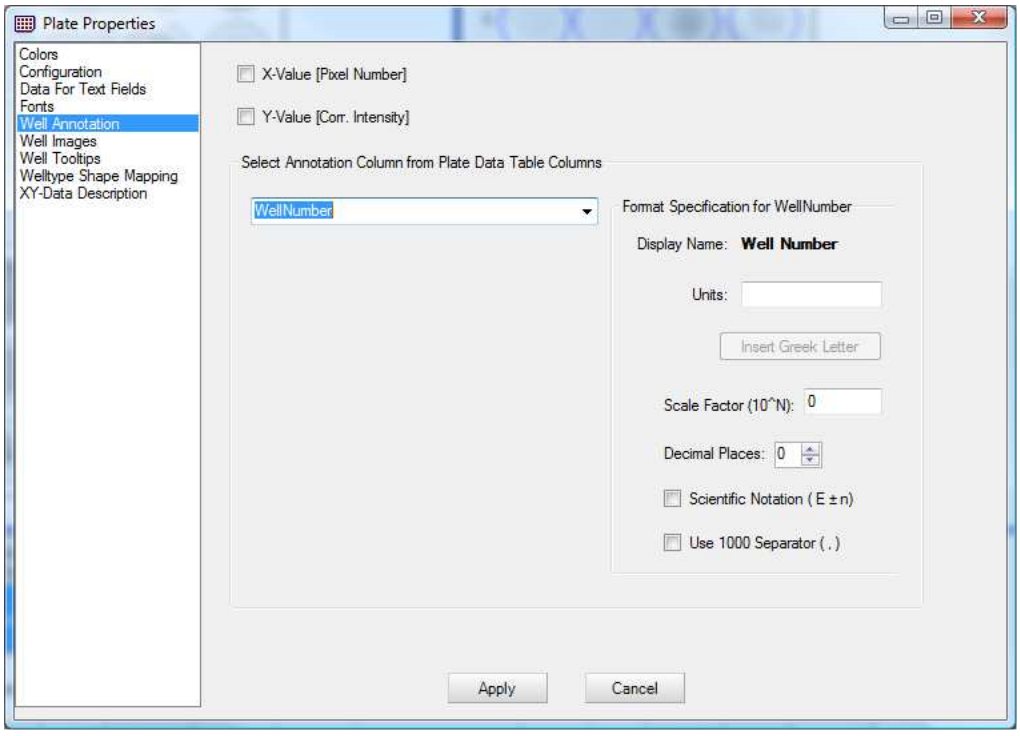


Figure 27: Form for specifying Well Annotation content.

6.4.6 Well Images

This form is used to specify whether well columns are to be filled with an image from the specified image column or not. In addition an option is provided that enable scaling of the image to the well size if the image is smaller than the well. When the image is larger than the well, it is automatically scaled to the well size.

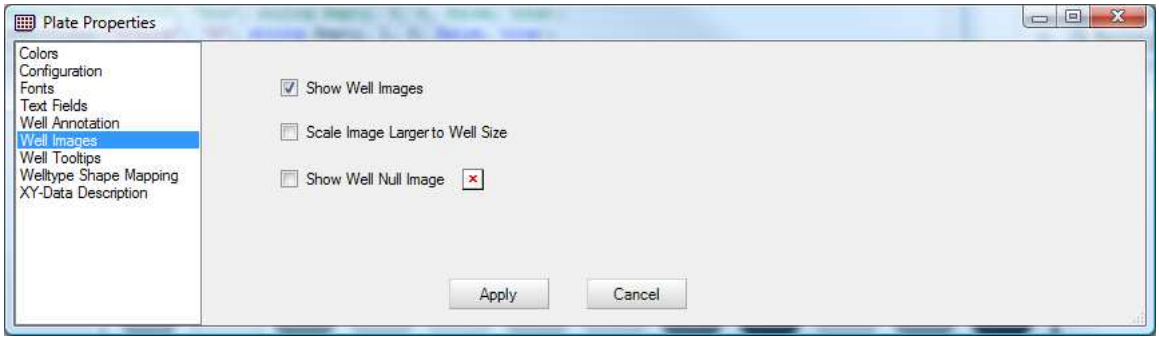


Figure 28: Form for specifying Well Image column and scaling.

6.4.7 Well Tooltips

Tooltips are small windows containing descriptive text that appear when the mouse cursor is moved over a well. This form has options for showing or not showing tooltips and for selecting the content of the tooltip. The appearance of the tooltip will be shown in the framed yellow rectangular area as each option is selected or de-selected.

Item	Description
Location	The row-column index value for the well is shown in parentheses.
Type	The designation for the type of well such as Sample, Control, Standard, Tag or Empty will be shown.
ID	The ID assigned to the well content will be displayed.
Brief Description of Well Content	The minimum plate DataTable provides a field for entering a brief description of the well and if this option is selected, that description will be displayed.
X-Value (…)	The value contained in the minimum plate DataTable for the X Value will be displayed. Also, in parentheses will also be the name for the X Value. In the example it is 'Concentration'.
Y-Value (…)	The value contained in the minimum plate DataTable for the Y Value will be

	displayed. Also, in parentheses will also be the name for the Y Value. In the example it is 'Absorption'.
Other Plate Data Table Columns	Checking this option activates the Check Listbox for selecting additional columns in the Plate DataTable that are to be used displayed in the Tooltip. If the selected column is a number column then a Number Format Window will be displayed for setting the format specification.
Omit Column Headings	This flag is used to remove the name of the column heading from the tooltip, that is, only the value is displayed.

Table 37: Description of Well Tooltip selectable components.

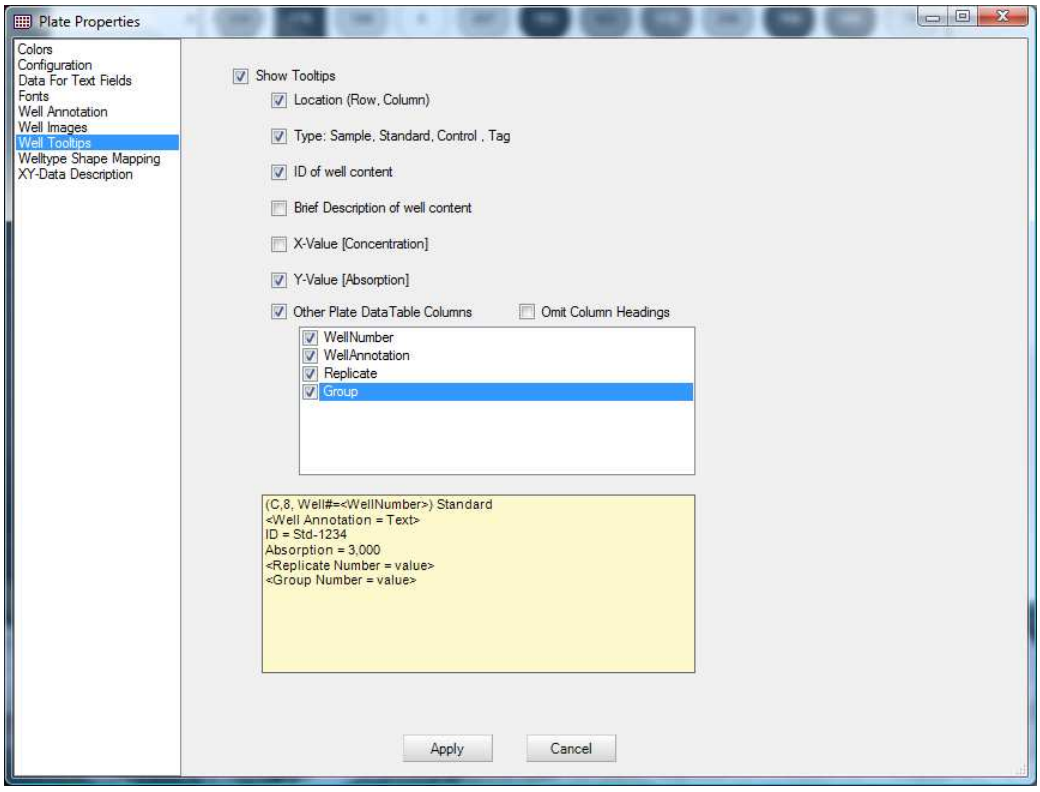


Figure 29: Form for specifying well tooltip content.

6.4.8 Well Type to Well Shape Mapping

The ability to quickly ascertain the type of wells within a plate can be facilitated by assigning shapes to well content. The Well Type to Well Shape Mapping form shown in Figure 30 can be used to make the desired associations. The combo box contains a list of all available shapes and as a shape is selected its image is shown to the left of combo box and to the left of the image what it is going to represent. For example, a Control is being represented by a hexagon and a sample by a circle. Each shape will be filled with a non-empty or empty color and with the selected fill style (Refer to Figure 20 for the color selection form).

This form is also used to associate user defined names for well types. In the ListBox in the upper right of the form, is a list of all unique well type names that were found in the plate data table. The selected user defined name can be drag-dropped onto the TextBox of the type of well it is to be used for. Similarly selected system names contained in the ListBox in the lower right of the form can be drag-dropped onto the well type TextBoxes.

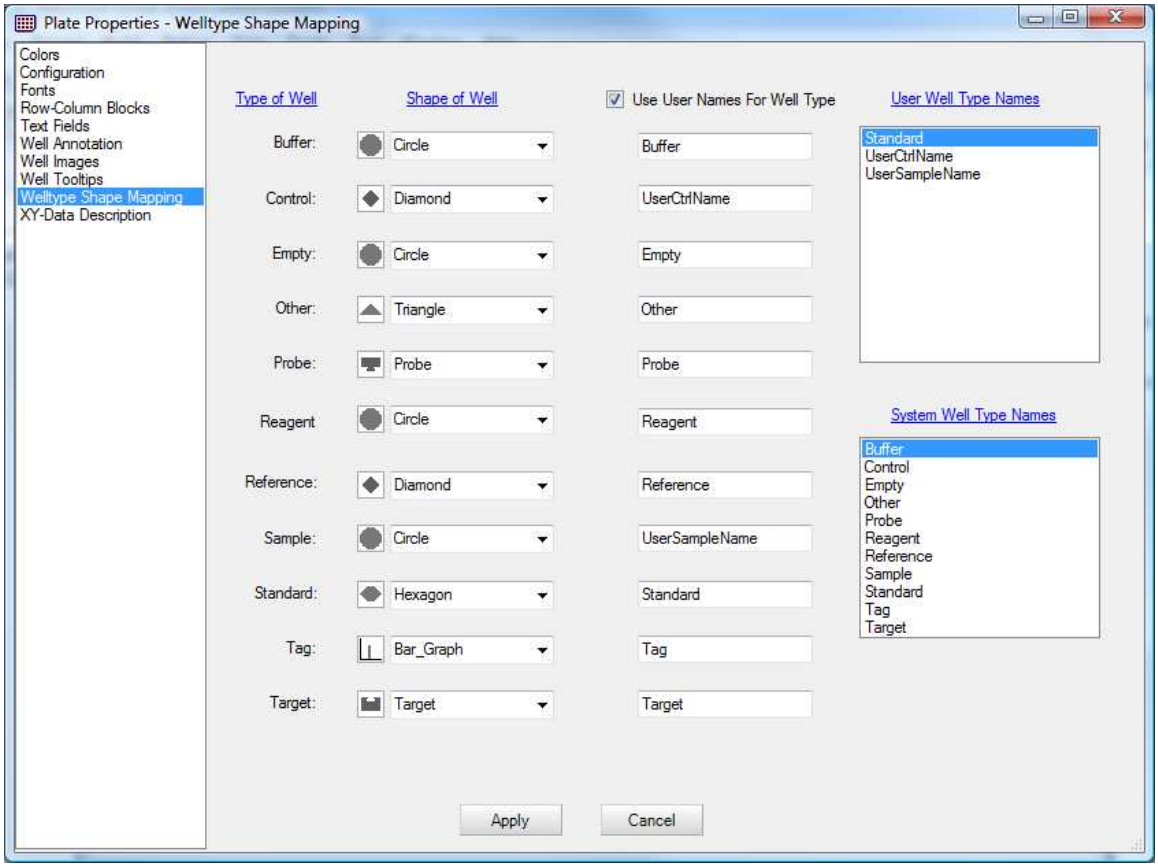


Figure 30: Form for mapping type of well content to a well shape.

6.4.9 XY-Data Description

This form is used to provide Data Descriptions and set format specifiers for the X and Y values. The names and units for the X and Y values can be entered. In addition, if Greek letters are required, then clicking on the 'Insert Greek Letter' will display a form for selecting the Greek letter to insert.

Sometimes the numbers that were provided should be scaled by factors of ten. A power of ten Scale-Factor can be entered for the X and Y values by entering positive or negative integer power value.

The X and Y values can have different display formats. The format can be either scientific notation or standard number format. The number of decimal places to display can be specified and in standard number format mode a thousand separator can be used.

The Gray scale radio buttons are used to specify whether the X-value or the Y-value will be used for generating the gray scale images for wells when the Well Fill Style is Gray-Scale.

The Well Annotation Check Boxes are for specifying that X-Values and/or Y-Values, with their respective display formats, are to be used when annotating a well.

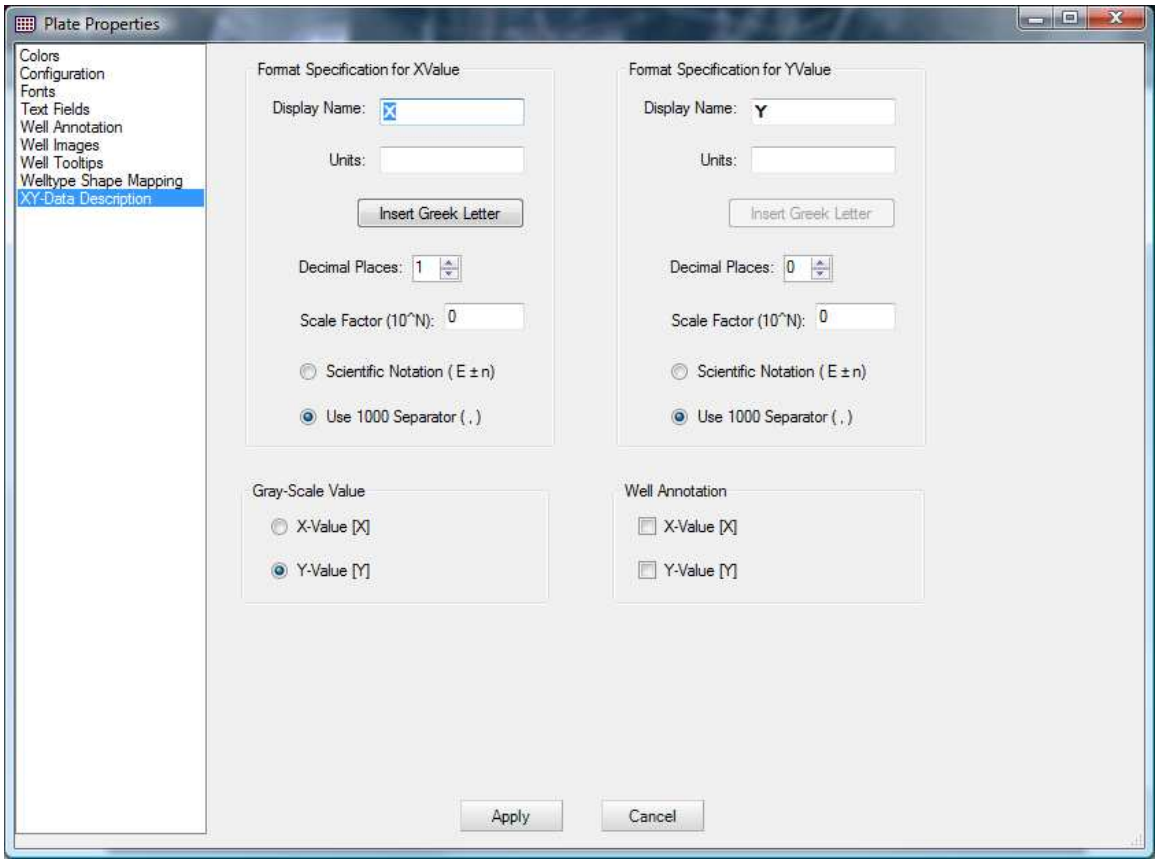


Figure 31: Form for XY-Data Description information.

6.4.10 Row-Column Blocks

This form is used to define blocks of rows and columns for partitioning a plate or matrix of wells into regions or blocks. A block is rendered with increased spacing between neighboring columns and rows and optional gridlines. The initial purpose was to facilitate representation of fraction collectors, but it could also be used for added graphical clarity or emphasis.

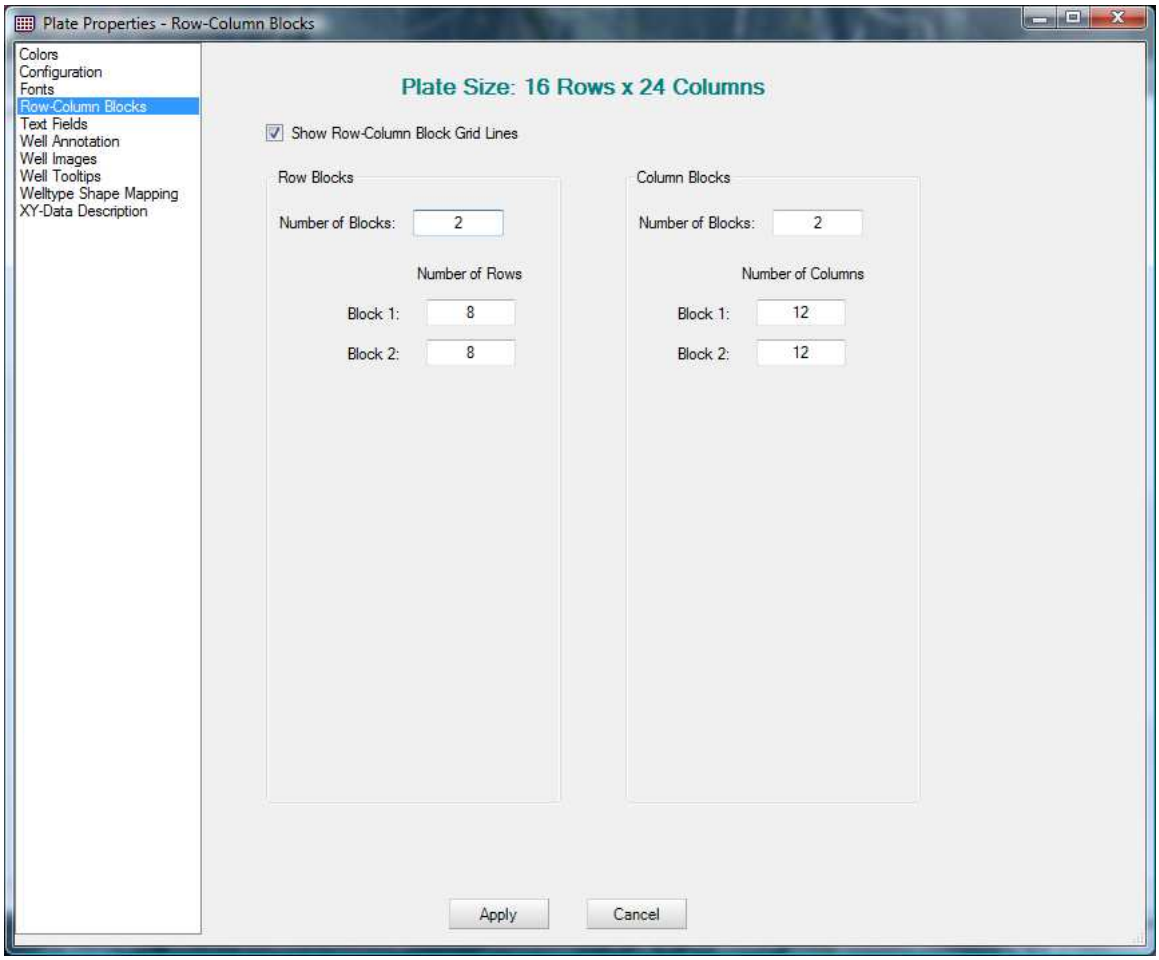


Figure 32: Row-Column Block Specification Form

Figure 33 shows a 384-well plate rendered with four regions defined by two blocks of 12 columns and two blocks of 8 rows. The optional grid lines are drawn between the blocks for emphasis.

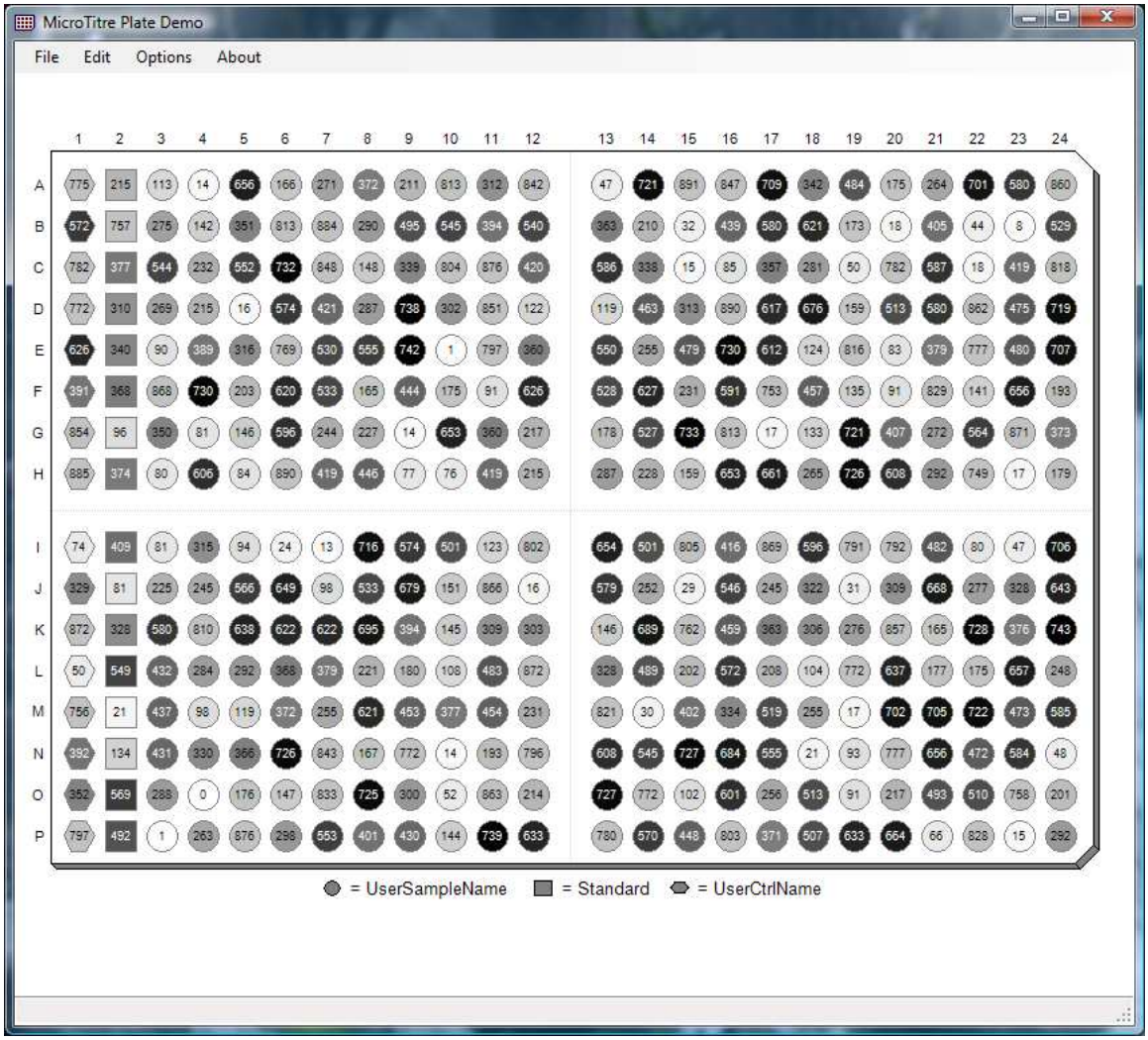


Figure 33: Example of dividing 384-well plate into four (4) blocks.

Figure 34 shows a 96-well plate containing Standards, Controls and Samples with three blocks consisting of 1 column for the Standards, 1 column for the Controls and 10 columns for the Samples.

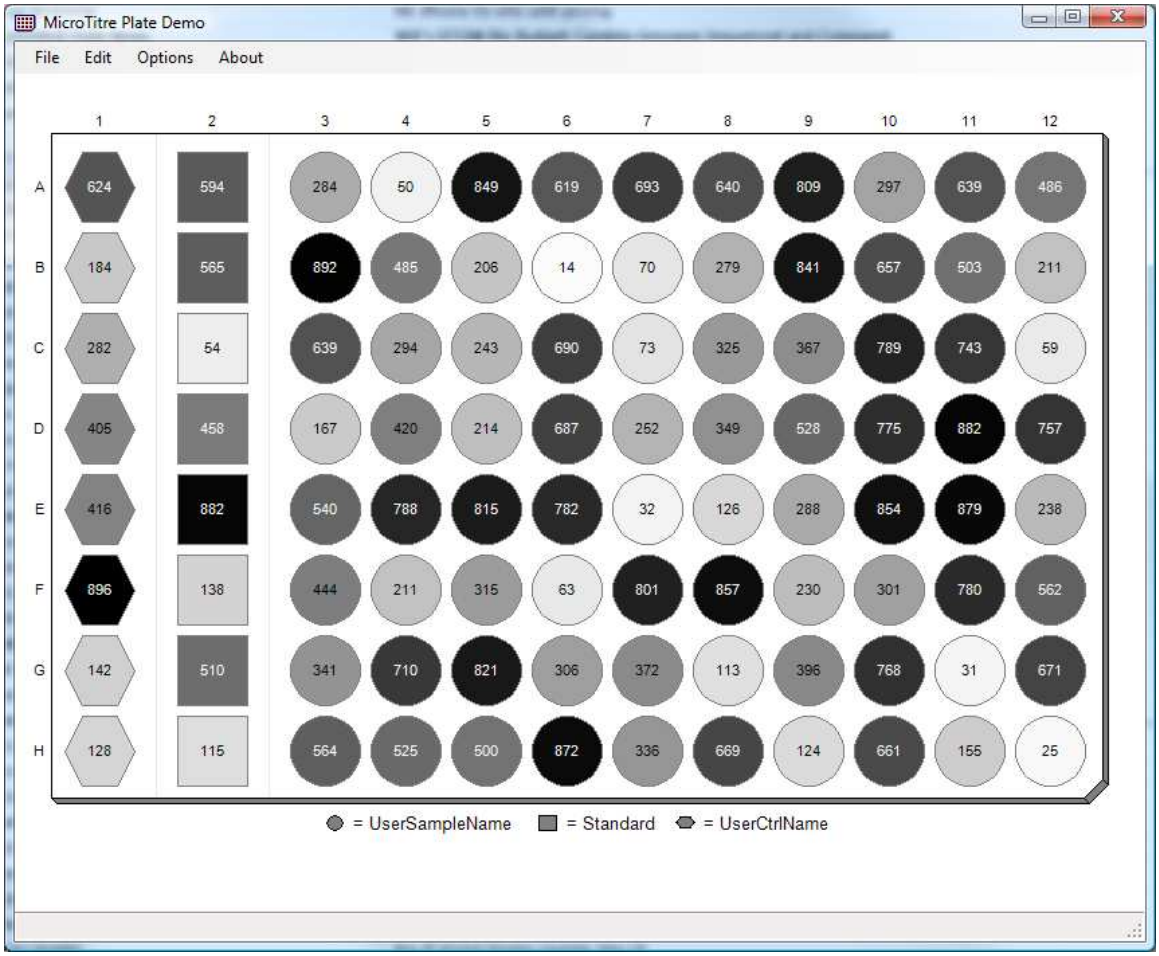


Figure 34: Example showing blocks containing Standards, Controls and Samples

Figure 35 shows a 12 x 12 matrix with two blocks of columns containing 6 wells or fraction collecting tubes each.

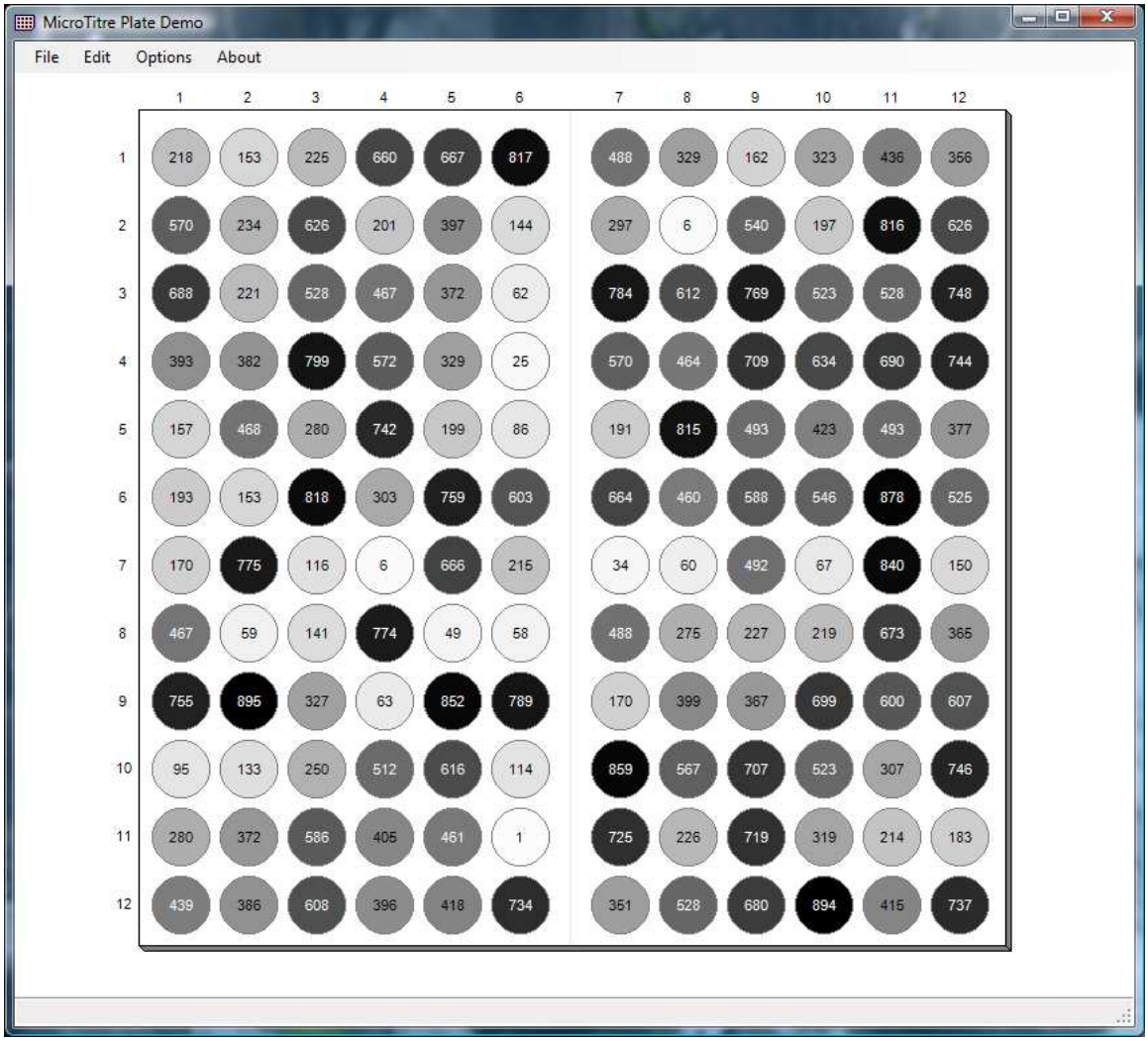


Figure 35: Example of a 12 x 12 matrix of wells divided into two blocks of columns.

7 Greek Letters

Incorporated within BxReporter are enum declarations and public methods for obtaining Capital Greek letters `(ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ)` or small Greek letters `(αβγδεζηθικλμνξοπρςστυφχψω)`.

There are four enum declarations where each member has a value that corresponds to its Unicode numeric value:

GreekCapitalLetter	Names for Greek letters are enum members
GreekCapitalLetters	Capital Greek letters are enum members
GreekSmallLetter	Names for Greek letters are enum members
GreekSmallLetters	Small Greek letters are enum members

The public method GreekLetter() is overloaded to accept an enum member from any of the four enum declarations. Also, the name of the Greek Letter can be obtained by using the enums that have Names for Greek letters and converting to a string.

```
BxComponents.GreekLetters bxGL = new BxComponents.GreekLetters();

string GreekLetter =
      BxComponents.GreekLetters.GreekSmallLetter.Beta.ToString().ToLowe
      r()
              + " = " +
      bxGL.GreekLetter(BxComponents.GreekLetters.GreekSmallLetters.β);
```

Output:

beta = β

There is also an overloaded method that accepts an integer Unicode value for a Greek letter which returns the corresponding Greek Letter as a string. An empty string is returned if the integer is not for a Greek Letter. This method could be used to return all of the letters as follows where Alpha is the first letter and Omega is the last letter which corresponds to the Unicode numbering.

```
string GreekLetters = string.Empty;

for (int i = (int) BxComponents.GreekLetters.GreekCapitalLetter.Alpha;
    i <= (int) BxComponents.GreekLetters.GreekCapitalLetter.Omega; i++)
    GreekLetters += bxGL.GreekLetter(i);

GreekLetters += " - ";

for (int i = (int) BxComponents.GreekLetters.GreekSmallLetters.α;
    i <= (int) BxComponents.GreekLetters.GreekSmallLetters.ω; i++)
    GreekLetters += bxGL.GreekLetter(i);
```

Output:

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ - αβγδεζηθικλμνξοπρςστυφχψω

8 Examples

This section contains example code snippets for using the MicroTitrePlate API.

8.1 96-Well data

```
// Instantiate MicroTitrePlate
BxTools.MicroTitrePlate mtp = new BxTools.MicroTitrePlate();
```



```
// subscribe to events
mtp.mtpMouseDown += new
 MicroTitrePlateMouseDownEventHandler(mtp_mtpMouseDown);
mtp.mtpMouseUp += new
 MicroTitrePlateMouseUpEventHandler(mtp_mtpMouseUp);
mtp.mtpMouseMove += new
 MicroTitrePlateMouseMoveEventHandler(mtp_mtpMouseMove);
mtp.mtpMouseEnter += new
 MicroTitrePlateMouseEnterEventHandler(mtp_mtpMouseEnter);
mtp.mtpMouseLeave += new
 MicroTitrePlateMouseLeaveEventHandler(mtp_mtpMouseLeave);
mtp.Error += new BxTools.ErrorEventHandler(mtp_Error);

// Initialize a few plate properties
mtp.ShapeStandard = MicroTitrePlate.WellShapeStyles.Square;
mtp.ShowWellAnnotation = false;
mtp.ShowCornerCutTopRight = true;
mtp.ShowCornerCutBottomRight = true;
mtp.WellFillStyle = BxTools.MicroTitrePlate.WellFillStyles.Gray_Scale;
mtp.WellColor = Color.Gray;
mtp.WellPrimaryColor = Color.Gray;

// Set number of rows and columns for 96 Wells
mtp.SetPlateRowColumnSize(8, 12);

// Create DataTable with minimum schema
DataTable dtWellData = mtp.GetMicroTitrePlateBaseDataTable();

// Fill dtWellData with data by calling an applications fill method
// which may get data from a data file, input DataGrid, Excel
// spreadsheet or other GUI.
// This method is defined by the developer
FillPlateDataTable();

// Assign the DataTable to Plate A and set
// specific properties for plate A
mtp.PlateDataTable = dtWellData;
mtp.PlateID = "ID: 262";
mtp.PlateTitle = "Pierce BCA Reagent";
mtp.SelectedPlate = BxTools.MicroTitrePlate.PlateSelection.Plate;

mtp.XValueName = "Concentration";
mtp.YValueName = "Absorption";

mtp.PlateAlignment = ContentAlignment.MiddleCenter;

// Add MicroTitrePlate control to the parent's Controls collection.
// That is, the parent that will be used to manage the control
Controls.Add(mtp);

// DockStyles can be used and the plate control can be
// managed using control Size, Location and Anchor properties such as:
//mtp.Dock = DockStyle.None;
//mtp.Location = new Point(50, 50);
//mtp.Size = new Size(300, 300);
```



```
//mtp.Anchor = AnchorStyles.Top | AnchorStyles.Left |
//             AnchorStyles.Right | AnchorStyles.Bottom;

mtp.Dock = DockStyle.Fill;


mtp.Visible = true;
```

8.1.1 Fill DataTable

The following is an example of filling a DataTable for the plate control. The well specific data can be obtained for example from reading in a data file or a spread sheet.

The global DataTable dtWellData is defined by:

```
DataTable dtWellData = mtp.GetMicroTitrePlateBaseDataTable();

private void FillPlateDataTable()
{

// Get data for assigning to fields within each row
GetWellData();

maxCols = 12;
maxRows = 8;
tableRow = -1;
DataRow dr;
for (int nRows=0; nRows<maxRows; nRows++)
for (int nCols=1; nCols<=maxCols; nCols++)
      {
            tableRow++;
            dr = dtWellData.NewRow();
            dr[“WellNumber”] = n; // n ={1,number of wells}
            dr[“WellLocation”] = w; // w = “A1,A2,A3 …”
            dr[“N”] = tableRow;
            dr[“Row”] = nRows+1;
            dr[“Column”] = nCols;
            // Specify type of well content (Sample, Standard,
            // Control, Tag, …)
            dr[“Type”] =
                  mtp.GetWellTypeEnumName
                  (BxTools.MicroTitrePlate.WellTypes.Sample);
            dr[“ID”] = <ID of Well content>;
            dr[“Description”] = <Description of Well content>;
            dr[“XValue”] = <x.dddd x-decimal value>;
            dr[“YValue”] = <y.dddd y-decimal value>;
            // Specify annotation, if any, that can be displayed
            // inside of well.
            dr[“WellAnnotation”] = <Brief description of well content>;
            dr[“Exclude”] = true or false;
            dr[“Empty”] = true or false;
            dr[“Replicate”] = NNN; // replicate number
```



```
            dr["Group"] = MMM; // group number

            dtWellData.Rows.Add(dr);

        }
}
```

8.1.2 Clone Plate

A plate may need to be cloned when a set of plates needs to be rendered with the same fundamental property settings but displaying different data. In addition there may be a need to clone a plate in order to change a few property settings such as removing the Caption, Title, Subtitle or ID or even just changing colors before printing or copying to the clipboard. A Clone() method is available and the following code snippet provides an example:

```
MicroTitrePlate mtpClone = mtp.Clone();
mtpClone.ShowCaption = false;
mtpClone.ShowTitleBorder = false;
mtpClone.ShowSubTitleBorder = false;
mtpClone.PlateSubTitle = "Plate";
mtpClone.WellFillStyle = MicroTitrePlate.WellFillStyles.Gray_Scale;
mtpClone.WellPrimaryColor = Color.Green;

Bitmap bm = new Bitmap(800, 650);
Graphics g = Graphics.FromImage(bm);
g.FillRectangle(Brushes.White, 0, 0, bm.Width, bm.Height);
mtpClone.ScaleToFit = true; // need to override
mtpClone.SelectedPlate = BxTools.MicroTitrePlate.PlateSelection.Plate;
mtpClone.DrawPlate(g, new Point(0, 0),
    new Rectangle(0, 0, bm.Width / 2, bm.Height / 2),
    ContentAlignment.MiddleCenter, true);
```

8.1.3 Print Plate

The following code snippet shows an example of printing the control as it appears in the GUI. That is, it uses the current property settings for each plate to provide a printed image of what is in the GUI. The property values can be overridden; however, if they are, then the original values should be saved for resetting them after printing is complete.

Alternatively, another instance of the User Control can be created and then initialized with the same property settings except for the ones that should be changed.

The first argument is a PrintPageEventArgs that is usually provided by the PrintPage event handler, the second argument is an offset from the top edge of the bounding rectangle that will contain the control usually connected to the MarginBounds for the page and the last argument prtDoc is a Boolean for controlling whether rendering of the control should actually be performed or simulated. Simulation is used, for example, when the total page count for a document is required for labeling each page with a Page N of M annotation, where M is the Total number of pages.

The return value provides the actual height of the control. This value can be used for spacing of additional information below the control.

```
private int printPlate(PrintPageEventArgs ev, float yOffset
     , bool prtDoc)
{

    Rectangle rect = new Rectangle(
        ev.MarginBounds.X, ev.MarginBounds.Y,
        ev.MarginBounds.Width, ev.MarginBounds.Height / 3);

    // Render the control using the printer graphics device
    mtp.DrawPlate(ev.Graphics,
        new Point(ev.MarginBounds.X, ev.MarginBounds.Y + (int)yOffset)
        , rect, ContentAlignment.MiddleCenter, prtDoc);

    // return the actual height of the rendered control
    return mtp.PlateContainerHeight;
}
```

8.1.4 Copy Plate to Clipboard

The following code snippet simply calls the public CopyToClipboard() method provided by the MicroTitrePlate control which copies the currently selected plate.

```
private void Copy()
{
     mtp.CopyToClipboard();
}
```

The following code snippet is used to create a bitmap view containing Plate A, Plate B and Plate A+B and then copying the bitmap to the clipboard. It will use the current property settings for each plate to provide a copy of what is in the GUI; however, they can be overridden. However, if they are to be overridden then the original values should be saved for resetting them after the copy is complete.

```
private void Copy()
{


Bitmap bm = new Bitmap(800, 650); Graphics g = Graphics.FromImage(bm);
```



```
ContentAlignment relativePosition = ContentAlignment.MiddleCenter;

g.FillRectangle(Brushes.White, 0, 0, bm.Width, bm.Height);

mtp.SelectedPlate = BxTools.MicroTitrePlate.PlateSelection.Plate;
mtp.DrawPlate(g, new Point(0, 0), new Rectangle(0, 0, bm.Width / 2,
   bm.Height / 2), relativePosition, true);


Clipboard.SetDataObject(bm, true);

}
```

8.1.5 Get Microtitre Plate Image

The GetMicroTitrePlateImage() method is used to get an Image of the Plate Control. Property values are used to control the content and appearance contained in the image such as whether a Title, SubTitle, ID or Caption are to be rendered as well as the color and fonts. The overloaded method has different types of arguments for specifying the size of the image.

For example the following could be used to get an image:

```
Image plateImage = mtp.GetMicroTitrePlateImage(new Size(800,650));
```

8.2 Customized Array Graphic

This section contains two examples of creating interactive arrays. The first is similar to a telephone keypad and the other is a numeric array of numbers 0 through 9.

8.2.1 Telephone Array

The MicroTitrePlate control can be used to create other types of interactive graphic arrays such as the Telephone Keypad shown in Figure 36.

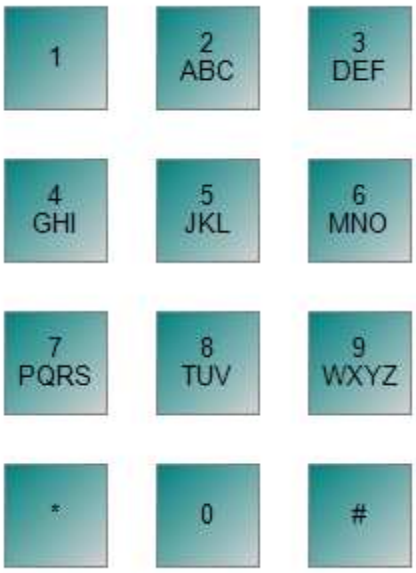


Figure 36: Example of a Telephone Keypad

The Following code snippet was used to create the telephone keypad shown in Figure 36 with the assumptions that 'mtp' represents an instantiated MicroTitrePlate control and has been added to a control collection.

```
private void Set43Table()
{
    int maxCols = 3;
    int maxRows = 4;
    int tableRow = -1;

    String[] wellAnnotation =
        {"1"
        ,"2\r\nABC"
        ,"3\r\nDEF"
        ,"4\r\nGHI"
        ,"5\r\nJKL"
        ,"6\r\nMNO"
        ,"7\r\nPQRS"
        ,"8\r\nTUV"
        ,"9\r\nWXYZ"
        ,"*"
        ,"0"
        ,"#"
        };

    DataRow dr;
    DataTable dtWells = mtp.GetMicroTitrePlateBaseDataTable();
    for (int nRows = 1; nRows <= maxRows; nRows++)
        for (int nCols=1; nCols<=maxCols; nCols++)
            {
                dr = dtWells.NewRow();

                dr["N"] = ++tableRow;
                dr["Row"] = nRows;
                dr["Column"] = nCols;
                dr["Type"] =  mtp.GetWellTypeEnumName
                    (BxTools.MicroTitrePlate.WellTypes.Sample);
                dr["ID"] = tableRow.ToString();
```



```
                dr["Description"] = string.Empty;
                dr["XValue"] = 0;
                dr["YValue"] = 0;
                dr["WellAnnotation"] = wellAnnotation[tableRow];
                dr["Exclude"] = false;
                dr["Empty"] = false;
                dr["Replicate"] = 0; // group number

                dtWells.Rows.Add(dr);
            }


        mtp.SetPlateRowColumnSize(maxRows, maxCols);
        mtp.WellColorPlate = Color.Teal;
        mtp.ShowTitle = false;
        mtp.ShowCaption = false;
        mtp.ShowPlateID = false;
        mtp.ShowSubTitle = false;
        mtp.ShowPlateContainerBorder = false;
        mtp.ShowPlateBorder = false;
        mtp.ShowWellAnnotation = true;
        mtp.ShowRowColLabels = false;
        mtp.WellFont = new Font("Arial", 10.0f);
    // change well shape to squares
        mtp.ShapeSample = MicroTitrePlate.WellShapeStyles.Square;

        mtp.SelectedPlate = MicroTitrePlate.PlateSelection.Plate;
        mtp.PlateDataTable = dtWells;
        mtp.Refresh();


}
```

8.2.2 Simple Numeric Array

A simple numeric array can be created as shown in Figure 37 by creating a 4 x 3 array and hiding wells at locations [4,1] and [4,3].

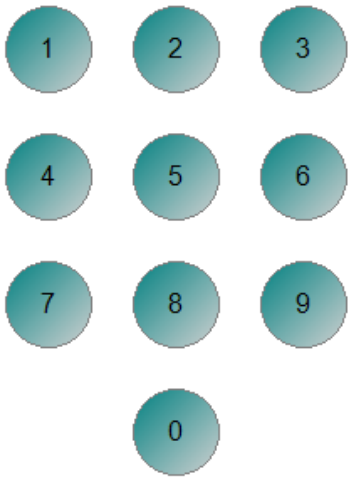


Figure 37: Custom Numeric Array with Hidden Wells

The numeric array in Figure 37 was created with the following code snippet. Note that the mtp.HideWell() method calls come after the DataTable is assigned; otherwise, the

system does not have a way to validate and to internally associate the hidden wells to a display.

```
private void Set43Table()
{
    int maxCols = 3;
    int maxRows = 4;
    int tableRow = -1;

    String[] wellAnnotation =
        {"1"
        ,"2"
        ,"3"
        ,"4"
        ,"5"
        ,"6"
        ,"7"
        ,"8"
        ,"9"
        ,"*"
        ,"0"
        ,"#"
        };

    DataRow dr;
    DataTable dtWells = mtp.GetMicroTitrePlateBaseDataTable();
    for (int nRows = 1; nRows <= maxRows; nRows++)
        for (int nCols=1; nCols<=maxCols; nCols++)
            {
                dr = dtWells.NewRow();

                dr["N"] = ++tableRow;
                dr["Row"] = nRows;
                dr["Column"] = nCols;
                dr["Type"] =  mtp.GetWellTypeEnumName
                    (BxTools.MicroTitrePlate.WellTypes.Sample);
                dr["ID"] = tableRow.ToString();
                dr["Description"] = string.Empty;
                dr["XValue"] = 0;
                dr["YValue"] = 0;
                dr["WellAnnotation"] = wellAnnotation[tableRow];
                dr["Exclude"] = false;
                dr["Empty"] = false;
                dr["Replicate"] = 0; // group number

                dtWells.Rows.Add(dr);
            }


        mtp.SetPlateRowColumnSize(maxRows, maxCols);
        mtp.WellColorPlate = Color.Teal;
        mtp.ShowTitle = false;
        mtp.ShowCaption = false;
        mtp.ShowPlateID = false;
        mtp.ShowSubTitle = false;
        mtp.ShowPlateContainerBorder = false;
        mtp.ShowPlateBorder = false;
        mtp.ShowWellAnnotation = true;
        mtp.ShowRowColLabels = false;
        mtp.WellFont = new Font("Arial", 12.0f);
    // change well shape to squares
        mtp.ShapeSample = MicroTitrePlate.WellShapeStyles.Circle;

        mtp.SelectedPlate = MicroTitrePlate.PlateSelection.Plate;
        mtp.PlateDataTable = dtWells;
   // Hiding of wells must come after well assignment
   // or designate Well Type as WellTypes.Hidden in the DataTable
        mtp.HideWell(4, 1);
        mtp.HideWell(4, 3);
```



```
        mtp.Refresh();

}
```

