NEW EPICS DISPLAY MANAGER IN WPF*

H. Nishimura, C. Timossi, G. Portmann, P. Pace, C. Ikami, M. Beaudrow, E. Williams,
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, U.S.A.

Abstract

We have been developing a new kind of EPICS Display Manager on Windows that we call WPF-DM. It is written in C# using Visual Studio and is based on the new Microsoft XML-based display building technology: Windows Presentation Foundation (WPF). The derivate of XML used by WPF, known as XAML, gives descriptive configuration of the display components. Therefore the GUI construction of an EPICS client display may be created without actual programming. Then, the C# code can be added to extend the functionality of a display. This paper is on the new toolkit WPF-DM with such features.

BACKGROUND

The first-half of the ALS high-level control software upgrade project, [1], will be completed using the following three tools to create EPICS clients.

- EPICS Display Manager (EDM)
- Matlab
- C# /ScaNET

EDM is often used by non-controls experts to write simple EPICS client displays,[2]. Matlab is typically used by physicists to create automated accelerator programs, [3]. C# is optimized to create highly customized programs with rich graphical user interfaces (GUI), [4].

If we focus on the control console programs that operators use interactively to tune up the machine, most of the programs are written in EDM or C#. It is desirable to have both the simplicity of EDM and the flexibility of C#. Instead of giving flexibility to EDM, our effort has been to take the descriptive nature of EDM to WPF/C#.

The new high-level control system uses Windows Vista for the console OS. By using a new standard of graphics programming called the Windows Presentation Foundation (WPF), this task can be done efficiently using tools available on the .NET Framework. The resulting toolkit we call EPICS Display Manager in WPF, or WPF-DM in short.

FEATURES OF WPF-DM

WPF-DM is designed to take full advantages of WPF to create EPICS client programs. It is developed and at runs at the ALS on Windows Vista and takes advantage of the existing power of Visual Studio and especially WPF.

Environment

WPF-DM is written in C# 3.0 on Windows Vista, using the Visual Studio 2008 IDE (using .NET Framework 3.5). Although WPF-DM itself can support both 32-bit and 64-bit modes, it is built in the 32-bit mode to be compatible with the lower layer of the EPICS system at the ALS.

GUI in XAML

Developing GUI in WPF is similar to designing a web page in HTML. Instead of using HTML, WPF uses an XML-based language that is called Extensible Application Markup Language (XAML). Visual Studio comes with a XAML designer for XAML. We can create a GUI either by editing an XAML file, or visually from within the IDE. The XAML code and its visual representation are always synchronized. We usually create a GUI in WPF by switching back and forth between the XAML text editor and the visual design pane, locating WPF components and modifying their properties. WPF-DM provides a set of WPF GUI components that can be used in the same manner.

There is also an XAML designer called Expression Blend that is separate from Visual Studio. It has much better graphics design capabilities than Visual Studio. Typically, it is used to refine and enhance the GUI that was originally created in Visual Studio. Actually, we can work on the same C# project both by using Visual Studio and Expression Blend simultaneously.

Expression Blend does not provide any direct access to the C# source code embedded in the WPF components. Instead, public component properties and routines are all accessible so a user can design, recompile and run the program in Expression Blend. By preparing the WPF-DM Components properly, it becomes possible to develop EPICS clients directly with Expression Blend. In this case, a user does not program in C# but writes the GUI and the actions among components in XAML in a descriptive manner. Once the fundamental properties of the WPF-DM components, such as device names, are assigned in XAML, the design work can be done in a visual manner. Therefore, it has become possible to develop an EPICS client program with a rich GUI without explicitly using any programming language.

Expandability

When extra functionality is necessary, it can be implemented in C#. For example, a user can add a new event handler to a WPF-DM component instantiated on a form. As a WPF-DM client program is a standard WPF application, any WPF programming can be linked to it. This means that WPF-DM is not a closed framework but one of many component libraries that are available on the .NET Framework.

*Work supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231

COMPONENTS OF WPF-DM

WPF-DM is best explained by means of an example, Fig. 1.

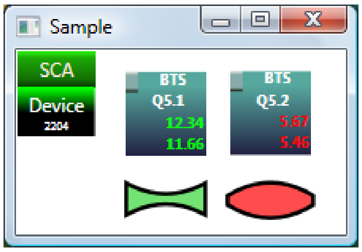


Figure 1: Device and view component example.

SCA Component

WPF-DM SCA Component is a thin wrapper around an SCA.NET component[4]. This component is displayed in the upper left hand corner of Fig. 1. The XAML for it is shown below.

```
<SCA Name="mysca" GroupAccess="False"/>
```

Device Manager Component

The *Device Manager Component* is located just under SCA. This component is an interface to the XML file that containing the device configuration. It's referenced in XAML as follows.

```
<DeviceDB  Name="DevDB"  XML="Devices.xml"/>
```

Device Component

WPF-DM supports various kinds of devices. The Magnet Power Supply (MPS) control is an example of the *Device Components* for the ordinary magnet power supplies. They are shown in Fig.1 as two rectangles each with 4 lines of text shown, and here is one of them defined in the XAML below.

```
<MPSControl  Name="Q5_1"
      DevDB="DevDB"  ScaCon="mysca"
      Caption="BTS;Q5.1"
      DeviceName="BTS_____Q5,1"  />
```

MPSControl gets the device definition from the Device Manager Component "DevDB" by querying the device name BTS_____Q5,1. At runtime, it reads the channel values from the SCA Component "mysca".

View Component

Q5.1 described above is a defocusing quadrupole magnet. Although the MPS Control icon shows the primary values, it is desirable to have a symbolic view. In this case it is a convex lens shape just under the MPS Control. This is the *View Component* that can be attached to a MPS Control (called MagShape). It is defined in XAML as shown below.

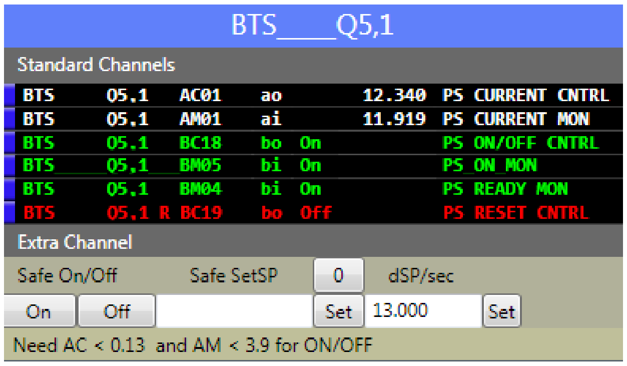
```
<MagShape Controller="Q5_2" MagType="QF" />
```

TV Paddle Component

This is another example of a Device Component. It is for beam profile monitors that use scintillation plate paddles and TV cameras. The TV Paddle Manager Component coordinates multiple TV Paddle Components that control the individual hardware paddles. There are six of them in Fig. 5

Knob-List Component

The Device and View Components are primarily used to display channel values. To control the settings, WPD-DM provides different components. One example is the Knob-List Component in Fig. 3. It is associated with the mouse-click event of a Device or a View Component and controls that device.



BTS____Q5,1

Standard Channels

```
BTS     Q5.1   AC01   ao          12.340  PS CURRENT CNTRL
BTS     Q5.1   AM01   ai          11.919  PS CURRENT MON
BTS     Q5.1   BC18   bo  On              PS ON/OFF CNTRL
BTS     Q5.1   BM05   bi  On              PS ON MON
BTS     Q5.1   BM04   bi  On              PS READY MON
BTS     Q5.1 R BC19   bo  Off             PS RESET CNTRL
```

Extra Channel

Safe On/Off   Safe SetSP   0   dSP/sec

On   Off   Set   13.000   Set

Need AC < 0.13 and AM < 3.9 for ON/OFF

Figure 2: Device and View Component example.

This component communicates with the Knob Panel program (Fig.3) that appears at the bottom of the desktop and makes use of the external USB rotary knobs, [5].

Figure 3: Knob panel .

GUI DESIGN

This simple example was made inside Visual Studio. After adding more components in XAML, Expression Blend can be used to improve the look of the graphics. Fig.4 shows a part of such an example.

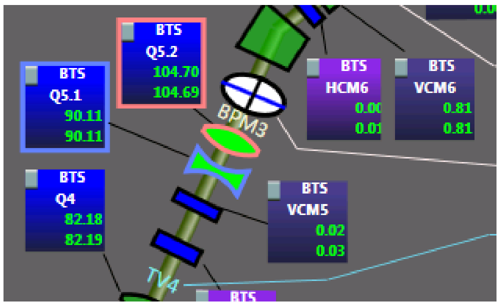
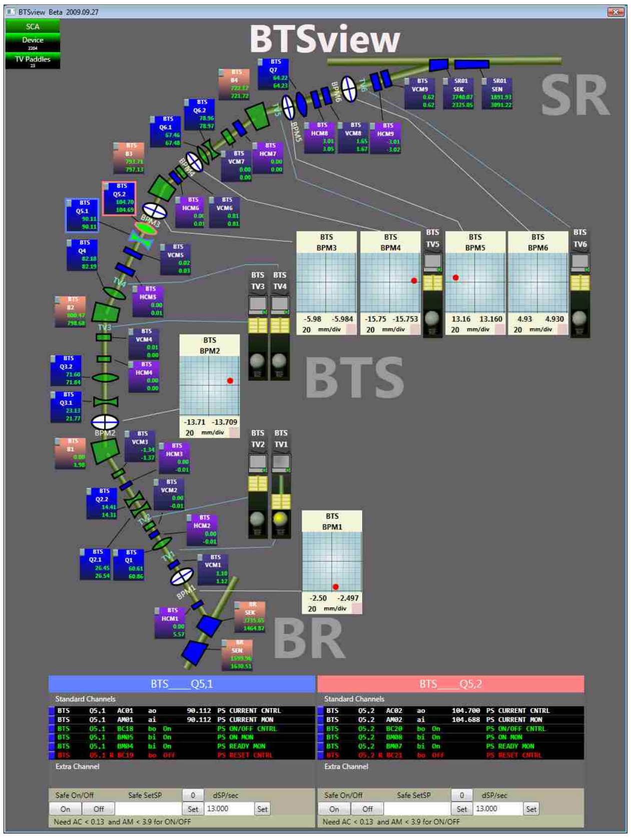


Figure 4. GUI designed by Expression Blend.

Fig. 5 shows an example of an EPICS client program entirely written in XAML by using WPF-DM – two Knob-List Components at the bottom and five TV Paddles near the center.



```
BTS____Q5,1
Standard Channels
BTS      Q5.1    AC01    ao         90.112  PS CURRENT CNTRL
BTS      Q5.1    AM01    ai         90.112  PS CURRENT MON
BTS      Q5.1    BC18    bo  On             PS ON/OFF CNTRL
BTS      Q5.1    BM05    bi  On             PS ON MON
BTS      Q5.1    BM04    bi  On             PS READY MON
BTS      Q5.1 R  BC19    bo  Off            PS RESET CNTRL
Extra Channel
Safe On/Off        Safe SetSP      0     dSP/sec
On   Off                          Set   13.000   Set
Need AC < 0.13  and AM < 3.9 for ON/OFF
```

```
BTS____Q5,2
Standard Channels
BTS      Q5.2    AC02    ao        104.700  PS CURRENT CNTRL
BTS      Q5.2    AM02    ai        104.688  PS CURRENT MON
BTS      Q5.2    BC20    bo  On             PS ON/OFF CNTRL
BTS      Q5.2    BM08    bi  On             PS ON MON
BTS      Q5.2    BM07    bi  On             PS READY MON
BTS      Q5.2 R  BC21    bo  Off            PS RESET CNTRL
Extra Channel
Safe On/Off        Safe SetSP      0     dSP/sec
On   Off                          Set   13.000   Set
Need AC < 0.13  and AM < 3.9 for ON/OFF
```

Figure 5: WPF-DM program entirely written in XAML.

This kind of EPICS client programs can be developed in Expression Blend entirely in XAML without touching any C# code by non-controls experts. If additional functionality is required, a controls expert can add it in C#. This flexibility improves software development productivity significantly.

CONCLUSION

At the ALS, WPF-DM has become the primary tool to create interactive EPICS client programs and serves a complementary role to applications developed using EPICS-EDM and Matlab.

AKNOWLEDGEMENT

We appreciate the support and encouragement from D. Robin, A. Biocca and R. Bloemhard. We also thank W. Byrne and D. Richardson for their constructive comments and test driving.

REFERENCES

[1] H. Nishimura et al, Submitted to PAC'09.
[2] J. Sinclair, http://ics-web.sns.ornl.gov /kasemir/ train_2006/ 1_4_EdmTraining.pdf
[3] J. Corbett, G. Portmann, A. Terebilo, PAC'03, 2369
G. Portmann, J. Corbett and A. Terebilo, PAC'05, 4009.
http://www-ssrl.slac.stanford.edu/at/
[4] H. Nishimura and C. Timossi, PCaPAC'05
H. Nishimura and C. Timossi, PCaPAC'06, 37
C. Timossi and H. Nishimura, PCaPAC'06, 56
C. Timossi and H. Nishimura, PCaPAC'08, 24
H. Nishimura, C. Timossi, G. Portmann, M. Urashka, C. Ikami and M. Beaudrow, PCaPAC'08, 122
[5] Griffin PowerMate USB Multimedia Controller http://www.griffintechnology.com