WPF BASED EPICS SERVER AND ITS APPLICATION IN CSNS

Yuliang Zhang, Ge Lei
Institute of High Energy Physics, Beijing, 100049, China

Abstract

The control system of China Spallation Neutron Source (CSNS) is under construction based on EPICS. The Linac low level RF (LLRF) local control program running on a local control PC uses Windows Presentation Foundation (WPF) as its development tool and uses the C# codes to implement the functionality. The Linac LLRF control system is non-EPICS, so the Linac LLRF local variables can't be accessed directly from EPICS. Therefore we need to port the Linac LLRF local control system to EPICS. This paper presents the WPF base EPICS server and its application in CSNS.

INTRODUCTION

The control system of CSNS is under construction based on EPICS. Local control system for CSNS Linac LLRF system is currently being developed by the LLRF people. Figure 1 shows Linac RF system prototype. RF control system consists two parts: local control program running on a control PC and local RF control hardware, i.e. customized I/O modules and PLCs. The local control program is hosted on an IPC running Windows Xp and it communicates with local RF control modules and local control PLCS vi 100Base-T Ethernet [1]. Linac LLRF local control system is a non-EPICS system and it can't be directly accessed from EPICS software. In order to access the Linac LLRF local control system via EPICS and to facilitate the operation and maintenance, the Linac LLRF control system must be ported to EPICS.

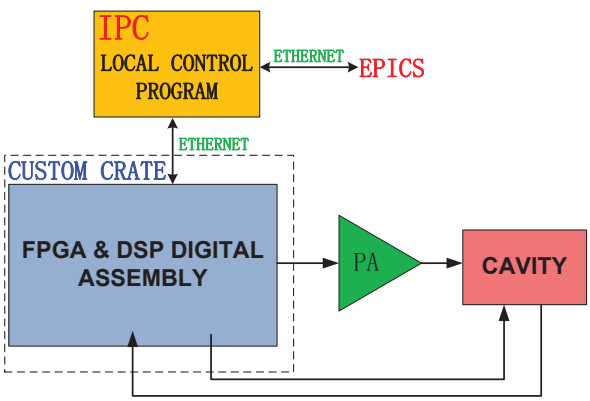

Figure 1: Linac RF system prototype block diagram.

We have considered two ways of porting Linac LLRF control system to EPICS. One is running an soft IOC independent of LLRF local control system, the other way is embedding an EPICS server to the WPF based local control program. For the first method, we need to develop the EPICS drivers for communication between the soft IOC and the LLRF local hardware via Ethernet, which will take a lot of work and the existing C# codes can't be reused. For second method, the work we need to do is embedding a C# EPICS server to WPF based local control program and associate the local variables to CA Records, and also we can reuse the existing C# codes.

MICROSOFT .NET FRAMEWORK AND WPF

.NET Framework

The .NET Framework [2] is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large number of libraries and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for the .NET Framework execute in a software environment, known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework. The .NET Framework's Base Class Library provides user interface, data access , database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with the .NET Framework and other libraries. The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

Windows Presentation Foundation

Windows Presentation Foundation is a computer software graphical subsystem for rendering user interfaces in Windows-based applications [3]. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Rather than relying on the older GDI subsystem, WPF utilizes DirectX. WPF attempts to provide a consistent programming model for building applications and provides a separation between the user interface and the business logic. It resembles similar XML-oriented object models, such as those implemented in XUL and SVG.

WPF employs XAML, an XML-based language, to define and link various UI elements. WPF applications can also be deployed as standalone desktop programs, or hosted as an embedded object in a website. WPF aims to unify a number of common user interface elements, such as 2D/3D rendering, fixed and adaptive documents, typography, vector, graphics, runtime animation, and pre-rendered media. These elements can then be linked and manipulated based on various events, user

interactions, and data bindings.

EPICS C# LIBRARY AND CASHARPSERVER

EPICS C# library was developed at the Paul Scherrer Institut. It is offering a complete EPICS environment containing CaSharpServer library [4] and EpicsClient library [5]. The Library is fully independent and implemented to offer a maximum of portability. It has very clean and easy interface and has a high performance (connects and receives first monitor value for 32‘000 channels in ~16 seconds). In the latest version, it splits to Epicsclient library and EPICS server library, i.e. CaSharpServer library.

The CaSharpServer handles everything from TCP & UDP listening to C# interface based on cosylab’s reverse engineered protocol specification. It is a standalone library and does not require any EPICS modules installed. It can run on .NET and MONO.

The CaSharpServer allows you to publish C# variables to the EPICS Network. The published variables are encapsulated in CARecords and will behave like they would be real EPICS records, so you can use all EPICS compatible tools on those records. It supports Monitors, Get, Put and is compatible with the C-Channel Access Library. The CaSharpServer library version 1.0.0 supports numeric, string and array type record, and on this basis we added the enumerated type record, and also we changed UDP port in class CABeacon to 5064 to compatible with the standard C-CA beacon UDP port. It is very easy to create an EPICS server using CaSharpServer library, the following codes is a simple example to create a C# EPICS server:

```
static void Main(string[] args)
{
    //create the server
    server = new CAServer(
    IPAddress.Parse("127.0.0.1"), 5064, 5064);
    //create a new string type record
    stringRecord= server.CreateRecord<
    CAStringRecord>("STRING:RECORD");
    //set scan frequency is 10Hz
    stringRecord.Scan = ScanAlgorithm.HZ10;
    //add event handler to the string record
    stringRecord.PrepareRecord += new EventHandler(
    stringRecord_PrepareRecor);
     …….
    }
```

PORTING CSNS LINAC LLRF CONTROL SYSTEM TO EPICS

CSNS Linac LLRF control system adopt WPF as the develop tool for the local control program running on local IPC, and uses C# to implement the functionality. We considered embed a C# EPICS server to the WPF based local control program to port Linac LLRF control system to EPICS. In this way, the C# EPICS server and local control program are connected seamlessly, the local variable can easily associated with CARecords resides in C# EPICS server. Figure 2 show the sketch diagram.

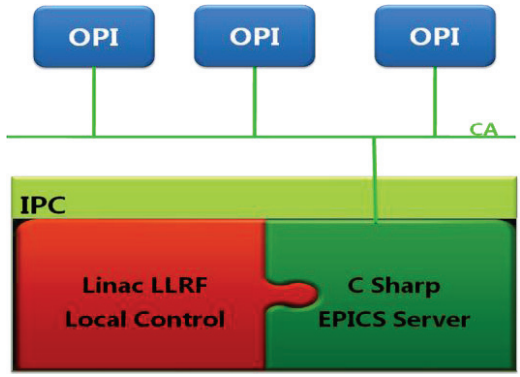

Figure 2: Embed C# EPICS server into LLRF control block diagram

In order to associate local C# variables with corresponding CARecords, one need to provide some extra C# codes to set local C# variables to CARecords or get value to local C# variable.

For example, if setting the value of a WPF TextBox widget to a CARecord named ‘stringRecod’ will result the C# server posts a value change ‘MonitorMask’, and EPICS clients can notice this value changing. It is just one line codes as below:

```
stringRecord.Value = textBox1.Text;
```

It is a bit more complex to update the widget’s value according to the CARecord’s value. The following codes show the detail:

```
stringRecord.PropertySet += new EventHandler<
PropertyDelegateEventArgs>(stringRecord_PropertySe
t);
void stringRecord_PropertySet(object sender,
PropertyDelegateEventArgs e)
{
    this.Dispatcher.BeginInvoke
    ((Action)(()=> { this.textBox1.Text =
    e.NewValue.ToString();
    }));
}
```

Still for this stringRecord, first we have to add an event handler to this record, second we need to implement the event handler function. This event handler function is like C-Callback function, which is a type-safe function pointer，it is known as delegate that can be used to implement callbacks. In event handler function ‘stringRecord_PropertySet’, it provide a no parameter delegate ‘Action’ to process the message, in this example assign stringRecod’s new value to TextBox.

CONCLUSION

The WPF base EPICS server has been developed and tested for the CSNS Linac LLRF control system in order to port LLRF local control system to EPICS. The preliminary work we have done for the Linac LLRF local control system prototype shows that this method works

ISBN 978-3-95450-122-9

well. We still need to do more tests and work in the next few months.

ACKNOWLEDGMENT

The author would like to thank Bertrand Alain Gregor from PSI for his support and helpful discussions.

REFERENCES

[1] Jian Li et al., "CSNS Linac RF System Design and R&D Progress," Proceedings of Linear Accelerator Conference LINAC2010, Tsukuba, Japan, THP046, p.863; http://www.JACoW.org.
[2] http://en.wikipedia.org/wiki/.NET_Framework.
[3] http://en.wikipedia.org/wiki/WPF.
[4] http://gfa-it.web.psi.ch/epicsSharp/index.php.
[5] Christoph Seiler, "Channel Access Library in C#," http://isacwserv.triumf.ca/epics09/meeting.pl?p=agenda.