DEVELOPMENT OF EPICS CHANNEL ACCESS EMBEDDED ACTIVEX COMPONENTS FOR GUI DEVELOPMENT

A. Roy[#], R. B. Bhole, S. Pal, VECC, Kolkata, India

Abstract

The paper describes the integration of Experimental Physics & Industrial Control System (EPICS) Channel Access (CA) protocol and Microsoft ActiveX technology towards developing a generalize operator interface (OPI) building facility for Windows platform. EPICS is used as the development architecture of the control system in Superconducting Cyclotron (SCC). Considering the operators' familiarity and compatibility with third party software, it was decided to use MS-Windows platform at operator interface level in SCC during commission. Microsoft Visual Basic (VB) is used on trial basis as OPI building platform to incorporate user specific features e.g. file system access for data storage and analysis, user authentication at OPI level etc. A set of EPICS Channel Access embedded ActiveX components is developed to ease the programming complexity and reduce developmental time of the OPI for Windows platform. OPIs, developed using these components and containing hundreds of process parameters, are being used reliably over a considerable period of time.

INTRODUCTION

The K_{bend}=520 Superconducting Cyclotron, under commissioning activity at the centre, is expected accelerate heavy ion beams to energy up to 80 MeV/A for fully stripped light heavy ions and about 10 MeV/A for heavy ions. The Microsoft Windows XP was considered as the best suited platform at operation console level during the commissioning phase due to its compatibility with third party software, commonly used Microsoft tools and operators' familiarity. EPICS [1], a standard open-source dual layer software tool for designing distributed control system, is adopted to implement the in-house supervisory control software in SCC. The lower layer of EPICS i.e. Input Output Controllers (IOC), are realised mostly on Linux platform of various flavours to minimise compatibility issues and maximise operational reliability against malwares. There are standard EPICS tools e.g. EDM, MEDM etc. for developing OPIs in general for Linux platform. But porting these X windows based OPIs to windows platform require third party X server e.g. Exceed, Xming etc with expertise during installation. The users, ranging from operators to physicists to system personnel, have the requirements e.g. data archiving into local file for offline / online analysis, authentication based service, integrity with third party system etc. during commissioning phase. As the overall control system was not grown up to its fullest and there are no standard facilities/methods available in the standard OPI tools to meet those custom requirements. Hence a methodology was required to be devised to meet specific requirements of this heterogeneous system. Developing OPIs for Windows platform integrated with standard Windows facilities, meeting user requirements, was the best option. The development of OPIs for Windows platform involved selection of developmental tools among the popular ones e.g. Microsoft Visual C or Visual Basic or National Instrument's LabVIEW, and Borland's Delphi and C++ Builder etc. The CA functionalities were to be incorporated for data access by integrating the CA library with developmental tool. This method, however, required us to understand each development tool's requirements for accessing C language function calls and maintaining this extra layer of code [2]. This results into longer developmental time and larger coding with associated complexity, efforts for testing and debugging for individual OPI.

Microsoft ActiveX technology is chosen to creating reusable, platform-independent, distributed, object-oriented binary software components with encapsulated CA functionalities. Microsoft Visual Basic (VB) is chosen as the OPI development platform considering its object oriented structure, rich GUI library and less complicated coding style. A number of different options currently exist for building CA clients on the Microsoft Windows platform [3]. These include Easy Channel Access (EZCA) [3], ActiveX CA, Simple CA (SCA) [2], Java CA (JCA), CA Java (CAJ) [4], and calling native code in CA via C++. EZCA is chosen to implement CA functionalities. Several CA ActiveX components, commonly used in OPI, are developed and are being used to develop OPIs in considerably reduced time frame.

EASY CHANNEL ACCESS (EZCA[†])

The EPICS channel access API was designed to implement a high performance network protocol including such features as data and connection call-backs, event notifications and smart aggregation of data requests [2]. EZCA is designed to provide an easy to use interface to CA [3]. The library composed of around twenty five API functions supporting CA functionalities e.g. data access (process value, precision, control limits, graphical limits etc.), error handling, grouping, monitors, and tuning. The data types supported by the library are byte, string, short, long, float and double. As VB is selected as developmental tool, EZCA is chosen due to its readily available VB interface supporting, not full but, necessary and sufficient CA functionalities. Although the tuning parameters of EZCA i.e. timeout and retry-count, are provided for improving reliability, the default parameter values are found to be suitable for our purpose.

#r_ani@vecc.gov.in

†Thanks to Nicholas T. Karonis, ANL for EZCA support.

ACTIVEX TECHNOLOGY

ActiveX control has a number of advantages for Microsoft Windows based systems. Among other well developed tools for building them, we use VB in VECC. These components are registered with the OS. This registration tells a client application, in a standardized manner, where they are and what features they have and which makes *late binding* possible. [2]. Due to late biding feature, once an ActiveX component is selected from the list of registered components, all methods and features of the component are readily available to the developer during coding. These components can be installed / uninstalled using standard Windows facility for adding and removing programs.

Since the ActiveX components make calls across process which may take more time compared to library function call. Hence the performance issues using ActiveX components may not be suitable for some time critical applications. In our case, mostly the OPIs are developed for systems e.g. Vacuum, LCW plant, trim coil cooling system, trim coil interlock monitoring etc. involving PC based soft IOC with minimum scan period of 100 msec. Hence the OPI response time for monitoring satisfies the purpose. There are control parameters e.g. set points, binary operations (ON/OFF or OPEN/CLOSE type) which require time criticality of the order of 100 msec. Thus the performance of ActiveX control is found to be satisfactory.

COMPONENT LIBRARY

After surveying the types of control components commonly used in Graphical User Interfaces (GUIs), it is found that the components can be divided into six categories comprising of display of textual values, display of status using colours & images, display of alarms in text, set-point modifier, command issuing button and trending w.r.t. time. Therefore six ActiveX components are developed. The CA interface is minimised to a property called, *pvName*, for attaching an EPICS *process variable* to the component. A function called, *init*, is required to be called only once per component to initialise CA connection management, while loading the GUI for execution. As these components are developed using MS Windows components, hence other properties e.g. *visibility, colour, font, border style, tool tip text* etc. are incorporated by default. A detailed description of the components with embedded features is described below.

CA Text

The *CA Text* control component is used to display a channel's value textually. There are properties to specify *HiLimit*, *LoLimit* and corresponding colours for specifying alarm conditions. The display value can be formatted either in *scientific* or in *general* format with *precision* and *engineering unit* specified by user.

CA Alarm

The *CA Alarm* component is used to display colour coded text information corresponding to *HiLimit* and *LoLimit* of a channel specified by user. It is developed to replicate the conventional alarm window used in control panels.

CA Image

The *CA Image* component is used to display the status (ON/OFF or OPEN/CLOSE) of field components e.g. valve, pump etc. with colour coded industry standard symbols. Each display state is represented by an image file. These may be bitmap, jpeg, png etc. The standard Windows Mouse actions are incorporated in this component for allowing user to take action.

CA Set-point

The *CA Set-point* component is used for controlling numeric value of a channel in textual format. This is required for modifying set-points in control system.

CA Button

The *CA Button* component is used for controlling state of a binary channel. This is required for ON/OFF or OPEN/CLOSE type operation of field components e.g. vales, pumps etc.

CA Trend

The *CA Trend* component is used for trending of process variable. This component has features e.g. *Auto scale, time span etc.*

SOME EXAMPLES

The ActiveX components, developed above, are used extensively for building GUIs for various subsystems of SCC, RTC and other projects. These GUIs are being used for a considerable period of time and have earned wide acceptance among the users. Some of the GUIs are shown in Fig. 1, 2, & 3 below.

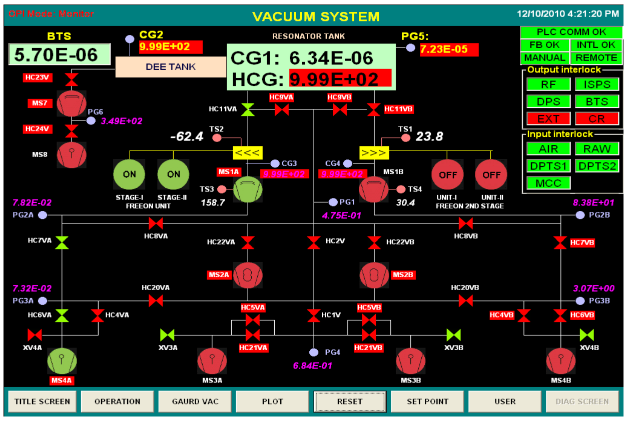

Figure 1: GUI for RTC Vacuum control system.

CONCLUSION

The use of the CA enabled ActiveX components results into more user friendly, colour & textually more rich

ISBN 978-3-95450-124-3

GUIs with less development time. As the GUIs are developed using VB environment, hence other Windows feature e.g. user authentication, message box, file handling for data storage, multiple windows can be incorporated without any extra effort.

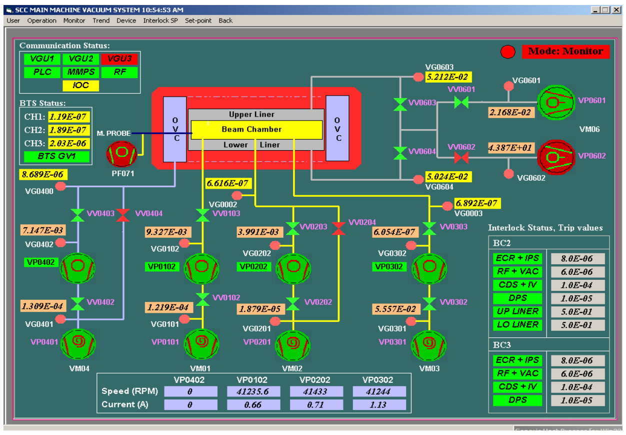

Figure 2: GUI for SCC Vacuum control system.

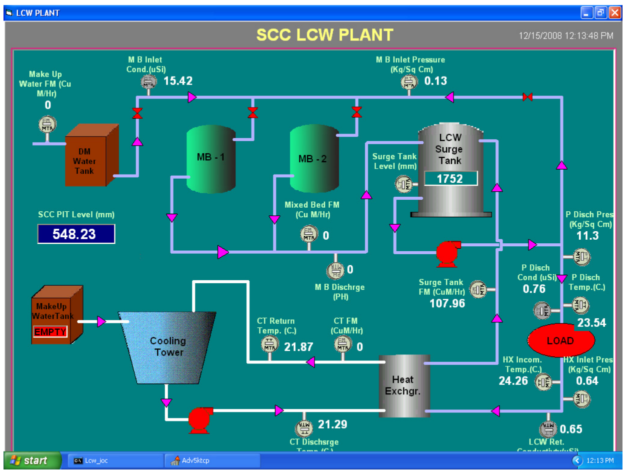

Figure 3: GUI for SCC LCW plat monitoring system.

Of course, an ActiveX control is not installable on other operating systems. ActiveX has now been subsumed by Microsoft .NET which defines a new object model. However, an ActiveX control can still be used directly in the .NET environment [2].

REFERENCES

[1] L.R. Dalesio, et al., “The Experimental Physics and Industrial Control System Architecture”, ICALEPCS, Berlin, Germany, 1993.

[2] C. Timossi, et al., “Experience with ActiveX Control for Simple Channel Access”, 2002.

[3] EZCA Primer, http://www.aps.anl.gov /epics /EpicsDocumentation /ExtensionsManuals /EzCa /EzcaPrimer.html

[4] M. Sekoranja, “Native Java Implementation of Channel Access for EPICS”, ICALEPCS, Geneva, October, 2005.

ISBN 978-3-95450-124-3