**FLEXUSI – INTERFACE BUILDER FOR COMPUTER BASED ACCELERATOR MONITORING AND CONTROL SYSTEM**

V.G.Kurakin, A.V.Koltsov, Lebedev Physical Institute, Moscow, Russia  
P.V.Kurakin, Keldysh Institute of Applied Mathematics, Moscow, Russia

**Abstract**

We have developed computer code for any desired graphics user interface designing for monitoring and control system at the executable level. This means that operator can build up measurement console consisting of virtual devices before or even during real experiment without recompiling source file. Such functionality results in number of advantages comparing with traditional programming. First of all any risk disappears to introduce bug into source code. Another important thing is the fact the both program developers and operator staff do not interface in developing ultimate product (measurement console). Thus, small team without detailed project can design even very complicated monitoring and control system. For the reason mentioned below, approach suggested is especially helpful for large complexes to be monitored and control, accelerator being among them.

The program code consists of several modules, responsible for data acquisition, control and representation. Borland C++ Builder technologies based on VCL-library have been used to achieve functionality mentioned at executable level. To date we have around dozens components for console designing. This are signal waveform digitizers, DAC and ADC, RF synthesizes, arbitrary form generator, timer and so on. Network components make it possible to realize distributed computing and control in real time scale, while interpreter component allows to pre-process data.

The program may run under Windows 9x and Windows 2000. Multi platform application (Windows and Linux), real time data pre-processing based on modern standard technologies (MathML) are among the next steps to do. The software developed is used already at Lebedev Physical Institute Radiation Complex and planned to be the basis for monitoring and control system for Lebedev Physical institute electron synchrotron under development now.

**INTRODUCTION**

User interface, that is the way of physical data representation in computer controlled experiment, depends on many items. Traditionally, experimenter determines how it has to look like in any particular case, and programmer develops appropriate computer code according to given specifications. Perhaps, this is the most direct way but not the best one of the experiment environment developing. We have developed computer code that allows building up any desired interface at program run time stage. The advantages of similar approach are the following. First, any corrections of base computer code are excluded while building new interface. This in turn excludes the possibility introduce any bug into code, thus providing the program reliability. Second, the entire work of experiment monitoring designing can be divided into several relatively independent parts, and the work of measurement environment designing might be committed to expert in physics rather than to programmer. The next is less evident but perhaps the most important for large experimental facilities automation. Project coordinator does not need detailed project specification at the project start step to manage the whole project – the details may be formulated at the final stage, and these details perhaps are not important for programmers staff at all. All of this is directly applied to accelerators and accelerator based facilities.

**THE TARGET SETTING**

To achieve functionality mentioned above one has to have an access to methods and properties of the graphical interface objects at program run time stage. In general, it is not the case for standard compilers of any object oriented programming languages, in C++ in particular. In oriented languages one may have such access in source code, but this functionality disappears after code compiling. We had chosen Borland C++ Builder as programming environment for several reasons. Besides the well known feature of rapid application development this programming tool possesses also the remarkable properties to move properties and methods accessibility to executable file stage. We want to remind that C++ Builder environment allows to create desired user interface using drag-and-drop method and then filling out appropriate forms in so called object inspector. Visual Component Library (VCL) is used to fill out form with necessary components that compose monitoring and control interface. It is worth to point to the fact that VCL components are real objects rather than abstract representation of classes used to facilitate and automate programmer routine work in writing program code. What is important for us, VCL supports the next functionality during run time. It allows enumerate all properties, which has been declared as __published. Also there are functions to read values of these properties and to change them as well. So called component persistence is supported. This means that component properties set can be saved to a stream (file or computer memory) and an object can be created on base of this description at any time. Abstract class TDesigner is declared in VCL library for visual editing support. In designing mode, all events sent to component, first can be handled by an object of the class inherited from TDesigner. It allows to handle user input and to prevent (in some cases) standard behavior of component as well.
Thus, besides majority of auxiliary tasks inherent to any program code developing two key problems have to be solved to achieve program functionality under discussion. User Interface is not some collection of elements for measured data visualization et al. Only. It may be an interaction between these elements – there are not “bricks” only but “glue” too. In other words, there must be some “lows”, embedded in real components that behave the components interaction, and this is the second task to be solved in addition to the first one just described above.

PROGRAM CODE

Two root object classes form the basis that determines the bases of program kernel. This are VBaseDisplay and VBaseDataModule. The first one correspond to a form (object, that has the window, that may contain displayable elements)and is derived class with the parent one TCustomForm. The second one corresponds to data module and is derived from base class TDataModule. An object of this class has not the window and as a result may contain other not displayable objects only. To provide designer mode at run time phase class VDesigner had been created and Borland C++ Builder TFormDesigner is the parent class of our class.

As has already mentioned necessary components property may be accessible at run time phase due to VCL library functionality described in previous section. Necessary function set is available in Borland C++ Builder environment in order to get as well to change component properties. User interface had been developed to edit component properties, and this is similar to object inspector in C++ Builder environment. This interface becomes visible after switching software to designer-editing mode. We do not go details of appropriate program code, since it is quite usual programming.

We follow usual programming style when standard functionality inherent to real component is determined by parent class, while components are created from derived classes with additional specific properties and methods. Listed below are the base (parent) classes that determine “interactions lows” between real components in user interface, responsible for data acquisition, preprocessing, representation and storing:

- Class VDevice determines interface with objects VStorage and VDeviceGroup. All classes that are responsible for data acquisition and have any interface with objects VStorage and VDeviceGroup have to be derived from this class.

- Class VDeviceGroup determines interface with objects VDevice and VDevice. All classes that are responsible for acquired data storing have to be derived from this class.

- Class VDeviceGroup determines interface of objects VStorage with VDevice in the case, when all devices behave as the group. This means, that only one object in the group initiates a process of data acquisition and storing.

- Class VOutputDevice determines interface of data channels enumerating as well as data themselves. It is derived class from parent one VDevice.

- Class VDACControl represents control component. We include it in base class set in order to have possibility for control object to be notified about data acquisition starting and ending.

- Class VChannelView represents component for data displaying. Component of this class may be data source as well.

- Class VBaseDisplay is the container for the components which are responsible for data displaying and interacting with user.

- Class VBaseDataModule is container for non-displayed components. E.g. component for data reading from physical device is an object of this class.

Components of three last classes mentioned above are notified about data acquisition starting and ending. These are notified about new data incoming as well.

- Class SDataValue is used for data transmitting between components. It contains data themselves as well additional information concerning this data – data identifier, their type and so on.

Data acquisition and processing algorithm looks as follows. Appropriate component is notified (over interrupt initiated by the signal of device controller and subsequent message sending in windows environment) about new data incoming. The component reads data, assigns new unique identifier to this data portion and calls global function that informs all objects of the classes VBaseDisplay and VBaseDataModule of this event. If appropriate component has a connection to this data source it checks if these are new data and then processes them. Data source in turn may have an interface with other data source, and if this takes place it sends appropriate request to this object.

PROGRAM INTERFACES

The program under discussion is realized as usual Windows application with usual functionality. For example, component multiplication may be done over copy and paste procedure. Several data sources are foreseen, two of them having been realized up to now. These are external controller as well network card. Appropriate interface is realized over corresponding component. Network component makes it possible to build distributed monitoring and control system of desired architecture, including hierarchic monitoring and control systems as well as peer one with the same software for any computer in local area network. Suppose, one wants to control accelerating voltage in electron synchrotron at the accelerating time interval as well as at electron beam dropping period. It is obviously that accelerating interval has to be control from the central control room with accelerator staff, while beam control with rf voltage changing during beam dropping period might be realized from the control panel of experimental area by experiment staff. Our software allows to solve this problem easily.
Acquired data can be displayed or retransmitted over network component mentioned already. Data can be preprocessed before their subsequent use. Interpreter component is realized as element of component library. Class TPifagor is responsible for vector data preprocessing in source code. Two basic public methods – Compile() and Calculate() - form interpreter interface. Compile() method gets string that contains mathematical expression for data preprocessing. String format makes it possible to use user library available from text file. String editing in design-editing mode is foreseen, followed by expression compiling before it using for data preprocessing with Calculate() method.

Extendable component library makes it possible to design user interface of any complexity with well known drag-and-drop procedure followed by filling in appropriate forms in object inspector like in Borland C++ Builder IDE (Integrated Developer Environment). Dozens components are available for data acquisition, transmitting, visualization, storing and control. Among these are ADC (analog-to-digit converters) and DAC (digit-to-analog converters), signal digitizers, waveform generator, rf synthesizer corresponding to real devices and many others supporting necessary functionality. Although there are important questions for detailed discussion all these are most likely the subject for user manual or separate article rather than the subject for the problem under discussion. Some details of user interface functionality one can find in our previous paper [1].

The software under discussion runs under MS Windows 95 and all subsequent Windows versions. It is used for computer based measurement and control at Lebedev Physical Institute Radiation Complex [2].

CONCLUSION

We started software development with small financial support and modest specification to reach our physical aim of Free Electron Laser commissioning. Step by step we have gotten result just described. Further program development is scheduled in the frame of the project of the Lebedev Physical Institute Electron Synchrotron automation under the way now [3]. Next steps will include new IDE – most likely it will be Borland C++ Builder X with multi platform (MS Windows and Linux) graphical library. There are plans to use modern programming technologies including MathML and Microsoft COM technologies. We expect that these standard technologies will extend our software functionality. Appropriate hardware of course the other constituent of automation project, but this is out of the paper topic.

ACKNOWLEDGEMENTS

We want to thank our division heads A.N.Lebedev and E.I.Tamm for long period support in project realization. Our colleagues V.P.Busygin and V.M.Alekseev have developed hardware for this project and this collaboration has appeared very fruitful.

REFERENCES

