INTERFACE EXTENSION IOLIB OF CLASS LIBRARY MFC FOR CONTROL APPLICATIONS

Y. Amemiya, Engineering Research Institute, School of Engineering, The University of Tokyo, Tokyo, Japan
Yu. A. Gaponov, Siberian Synchrotron Radiation Center, ISSC SB RAS, Novosibirsk, Russia
K. Ito, Photon Factory, Institute of Material Structure Science, National Laboratory for High Energy Physics, Tsukuba, Japan

Abstract

The hierarchical scheme of C++ object class library IOLib is described and discussed. This library is interface extension of the class library MFC for control applications in Windows 95/NT. The extension consists of three main parts: external device interface, windows interface and programming interface. The external device interface supports a control of CAMAC modules. The windows interface supports different types of windows objects: controlling (switches, buttons), numerical (integer, double, string), graphical (2D). Programming interface supports operations with binary and ASCII text files; multithread operations.

1 INTRODUCTION

The project of Interface Object Library is planned to be the next realization of the class library for object-oriented programming of control applications. Previous versions of the Interface Object Library were written for Windows 95 (using Borland C++ with OWL) and for Solaris 2.4 (using ProCompiler and GnuCC with Motif[1]) [1].

The programming of the experimental control applications is connected with solving several different tasks. It is necessary to create the experimental control device interface to supply the connection and interaction of a computer with different modules needed for building the experiment (step motors, detectors, monitors, temperature or pressure regulators). The display user's interface is necessary to provide input and output of experimental and control information (presentation of the digital and graphical information on the screen, control of the keyboard and mouse for input of the experiment parameters and control of the experiment operation). It is also very convenient to have some experiment procedure interface to provide simple programming of different experimental conditions.

Very often creation of the control application for new hardware forces the programmer to make system programming. In this case one problem is to make the programming extension with flexible and simplified interfaces (graphical, hardware, experimental process). Another problem is the possibility to make system programming (programming in low level language). One may solve these two problems and realize it by using the object oriented programming. From this point of view C++ language is a very convenient, useful programming language [2].

During several last years Windows 95 and Windows NT operating systems became more suitable and more convenient for control programming. There is a mechanism of multitasking (multiprocess and multithread mechanisms). It allows one to execute several parallel tasks to provide the real time mode of operation of the application. Creating the MFC class library stimulates the writing of the easily managed and controlled graphical applications [3]. From the other hand, there is a quite cheap way of creation of electronics for scientific equipment (PC hardware and CAMAC logical devices based, for example, on ALTERA chips). Due to these features, the Windows 95/NT operating systems become quite popular among programmers who are working with the experiment control applications. Nevertheless, simple using of classes from MFC class library is not very convenient for control tasks. The objects of these classes are the objects for general and universal purposes. Taking into account the specialization and complexity of control tasks, the necessity of coding the different kind of communications with both MFC objects and control devices objects, one can conclude that the next extension of MFC class library is actual task for control application programming.

The main goal of this project is to create extension of MFC class library (Interface Object Library) for the writing of experimental control applications using Visual C++.

2 INTERFACE OBJECT LIBRARY

Figure 1 shows a general overview of the Interface Object Library. The library has hierarchical structure. The I level classes define the main objects of the controlling application. All of classes of the display user's interface are constructed with the MFC and Windows API Libraries.

[1] http://www.inp.nsk.su/~gaponov/cpp_programming.html

Classes for the CAMAC access are based on the CAMAC Access Library.

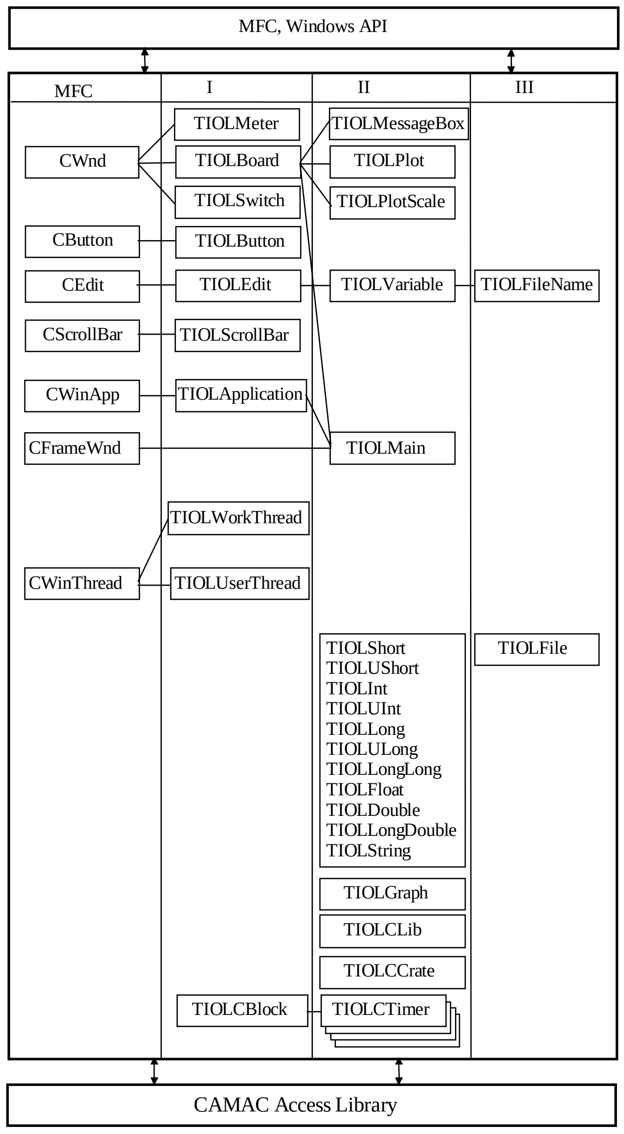

Figure 1: General overview of Interface Object Library with MFC.

2.1 MFC extension

Because of the I classes are primitive ones, for practicable programming, a set of derived classes were provided.

The object of the TIOLMain class is used as an application with main application window. It is analogous to the main() function in C language. After creation of this object the main window of application appears on the screen. Creating the object of the TIOLBoard class as a child object of the main window one may to define (it is logically to use menu bar in main application window) the message map functions for the main menu bar (with sub-menus), defined in the object of CFrameWnd class. All window objects, created later, have to be pointed as a child of the corresponding parent object to be correctly managed.

The different controls are represented by the objects from the TIOLEdit, TIOLButton, TIOLSwitch and TIOLScrollBar classes of the library. The mechanism of message map functions and defined additional functions (for objects of TIOLScrollBar class) is used in the Interface Object Library. When any control is made active the defined message map function is called. The object of the TIOLEdit class may check the type of input character. Defining the check rules one may organize, for example, input of only characters that represent the integer numerals. In this case checking rules are written very simply as a string "++--09" (ASCII codes 43, 45, 48-57). Other characters during the input will not be inputted and will not be reflected on the screen.

The diagnostic indicator is represented by the object of the TIOLMeter class. This is a bar with changeable size of the another color bar inside the main one. This indicator represents some application variable changing within some limits.

A very important class of the library is TIOLVariable. Objects of this class represent different variables of the application. The pointer of the application variable is passed to the object of this class during creation. Also the map function is defined. During the input operations in the editable field, the value of the variable is automatically updated. It is considered to have two ways of inputting the value of variable: without and with a special key to complete the input operation. There is an additional feature of the object compared with the TIOLEdit class that may check the type of the input character. In this case there is possibility to define the valid value range. If the input value does not belong to the defined range, the color of the numeric information and of the editable area border changes indicating the incorrect input. This incorrect value will not be updated in the application. The application value will have the last valid value.

The group of the classes was derived for operations with the window objects like with the different kind of language variables. This group of classes is TIOLInt, ..., TIOLString. Creation of objects of these classes is similar to creation of ones of the language general types. The only difference is that additionally it is necessary to initialize the created object. During the initialization one can define the limits of the object changes, the completion function. The completion function will be called after every successful input to the editable field of the object. The objects of these classes may be used in the arithmetical equations.

In the library the specialized class TIOLFile for the file operations is foreseen. After creation of the object of this class one has to initialize it. It is possible to initialize two different map functions. One is for global checking the experimental situation without the analysis of the structure of the file. Another one is a traditional function that is called after successful analysis of the structure of the file. The file in the Interface Object Library has a heading block and several user's blocks. The heading block consists of the identifier of the file, full date of the creation, comment string and the reserved space for future use. The identifier is used by checking procedure. The user's block consists of the number of records, the format of the record, the string of the comment for this block of data and the column of the data. All user's blocks are defined after initialization by calling the method function of the object AddBlock(...) with the description of the structure block and the pointer to the data that will be used during the read/write operations. Read/write method functions operate with at least one block of the Interface Object Library file. For optimizing the time for file access the structure after successful opening or creating the file is stored in the Interface Object Library. The pointer to the file block is defined as a pointer to the current file block. Read/write operations increment the value of this pointer. For reading or writing the file block one has only to point on the necessary block by using the method function Seek(nBlock) and call then the read/write method function.

For presentation of the one dimensional array the TIOLGraph class was derived. To optimize the quantity of information output the graphical controls are "hidden" in an additional dialog panel that is called by pushing the middle mouse button under the plot area. After creating the object one have to initialize it. Then several arrays may be connected to this object with different color. To update the information of this object one has to use method function Update() after any modifications of the arrays.

For simplifying the experiment programming the TIOLUserThread and TIOLWorkThread classes were inserted into library. The object of TIOLUserThread class is initialized with the pointer to some window object to be connected with it. The virtual function Run() is defined to make structure of the thread: initial part, body part and final part. The body part of code contains the cycle of different cases, handled by an object of CEvent class. Every part of different cases contains (optionally) the parts of code, controlled by objects of CCriticalSection and CSemaphore classes. The object of TIOLWorkThread class is initialized with the pointer to some procedure that has to be carried out during the experiment. Data buffer may be defined for data exchange between different objects of that classes. Synchronization during the operation with the data buffer is carried out automatically. This allows one to use method-functions of the thread objects for data exchange, synchronization of the, for example, CAMAC access.

2.2 CAMAC classes

The main class for the CAMAC access is TIOLCLib. The object of this class allows one to organize the operation with several CAMAC crates, to check it and check its content (CAMAC modules) easily through additional dialog panel from the library. The object of the TIOLCLib class uses objects of the TIOLCCrate class previously defined with the objects of the TIOLCBlock class or of the derived from it.

3 CONCLUSION

This project is planned to be realised under Windows 95/NT operating system using Visual C++ 7.0 with MFC. The main part of Interface Object Library with graphical user's, CAMAC and process interfaces was created in Solaris 2.4 Operating System.

There is some smaller IBM PC realization of this library in Windows 95 (written in Borland C++ 4.0) that has been in operational use for about 4 years. It is managed by specialists with different professional orientation. There were no problems for them to extend and modify this software without the help of authors.

There is possibility to add another device interface with using corresponding device access library. On the example of the OD3Tools application for the controlling the fast X-ray parallax-free position sensitive detector the good real-time response of the application during the actual experiments was achieved [1,4].

REFERENCES

[1] Yu.A. Gaponov, K. Ito, Y. Amemiya, “Object library for a new generation of experiment-controlling applications under the UNIX operating system”, J. Synchrotron Rad. (1998). 5, 593-595.

[2] B. Straustrup, (1991). “The C++ programming language”, Reading, MA: Addison Wesley.

[3] G. Schildt, (1996). “MFC programming from the ground up”, Osborne/McGraw-Hill.

[4] V.M. Aulchenko, et. al., “Fast, parallax-free, one-coordinate X-ray detector OD3”, Nucl. Instrum, and Meth. In Phys. Research (1998). A405, 269-273.