C/C++-based NX Secondary Development Technology and Applications

LianJin DENG

(Jiangnan Mechanical and Electrical Design Research Institute, Guiyang, Guizhou 550009, China)

Abstract: NX is a software further updated and upgraded on UG, NX secondary development technology refers to the existing NX software functions on the basis of writing programs to add, expand the functions that the user wants to achieve, so that the NX software is more in line with the needs of the designer. there are various ways of NX secondary development, such as GRIP, KF, SNAP, NXOpen, etc., and the development of these ways of The application programming interface (API) is licensed by NX software to access the shared C/C++ API and the NX kernel. Among these development methods, NXOpen is the most commonly used secondary development tool, which provides a collection of APIs that enable developers to create customized applications for NX using programming languages such as C/C++, Visual Basic, C#, Java, Python, and so on, in an open architecture. Through NXOpen, users can edit the program to develop functions, create customized menus, build a complete user interface, and ultimately realize the development of human-computer interaction interface and related functions.

Keywords: NX secondary development, NXOpen, C/C++, API, Technical application

I. Introduction to NXOpen C/C++

NXOpen C/C++ is a secondary development tool that combines the foundation of the C++ language with the API functions contained in the C language[1], and it contains the most complete API functions, the most perfect performance, and the most varied, as shown in Table 1, NXOpen C/C++ supports the new version of the UI interface, which is continuously updated by the official, and the development tool adopts Visual Studio, which has greater superiority than other development methods. Compared with other development methods, NXOpen C/C++ contains Open C API interface and Open C++ program interface, developers can check the help document, call the corresponding API function for program writing and system development[2].

Table 1 Comparison of NX secondary development methods

Development Method	Generality	Completeness	UI Interface	Official Update Frequency	Development Tool
GRIP	Good	Incomplete	Old	Not updated	Text compiler
KF	Good	Incomplete	New	Not updated	Text compiler
SNAP	Poor	Incomplete	New	Not updated	Visual Studio
NXOpen C	Good	Incomplete	Old	Not updated	Visual Studio
NXOpen C++	Poor	Complete	New	continuously updated	Visual Studio

Development Method	Generality	Completeness	UI Interface	Official Update Frequency	Development Tool
NXOpen .NET	Poor	Complete	New	continuously updated	Visual Studio
NXOpen Java	Poor	Complete	New	continuously updated	Eclipse
NXOpen Python	Poor	Complete	New	continuously updated	Eclipse

1.1 Introduction to NXOpen C

The main function of NXOpen C's Application Programming Interface is to connect the NX software with the external environment. The API consists of: (1) a complete and large set of functions and subroutines that can be called by the developer, covering almost all the functionality of the NX software; (2) a set of commands that link and run the designer's program; (3) an interactive interface that enables the program to run in the NX software. interactive interface that enables programs to run in the NX software.

Programs of the NX Open C API are able to run in two different environments, internal and external; external programs are stand-alone programs that are able to run independently of the operating system outside of NX, while internal programs can only run within the NX software. Most of the programs of the NX Open C API can run both inside the NX software and outside of the NX software on the operating system, which mainly depends on how the program is linked. Programs that run internally have smaller executable files and are faster to link compared to programs that run externally. Once an internal program is loaded into memory, it remains continuously in the NX session and can be executed without being reloaded when the program is called again, provided that the program has not been uninstalled.

1.2 Introduction to NXOpen C++

NXOpen C++ is an object-oriented programming interface for NX software that provides access to NX objects and functions through a hierarchy of C++ classes. You can use the methods of this C++ class hierarchy to create, read, and modify NX objects and to control NX programs.NXOpen C++ has the following features.

(1) NXOpen C++ is based on C++ programs using standard C++ syntax, which makes the program code more concise and readable.

(2) NXOpen's C++ objects are referenced by pointers to them, rather than using a label, which makes its programming style more natural and expressive.

(3) NXOpen C++ programs report errors through the C++ exception mechanism, which allows error handling to be located in multiple places and avoids explicitly checking for error code returned by a single function call.

(4) NXOpen C++ is strongly typed, which means that any attempt to call a function on an inappropriate NX object will be caught by the C++ compiler and will not result in a runtime error.

(5) NXOpen C++ takes advantage of the encapsulation of C++ programs, making it easier to identify the set of functions associated with a particular type of NX object.

(6) NXOpen C++ provides a framework for defining new object classes using standard C++ inheritance, allowing your derived classes to overlay virtual methods, and allowing users to define NX operations such as updating the meaning of a new object.

(7) NXOpen C++ can be used in conjunction with traditional Open C function calls, allowing objects created in Open C++ to be accessed in Open C and vice versa.

(8) NXOpen C++ allows the creation of context management of NX objects by specifying a proprietary section in each create method.

(9) NXOpen's C++ program unifies the

evaluation and querying of curves and edges, allowing you to write code without having to write special branches for wireframe curves and solid edges.

(10) The NXOpen C++ program makes it easy to edit objects without having to query whether an object is a prototype or an occurrence.

II. NXOpen secondary development environment configuration and platform construction

2.1 Environment Configuration

Before carrying out secondary development, it is necessary to configure the environment of NX software and computer operating system, NX software and Visual Studio software in order to be able to realize the relevant functions of NX software through the relevant program code. This paper is based on the NX10.0 platform, and computer system configuration, add environment variables in two ways. The first in the NX10.0 directory to find UGII/menus/custom_dirs.dat file, and then "UGII_USER_DIR = (development menu directory path)" added to the end of the text; the second set the computer's system variables, the specific steps as follows "Advanced System Settings → Environmental Variables → New Variables", as shown in Fig.1, create UGII_BASE_DIR, UGII_ROOT_DIR and UGII_VENDOR_DIR, which are the system variables for the path of the base path, the path of the root directory and the path of the development menu directory respectively. The following are the system variables for the base path, root directory path, and development menu directory path.

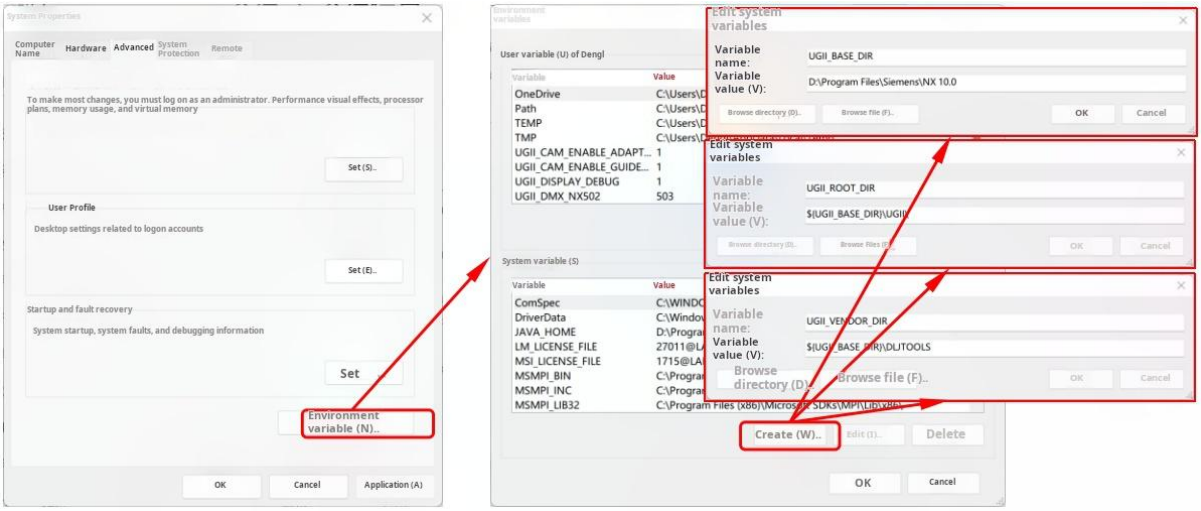

Fig.1 System Environment Configuration

Different versions of NX software need to correspond to different versions of Visual Studio development environment, this article is based on the NX10.0 corresponds to Visual Studio 2012 software. When configuring the development environment of NX and Visual Studio, copy all three folders of VB, VC and VC# in the installation directory NX10.0/UGOPEN/vs_files of NX10.0 software to the installation directory of Visual Studio 2012 software, and restart Visual Studio 2012 software, then when creating a new project, there will be more VB, VC and VC# folders in the list. Restart Visual Studio 2012, when you create a new project, there will be two more options, NX10 NXOpen Wizard and NX10 NXOpen C++ Wizard in the list, select one of them to create a new project, open the properties of the project, and follow the guideline in Fig.2, select C/C++ → General → Additional Containing Directory and set it to the base directory's $(UGII_BASE_DIR)\\ugopen, and select Linker → General → Output Files set to $(UGII_VENDOR_DIR)\application in the development menu directory. in addition to this, the contents of C/C++ → Preprocessor → Preprocessor

Definitions are all replaced with the items _CRT_SECURE_NO_WARNINGS, _SECURE_SCL=0, and USRDLL, and linker → Input → Additional Dependencies are replaced by libugopenint.lib, libufun.lib, libnxopencpp.lib and libnxopenuicpp.lib. Finish the above operation, you can complete the NXOpen secondary development of the development environment configuration.

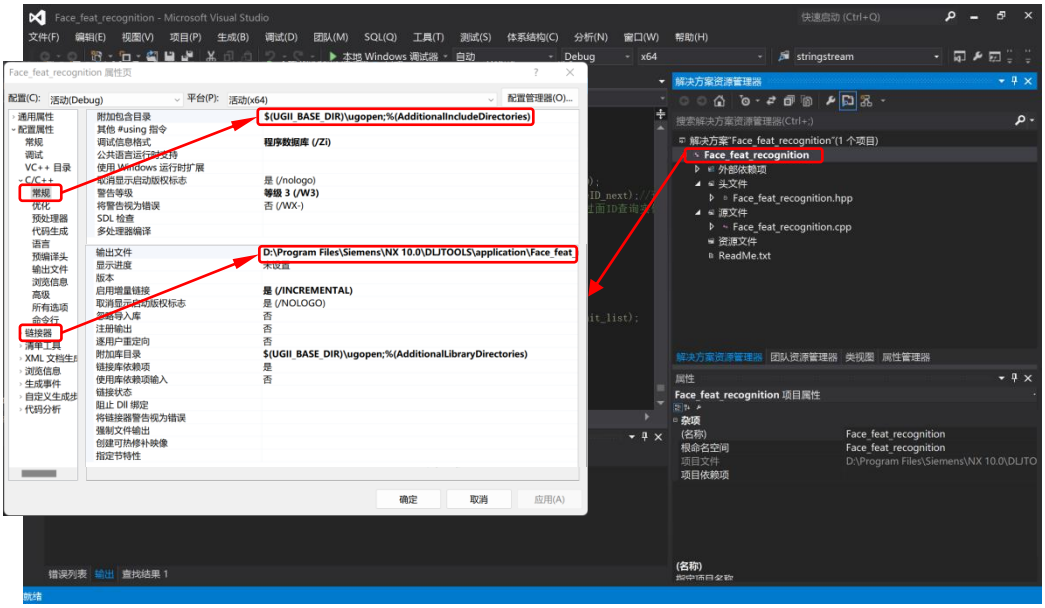

Fig.2 Development Environment Configuration (in Chinese software interface)

2.2 Platform Construction

In order to facilitate the secondary development work, we need to create three sub-folders in the path of the NX secondary development directory, namely, startup, application, and code, which are responsible for storing various necessary resources of the system[3]. The startup sub-files include menu files (.men), toolbars (.rtb), which are mainly used to generate the system's pull-down menus and toolbars. application folder is mainly used to store bitmap (.bmp), dialog box files (.dlx), dynamic link library files (.dll), etc. Bitmap files refer to the icons displayed in NX's operation dialog boxes, Bitmap files refer to the icons and diagrams displayed in the NX operation dialog box, dialog box files refer to the files generated by the developers for designing the UI interface, and dynamic link library files refer to the files generated by debugging and running the program code to directly realize the functions of the code. code folder is a folder for storing the secondary development source code solutions (.sln), C++ program code (.cpp), etc. The files in the development path each performs the relevant functions. The files in the development path each perform the relevant functions and work together to form the human-computer interface of NX and complete the development of product examples, as shown in Fig.3.

(1) MenuScript script code can be written directly through the text document, and its main code is as follows:

① First-level main menu

```
VERSION 120          //Version number
EDIT UG_GATEWAY_MAIN_MENUBAR
                     //Edit Main Menu
AFTER UG_HELP        //Main menu storage location
    CASCADE_BUTTON
CORE_PULLING_MECHANISM
                     //Menu button name
    LABEL “System Name” //Chinese label
END_OF_AFTER         //End main menu editing
MENU CORE_PULLING_MECHANISM
    ...              //Secondary menu editing area
END_OF_MENU          //End secondary menu
```

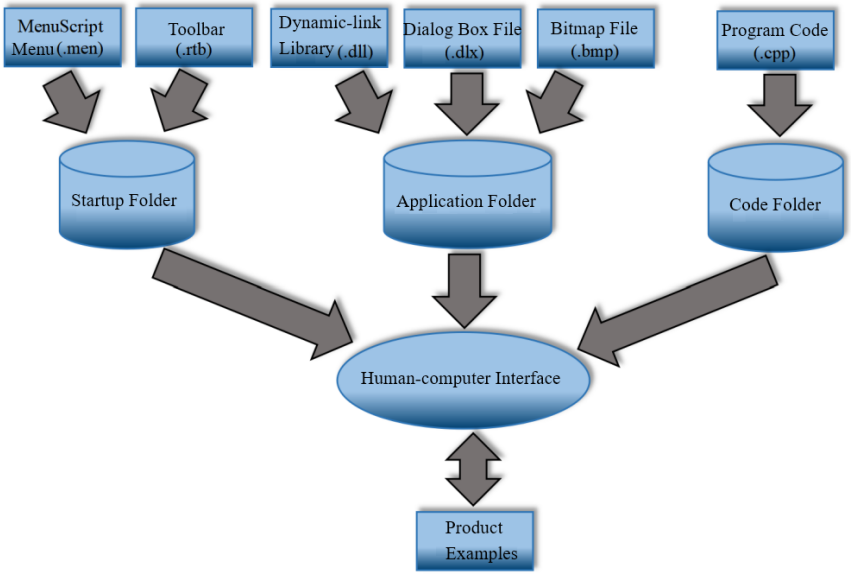

Fig.3 Development Program Workflow

② Secondary Menu

```
BUTTON CASTING_GEOMETRIC_ANALYSIS
                    //Secondary Menu Name
LABEL "Secondary Menu Name"    //Chinese label
BITMAP cae_geometry_revision    //Bitmap Name
ACTIONS Casting_Information_Analysis  //DLL file
```

(2) Toolbar file program code can also be written through the text file, the main code is as follows:

```
TITLE "System Name"          //Title
VERSION 170                  //Version number
BEGIN_GROUP    "Subsystem Name"
                             //Module start
    BUTTON LOGIN_SYSTEM
                             //Introduction of MenuScript
    RIBBON_STYLE LARGE_IMAGE_AND_TEXT
                             //Bitmap and text font size
    GROUP NXOpen_dlj_part.gly//Gallery library files
    COLLAPSED TRUE     //Folding toolbars
END_GROUP                    //End of module
```

The toolbar layout of the human-computer interaction design system for inclined guide pillar side core pulling mechanism developed in this paper is shown in Fig.4, which contains two modules, namely, intelligent feature recognition system and core pulling mechanism part design. The intelligent feature recognition system contains toolbars for molded part analysis, typical feature library, solid feature recognition, molded feature recognition, surface feature recognition, etc. The core pulling mechanism component design module contains design parameter calculation, tilting pin design, slide mechanism design, attribute filling and other auxiliary design toolbars.

Fig.4 Layout of the toolbar of the human-computer interaction design system for inclined guide pillar side core extraction mechanism

III. UI Design

User Interface (UI)[4-5] is the interface designed for user's human-computer interaction, and UI interface design can improve the operation experience and interface appearance. In the previous UI interface design, people often use manual programming to design, but this is not friendly to designers and wastes a lot of time. However, adopting Block UI Styler to design UI interfaces allows users and developers to interactively build

dialog boxes consistent with the NX style, which reduces designers' development time, automatically generates code, and maintains compatibility with MenuScript.

In NX software, you can access the new version of UI interface design by clicking File → Launch → All Application Modules → Block UI Style Editor in the toolbar. The block directory of the new version of UI interface design mainly consists of five categories: Basic, Numeric, Layout, Selection and Special. After selecting the corresponding UI Block type, you can modify the corresponding attributes in the System Dialog Box, and the design result will be instantly displayed in the dialog box of the design. The window of the system dialog box includes two tabs, namely “Dialog Box” and “Code Generation”. The function of “Dialog Box” is to set the layout and properties of the UI interface. The role of “Dialog” is to set the layout and properties of the UI interface, you can set its name, bitmap, display style and so on, and “Code Generation” is to configure the way of generating the code of the dialog box, and the code generated by the UI interface of this system is C++. Fig.5 shows the complete UI interface as designed.

In the dialog box design of UI interface, designers only need to click the UI Block type in the block catalog to edit the properties of the type and add it to the design interface. However, different UI block types correspond to different Classes and different functional implementations. Commonly used UI block types are shown in Table 2, and developers can find the corresponding APIs in the help file according to the classes in the table, and use them to realize getting, setting, and referencing the values of the type.

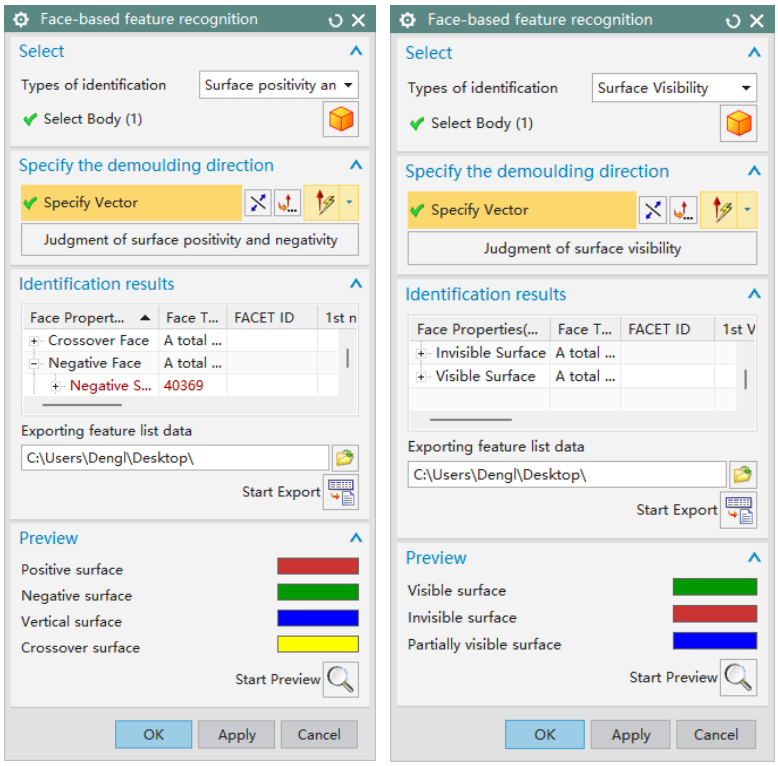

Fig.5 Schematic diagram of UI interface design

Table 2 Common UI Block Information

UI Block Type	Class	Description
Label/Bitmap	BlockStyler::Label	Display text or bitmap
String	BlockStyler::StringBlock	Single line text input box

UI Block Type	Class	Description
Enumeration	BlockStyler::Enumeration	Enumeration box to perform selection operations
Action Button	BlockStyler::Button	Button, click to perform an action
List Box	BlockStyler::ListBox	List box, lists all options
Integer	BlockStyler::IntegerBlock	Integer input box
Double	BlockStyler::DoubleBlock	Double precision number input box
Expression	BlockStyler::ExpressionBlock	Expression input box
Select Object	BlockStyler::SelectObject	Selection of objects
Curve Collector	BlockStyler::CurveCollector	Curve Collector
Face Collector	BlockStyler::FaceCollector	Face Collector
Body Collector	BlockStyler::BodyCollector	Body Collector
Select Feature	BlockStyler::SelectFeature	Select Features
Specify Point	BlockStyler::SpecifyPoint	Specify Points
Specify Vector	BlockStyler::SpecifyVector	Specify Vector
Specify Plane	BlockStyler::SpecifyPlane	Specify plane
Tree List	BlockStyler::Tree	Tree list

For the query and use of the classes of UI design, developers can enter the help document of NX secondary development, find the NXOpen C++ Reference Guide, enter the Classes, find the class of BlockStyler, and then query the use of the relevant classes. The new UI design greatly reduces the workload of the developers and allows them to design a beautiful and very humanized human-computer interface.

IV. Specific Applications of NXOpen C/C++ in the Development Process

4.1 Specific Applications of NXOpen C

The help documentation of NX secondary development provides nearly 92 API categories, which contain more than 5000 APIs, including API files, functions, structures, enumerations and so on. The commonly used NXOpen C API types and their functions are shown in Table 3.The NXOpen C APIs are named with certain rules, which are related to the module it is located in and the functions it implements, and its naming rule is UF_<area>_<name>.UF is the abbreviation of User Function, which refers to the user function. <area> usually refers to the application module or function module, such as MODL, ASSEM, PART and so on. <name> is usually a description of the function realized by this API, which is usually composed of verbs, nouns, etc., such as ask (query), get (get), create (create), delete (delete), etc. For example, UF_MODL_ask_body_faces() refers to the MODL module, query all the faces on the body, input the TAG of the body can be output to the body of all the faces of the chain table, the function body is as

follows:

```
int UF_MODL_ask_body_faces        //API function body
(
    tag_t body;                   //TAG of the body
    uf_list_p_t * face_list;//Chained lists of TAGs on the face
);
```

NX contains many objects, such as Feature, Body, Face, Edge, Point, etc. Common objects are defined in the header file in the path directory $(UGII_BASE_DIR)\ugopen\uf_object_types.h.

NXOpen C objects are defined in two parts, Type and Subtype. are defined as Type and Subtype, where Subtype describes the object in more detail. The API function for object type acquisition is: UF_OBJ_ask_type_and_subtype. For example, the object type of Face can be categorized into CYLINDRICAL_FACE (cylindrical face), CONICAL_FACE (conical face), SPHERICAL_FACE (spherical face), PLANAR_FACE (planar face) and so on.

Table 3 NXOpen C API Types and Their Functions

Types	Functions
UF_ASSEM	Function modules related to assembly, such as UF_ASSEM_activate_sequence()
UF_ATTR	Function modules related to attributes, e.g. UF_ATTR_count_user_attribute_titles()
UF_CAM	CAM-related function modules, such as UF_CAM_ask_auto_blank()
UF_CFI	Function modules related to file operations, e.g. UF_CFI_ask_file_exist()
UF_CSYS	Function modules related to coordinate system operations, e.g. UF_CSYS_ask_csys_info()
UF_CURVE	Function modules related to curve operations, e.g. UF_CURVE_add_faces_ocf_data()
UF_DISP	Function modules related to display, such as UF_DISP_activate_grid()
UF_DRAW	Function modules related to mapping, e.g. UF_DRAW_add_auxiliary_view()
UF_FACET	Function modules related to small planar body operations, e.g. UF_FACET_cycle_facets()
UF_KF	Function modules related to Knowledge Fusion, such as UF_KF_ask_list()
UF_LAYER	Function modules related to layer operations, e.g. UF_LAYER_ask_category_info()
UF_MODL	Function modules related to model operations, such as UF_MODL_active_part()
UF_MTX	Functional modules related to matrix operations, e.g. UF_MTX2_copy()
UF_OBJ	Function modules related to object operations, such as UF_OBJ_ask_cre_mod_versions()
UF_PART	Function modules related to part operations, such as UF_PART_ask_customer_area ()
UF_SKET	Function modules related to sketch operations, e.g. UF_SKET_add_conics()
UF_UI	Function modules related to user interface operations, e.g. UF_UI_set_status ()
UF_VEC	Functions related to vector operations, e.g. UF_VEC3_distance_to_plane ()

In addition to the C/C++ standard data types, NXOpen C also contains a large number of data types, such as structures, joint structures, enumerations, pointers, etc., whose suffixes are indicated by _s, _u_t, _e, _p_t, etc. The common structure types are as follows:

1) **_t; //Data type (Data type)**
2) **_p_t; // Data type Pointer to that type**
3) **_s; //Structure type**
4) **_e; // Enumeration type**

5) _u_t; //Union type

6) _u_p_t; // Pointer to a union type

7) _f_t; //Pointer to a function

Developers master the NXOpen C help file and the use of related API functions, you can develop the functions you want to achieve in the NX secondary development platform, but if you want to develop a beautiful human-computer interface in conjunction with the UI interface and the realization of its functions, you need to developers on the basis of the NXOpen C API, to be able to skillfully master and use the NXOpen C ++ related programs and code. In the following, the practical application of NXOpen C++ in the secondary development of NX will be carried out in conjunction with the UI interface.

4.2 Specific Applications of NXOpen C++

After the UI design is completed, the completed UI design will be used for code generation to generate three files: *.dlx, *.hpp, and *.cpp. The *.dlx file is the layout file of the UI interface, while the developer can modify and add the program code of the *.hpp and *.cpp files to realize the functions of the UI interface.Block UI Styler dialog box code has its own unique template framework, on these existing frameworks, using the NXOpen C API and C++ help files can be achieved by editing and writing functions. The code required for the different UI Block types of the Block UI Styler dialog box varies, and this section also describes the specific use of typical code for several UI Block types.

(1) NXOpen C++ template code

NXOpen C++ code consists of *.hpp and *.cpp files. The code file generated by any Block UI Styler dialog box has the same template code, and the developer only needs to write code on top of the template to realize the corresponding functions. function, as well as the display function Show(), initialize function initialize_cb(), dialog box display function dialogShown_cb(), apply function apply_cb(), dialog box update function update_cb(), confirm function ok_cb(), dialog box properties function GetBlockProperties (), etc., the need to use these functions, just write the code inside its function, and then finally callback the corresponding function, you can achieve the corresponding function.

(2) Object Selection Filtering

In the dialog box of the UI interface, in the use of the specified object Select Object function, if you do not specify the selected object (body, surface, edge, etc.), you need to filter the selected object, so that the user in the selection of any object, the function can be applied to the selection of objects filtering the key code is as follows:

```
Selection :: SelectionAction action = Selection :: SelectionActionClearAndEnableSpecific;
        //Activate object selection
std::vector<Selection::MaskTriple> maskArray(1);
        //Only one type of object can be selected at a time
maskArray[0] = Selection::MaskTriple( UF_solid_type, UF_solid_body_subtype, 0 );
        //Define the matrix of Bodies
selection0 −> GetProperties ( ) −> GetselectionFilter ( "SelectionFilter", action, maskArray ) ;
        //Getting the properties of the selected objects
```

After the object selection is complete, you can use the following code to get the TAG value of the selected object:

```
PropertyList *blockSelectPros = blockSelect −> GetProperties();
std::vector <NXOpen::TaggedObject*> theblockSelect = blockSelectPros −> GetTaggedObjectVector("SelectedObjects");
delete blockSelectPros;
blockSelectPros = NULL;
```

(3) Enumeration value acquisition

The enumeration type is the Block UI type commonly used in the UI interface of NX secondary development, in the secondary development code writing, the acquisition of the value of the enumeration type and display the contents of the dialog box according to the different enumeration value is the focus of the use of enumeration type. To get the position of the enumeration type and determine whether to hide or show, the code should be written not only in the dialogShow_cb() function, but also in the update_cb() function. The code for the enumeration type is as follows:

```
NXString NX_name = enum0 -> ValueAsString();
string libName = NX_name.GetLocaleText();
PropertyList *enum0Pros = enum0 -> GetProperties();
double theenum0 = enum0Pros -> GetEnum("Value");
delete enum0Pros;
enum0Pros = NULL;
```

(4) Print Window

The key information in the development process sometimes needs to be displayed through the print window, short information can be printed through the uc1601() function, print more information can be printed through the UF_UI_write_listing_window() function to print the key code is as follows:

```
char msg[256];
sprintf(msg, "%d", TAG);
UF_UI_open_listing_window();
UF_UI_write_listing_window(msg);
```

(5) Access to the relative path of the file

In the process of system development, sometimes it is necessary to output and save various information generated during the design process in the form of other text files, Excel files, etc. Sometimes it is also necessary to import the corresponding file information, and at this time it is necessary to access the relative path of the file. Access to the relative path of the file is a common function in the development of the system, the realization of the process of the key code is as follows:

```
char *translation;
UF_translate_variable("UGII_BASE_DIR", &translation);
string BaseDIR1 = translation;
string BaseDIR2 = "\\DLJTOOLS\\Prt\\dlj.prt";
string BaseDIR3 = BaseDIR1 + BaseDIR2;
```

This section lists some of the key code of the commonly used Block UI types, in addition to this, there are many specific applications of NXOpen C++, as well as the implementation of some key features, which will be described in detail in the subsequent paper of this study.

V. conclusion

This paper elaborates on the features of NXOpen C/C++ language, systematically introduces the types and features of NXOpen C API and NXOpen C++, as well as the specific applications in the process of NX secondary development, and also describes in detail the detailed steps and process of NX secondary development environment configuration and platform construction, the workflow of the NXOpen program in the NX software and the UI interface design and its commonly used Block UI type's role and realized functions, to create a theoretical foundation and use of the environment for NXOpen secondary development, so that beginners of secondary development can quickly integrate into the atmosphere and environment of NXOpen C/C++ language.

Reference

[1] Tang Kanglin. Siemens NX secondary development [M]. Beijing: Electronic Industry Press, 2021:1-4.

[2] Andrei Lobov, Tuan Anh Tran. Object-oriented approach to product design using extended NX Open API[J]. Procedia Manufacturing, 2020, 51:1014-1020.

[3] DENG L J. NXOpen-based human-computer interaction design system development for lateral core extraction mechanism of inclined guide pillar[J]. Journal of Advanced Mechanical Design, Systems, and Manufacturing, 2024, 18(5): JAMDSM0056-JAMDSM0056.

[4] Meng Guangjun, Zhang Ming, Wang Anmin. User interface development technology of UG software[J]. Machinery, 2004(5):39-42.

[5] Zuosheng Zhang, Zhenguo Zhu, Chuansheng Chen. Discussion on Ribbon interface design based on UG/Open secondary development platform[J]. Mold Industry, 2021,47(12):4-8.