How to integrate straton controls in a C++ or C# application

Revision

Revision	Date	Description	Author
1	12/11/2013	Initial version (last cmd: HIDEOPTIONDLGVAR).	P. BREYSSE
2	27/02/2014	Last command SETDISPLAY (CTR0569) Add schema for control interfaces Add annexes Few corrections	P. BREYSSE
3	10/07/2014	Add commands (last cmd: MOVESTYLE (CTR0573)) Add reference list for all commands Add CTRxxxx test ID	P. BREYSSE
4	28/07/2014	Add commands: CANFILTER (ID574) SETITEMBREAKPOINT (CTR0575) SETITEMCURPOS (CTR0576)	P. BREYSSE
5	07/01/2015	Change command description INSERTTEXT Add commands: SETCSVSEPARATORCOMA (CTR0577) CANGROUPVAR (CTR0578) GROUPVAR (CTR0579) SELECTTREEITEM (CTR0580)	P. BREYSSE
6	20/01/2015	Add command: DEBUGPROJECT (CTR0581)	P. BREYSSE
7	31/03/2015	Improve description, add notification from controls Add Command: EnableNotif and SetTooltipInfosEx Adding explanation for using structures in C# commands	P. BREYSSE
8	23/04/2015	Add Command: SelectItemType (CTR0584)	P. BREYSSE
9	01/06/2015	Add command: SetItemImageStatus (CTR0585) Change return of “LockItemType” command to boolean	P. BREYSSE
10	16/12/2015	Change Template and check file	P. BREYSSE
11	27/07/2016	Clarify properties PromptInstance and AutoDeclareInst	P. BREYSSE
12	02/08/2017	Add command AutoremoveInst	P. BREYSSE
13	11/09/2017	Add notification W5EDITN_CONTROLDBG	P. BREYSSE
14	05/04/2018	Add commands CanGroupItems and GroupItems	P. BREYSSE
15	03/05/2018	Add command INSERTRUNGAFTER	P. BREYSSE
16	1907/2018	Add command GETSTKEYWORDS	P. BREYSSE
17	23/07/2018	Update SETCALLBACK description	P. BREYSSE
18	04/10/2018	Update GENERATESHARED command	P. BREYSSE
19	13/11/2018	TRACKCHANGES ISTRACKCHANGES GETTRACKCHANGES SETTRACKCHANGES	P. BREYSSE
20	26/03/2019	Add FAQ (how to save/restore scroll position)	P. BREYSSE
21	15/07/2019	InsertMacroBody and InsertMacro are now deprecated (MACRO are not supported)	P. BREYSSE
22	09/09/2019	Add “GetBookmarks” and “SetBookmarks” commands	P. BREYSSE
23	20/01/2020	Add “UseOccurST”	P. BREYSSE
24	28/04/2020	Add “SetFBDOrder”, “GetFBDOrder” and “ActivateFBDOrder” commands	P. BREYSSE
25	30/07/2020	Add InsertCoilParallel, CanInsertCoilParallel, CanInsertCoilAfter, InsertCoilAfter, CanInsertCoilBefore, InsertCoilBefore and SetAutoRemoveVariable commands	P. BREYSSE
26	10/11/2020	Add CanExportHTML and ExportHTML commands	P. BREYSSE
27	26/04/2021	Add UseDBListBox command	P. BREYSSE
28	06/05/2021	Update SETCALLBACK description	P. BREYSSE
29	05/10/2021	Add commands (defines) for command ALIGN	P. BREYSSE
30	14/03/2022	Add command SETAUTORETYPEINST	P. BREYSSE
31	07/04/2022	Update and clarify document + add CANUNDEF command	P. BREYSSE
32	07/11/2022	Add command “SETDEFAULTFBWIDTH”	P. BREYSSE
33	18/01/2023	Add UNICOD interface explanations	P. BREYSSE
34	18/09/2025	Add ExportHTML interface explanations	P. BREYSSE

straton
AUTOMATION

Content

1 Purpose

This document explains how to integrate and use editing straton controls/widgets in your application/IDE.
This document also includes ALL existing commands (return value, parameters…).
But each of these commands are not supported by all controls. To have an overview of which control supports which commands, please refer to the "*Control Interfaces.xls*" file.

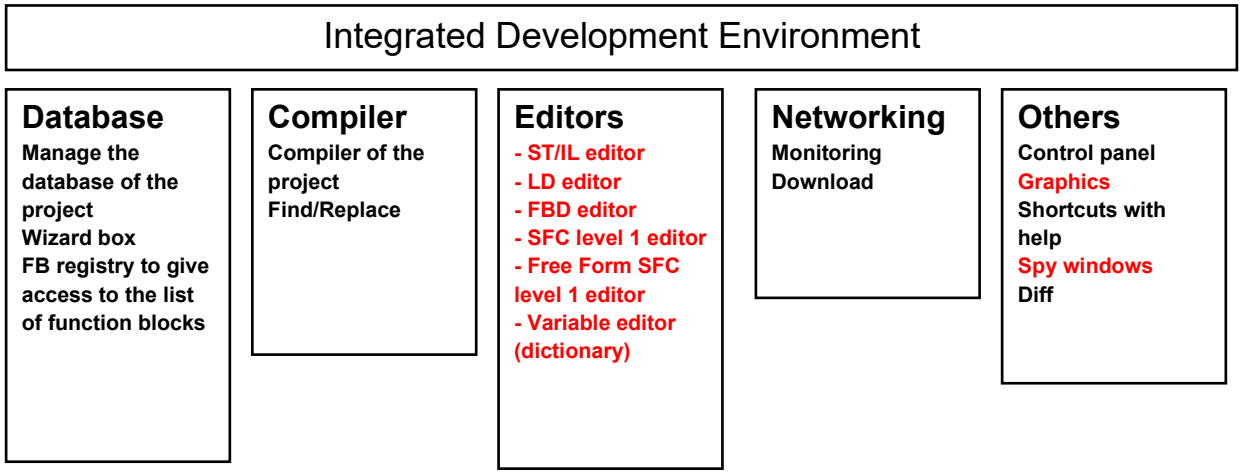
2 Architecture/prerequisites

Controls and widgets are supported only by Windows Operating Systems (from Windows 7 to 11).
Last controls development has been done using Microsoft Visual Studio C++ 2022.
Controls DLLs are statically linked to MFC 143, so you don't need to install MFC on your PC

3 Overview

3.1 Technology

straton IDE (Integrated Development Environment) is based on components that could be provided as controls (we call them *widgets* too). This document deals with component in **red** above:

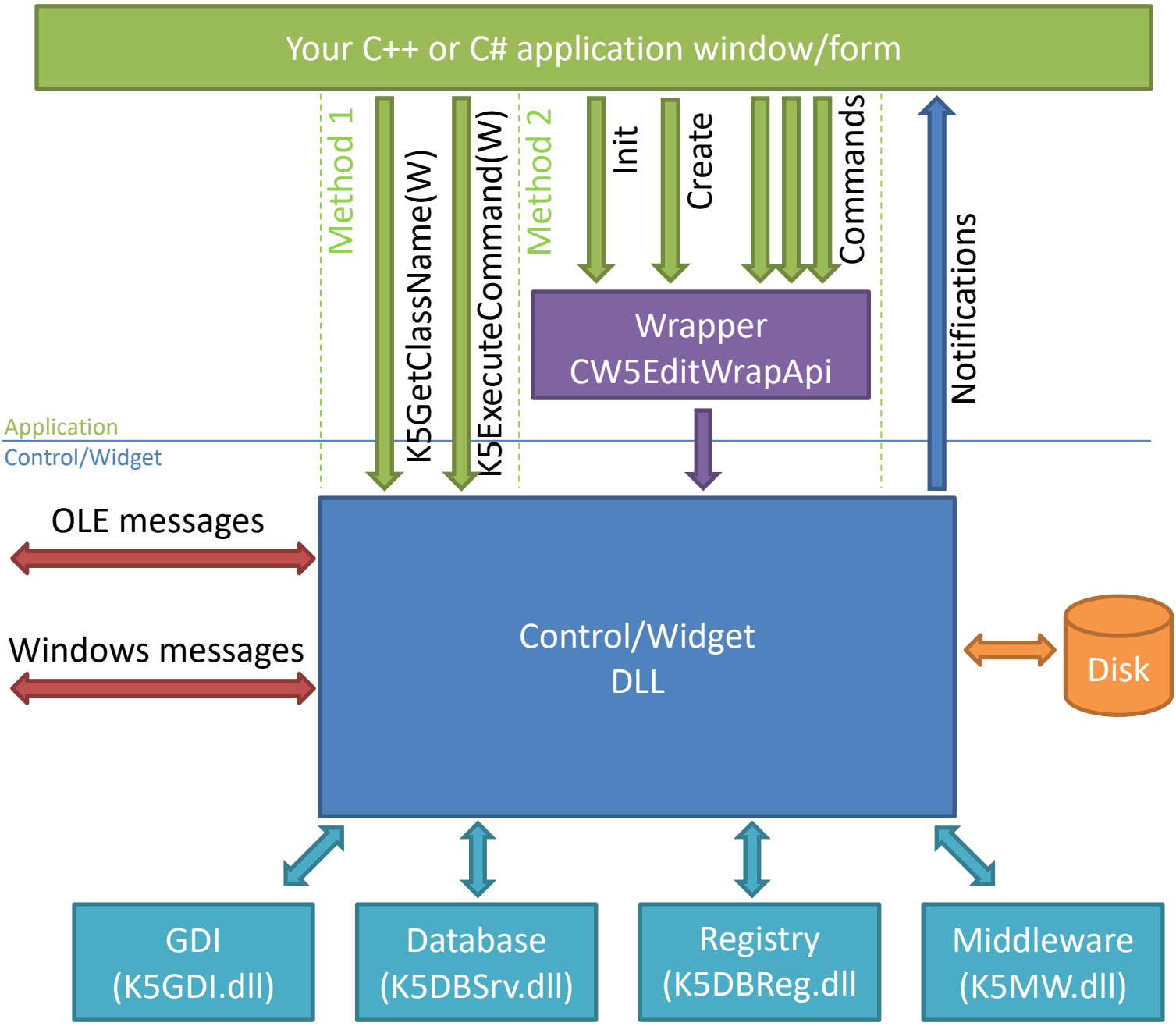


Each editing control is implemented in a Dynamic Link Library (DLL):

- *ST/IL* control is implemented in W5EditST.dll
- *LD* control is implemented in W5EditLD.dll
- *FBD* control is implemented in W5EditFBD.dll
- *SFC* level1 control is implemented in W5EditSFC.dll
- *Free Form SFC* level1 control is implemented in W5EditFF.dll
- *Graphics* control is implemented in W5EditATC.dll
- *Variable editor* (*dictionary*) and *spy windows* are implemented in same W5EditTL.dll (spy windows are *spylists, recipes, softscopes*)

3.2 Control interfaces

This schema describes interfaces between the control/widget and around world.



1. **Interface with Windows messages**

- The control will receive standard window messages (mouse events, keyboard events…)
- The control is able to drag and drop ole data and receive ole data from OLE.

2. **Interface with Disk**

- Control can load and save data from disk

3. **Interface with other straton components**

- Control gets GDI objects (colors, brush, pens…) from **GDI** server (*K5GDI.dll*)
- Control gets information from **database** server (*K5DBSrv.dll*) and can modify database (variable declaration…). The control also receives notifications from data base server (events)
- Control gets information from **registry** server (C block definitions…) (*K5DBReg.dll*)
- Control can connect to **middleware** (for debug mode) and receive notifications from middleware server (*K5MW.dll*)

4. **Interface with your application**

- Control posts **notifications** to its parent window/form
- Control receives **commands** from parent window in direct mode (method1) or using the **wrapper** object (method2). If your application is C# native, control can receive commands from **C# wrapper** **only**.

3.3 Integration of control in customer project

3.3.1 Integrate controls in a C/C++ application

3.3.1.1 Using exports (method1) – MULTIBYTE version

In each DLL editing control there are two export functions dedicated to be used by MULTIBYTE applications:
const char* **K5GetClassName** ();
long **K5ExecuteCommand** (HWND hWnd, int argc, char** argv);
These two export functions are used to create and use each editing control.
Before using control, these steps must be done:

1. Load the control DLL calling *LoadLibrary* function
2. Get the class name of the control calling export function **K5GetClassName**
3. Create a control window using the class name and *CreateEx* function
4. When control is created, all commands can be called using export function **K5ExecuteCommand**
   This function takes 3 arguments:
   - *hWnd*: handle of the control window
   - *argc*: number of strings in array *argv*
   - *argv*: list of parameters (list of char*) to send to control.

Example to create and use a FBD control:

```
HINSTANCE _m_hLibFBD; /* DLL must be loaded as long as you use the FBD control (_m_hLibFBD
should be a class member) */

//step 1: load the DLL control
_m_hLibFBD = LoadLibrary("W5EditFBD.dll");

//step 2: create the FBD window using its class name
typedef const char* (*K5GETCLASSNAME)();
K5GETCLASSNAME pf = (K5GETCLASSNAME)GetProcAddress(_m_hLibFBD, "K5GetClassName");
       const char* ptrClassName = pf(); /* ptrClassName contains now the name of the FBD
class */

//step 3: create the window using the class name:
#define IDC_CTRL      1000
HWND hWnd = CreateWindowEx(0, // Optional window styles.
       ptrClassName, // Window class
       "", // Window text
       WS_CHILD | WS_VISIBLE | WS_BORDER, //window style
       x, y, cx, cy, //position of the window
       hParent, // window handle of the parent container of the control
       IDC_CTRL, // control ID
       hInstance, // Instance handle
       NULL // Additional application data
); //see Visual Studio documentation for CreateWindowEx

//Beware: do not destroy the pointer "ptrClassName"!

//step4: using command (example "SetEnable")
typedef LRESULT(*K5EXECUTECOMMAND) (HWND hWnd, int argc, LPSTR* argv);
K5EXECUTECOMMAND pfExecute = (K5EXECUTECOMMAND)GetProcAddress(_m_hLibFBD,
"K5ExecuteCommand");

char* argv[2]; //SetEnable has 2 arguments
argv[0] = "SetEnable"; //first argument, the command name
argv[1] = "1"; //second argument 1 or 0 for Enable or disable
return pfExecute(hWnd, 2, argv); //command call
```

All other control commands supported by the control are also passed through export **K5ExecuteCommand**.

3.3.1.2 Using exports (method1) – UNICOD version

In each DLL editing control there are two export functions dedicated to be used by MULTIBYTE applications:
const wchar_t* **K5GetClassNameW** ();
long **K5ExecuteCommandW** (HWND hWnd, int argc, wchar_t** argv);
These two export functions are used to create and use each editing control.
Before using control, these steps must be done:

5. Load the control DLL calling *LoadLibrary* function
6. Get the class name of the control calling export function **K5GetClassNameW**
7. Create a control window using the class name and *CreateEx* function
8. When control is created, all commands can be called using export function **K5ExecuteCommandW**
   This function takes 3 arguments:
   - *hWnd*: handle of the control window
   - *argc*: number of strings in array *argv*
   - *argv*: list of parameters (list of wchar_t*) to send to control.

Example to create and use a FBD control in an UNICOD application:

```
HINSTANCE _m_hLibFBD; /* DLL must be loaded as long as you use the FBD control (_m_hLibFBD
should be a class member) */

//step 1: load the DLL control
_m_hLibFBD = LoadLibraryW(L“W5EditFBD.dll”);

//step 2: create the FBD window using its class name
typedef const wchar_t* (*K5GETCLASSNAMEW)();
K5GETCLASSNAMEW pf = (K5GETCLASSNAMEW)GetProcAddress(_m_hLibFBD, “K5GetClassNameW”);
       const wchar_t* ptrClassName = pf(); /* ptrClassName contains now the name of the FBD
class */

//step 3: create the window using the class name:
#define IDC_CTRL      1000
HWND hWnd = CreateWindowExW(0, // Optional window styles.
       ptrClassName, // Window class
       L“”, // Window text
       WS_CHILD | WS_VISIBLE | WS_BORDER, //window style
       x, y, cx, cy, //position of the window
       hParent, // window handle of the parent container of the control
       IDC_CTRL, // control ID
       hInstance, // Instance handle
       NULL // Additional application data
); //see Visual Studio documentation for CreateWindowExW

//Beware: do not destroy the pointer “ptrClassName”!

//step4: using command (example “SetEnable”)
typedef LRESULT(*K5EXECUTECOMMANDW) (HWND hWnd, int argc, LPWSTR* argv);
K5EXECUTECOMMANDW pfExecute = (K5EXECUTECOMMANDW)GetProcAddress(_m_hLibFBD,
"K5ExecuteCommandW");

Wchar_t* argv[2]; //SetEnable has 2 arguments
argv[0] = L“SetEnable”; //first argument, the command name
argv[1] = L“1”; //second argument 1 or 0 for Enable or disable
return pfExecute(hWnd, 2, argv); //command call
```

All other control commands supported by the control are also passed through export **K5ExecuteCommandW**.

3.3.1.3 Using MFC wrapper (method2) – (for both char and UNICOD versions)

A MFC wrapper class is provided. This class called **CW5EditWrapApi** is declared in "*W5EditWrapApi.h*" file (this class inherits from *CWnd* class). This class loads library, creates control and wraps all commands from the DLL. This call be used in a UNICOD or a MULTIBYTE appllications.
Step for using the class:

1. Includes the file "*W5EditWrapApi.h*" and "*W5EditApi.h*"
2. Creates an instance of the class **CW5EditWrapApi** in your application
3. Call function **Init**(LPCTSTR szDll) with the correct DLL path
4. Call function BOOL **Create**(DWORD dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID)
5. When all these steps are done, the object can be used (all commands can be called: **SetEnable**, **IsReadOnly**… see documentation)

Example to create and use a FBD control:

```
//Step1: include "W5EditWrapApi.h" in your application

//step2: create an instance of CW5EditWrapApi class
CW5EditWrapApi _m_ctrlFBD;

//step3: initialize FBD component
_m_ctrlFBD.Init(_T("W5EditFBD.dll"));

//step4: create FBD window
_m_ctrlFBD.Create(WS_CHILD | WS_VISIBLE | WS_BORDER, CRect(0, 0, 100, 100), pParent,
IDC_CTRL);

//step5: using one of the FBD command
_m_ctrlFBD.SetEnable(TRUE);
```

3.3.2 Integrate controls in a C# application

To integrate the editing control, you will find a wrapper file called "*W5WrapEdit.cs*". You will have to integrate it in your C# project.
The wrapper is in charge to wrap all commands supported by controls.

Example to create and use a FBD control:

```
//step 1: create the FBD form
W5.EditFBD ctrl = new W5.EditFBD(); //the FBD DLL is loaded AND the editing control
is created

//step2: using one of the FBD command
ctrl.SetEnable(true);
```

3.4 Manage notifications from control

When control needs information from its parent window or needs to send information to its parent window it sends notification through Window messages map.

3.4.1 List of available notifications from control

Here is the list of all notifications sent by controls (see "*W5EditApi.h*" for complete description):

W5EDITN_DBLCLK	Sent by control when user double click on control and double click is not traited by control
W5EDITN_RCLICK	Sent by control when user right click on control
W5EDITN_SELCHANGE	Sent by control when selection has changed
W5EDITN_EDITPROPS	Sent by control to edit properties of the current selected item
W5EDITN_SETVARFIRSTCHAR	Sent by control when a character that should be first of identifier is hit
W5EDITN_SETVAR	Sent by control when double click on symbol (var, io function block, instance name)
W5EDITN_SETFB	Sent by control when double click on function block body.
W5EDITN_GETFOCUS	Sent by control when gain focus.
W5EDITN_COMPLETION	Sent by control when autocompletion is called
W5EDITN_ADDITEM	Sent by control after new item has been inserted
W5EDITN_REMOVEITEM	Sent by control after item has been deleted
W5EDITN_MODIFYITEM	Sent by control after item has been modified
W5EDITN_LINKTO	Sent by control after hit on a link object
W5EDITN_BEGINDRAG	Sent by control when user begin to drag the selection
W5EDITN_INITARRAYVAR	Sent by control when user has double click on a initial value that is an array
W5EDITN_HEADERSIZECHANGED	Sent by control when user has resized a column
W5EDITN_HEADERSELCHANGED	Sent by control when current column has changed
W5EDITN_HEADERDBLCLK	Sent by control when user has double click on the header
W5EDITN_LISBLOCKCHANGED	Sent by control when list of used blocks has changed
W5EDITN_MODIFIED	Sent by user to indicate that modified flag has changed
W5EDITN_VSCROLL	Sent by control when user has scroll vertically
W5EDITN_HSCROLL	Sent by control when user has scroll horizontally
W5EDITN_CHAR	Sent by control when user has hit a char and control has not trapped it
W5EDITN_KEYDOWN	Sent by control when user has hit a key and control has not trapped it
W5EDITN_RCLICKGRID	Only for fieldbus controls
W5EDITN_RCLICKLIST	Only for fieldbus controls
W5EDITN_UDFBTOUPDATE	used to indicates to parent that there are many udfb to update
W5EDITN_LISTKEYWORD	Sent by control where user want to see keyword list
W5EDITN_OPENUDFB	Sent by control where user want to open selected UDFB
W5EDITN_SFCOPENDEFAULT	Sent by control when user double click on action block
W5EDITN_SFCOPENNOTES	Sent by control when user double click on notes
W5EDITN_SFCOPENP1	Sent by control when user double click on qualifier P1
W5EDITN_SFCOPENN	Sent by control when user double click on qualifier N
W5EDITN_SFCOPENP0	Sent by control when user double click on qualifier P0
W5EDITN_PROPCHANGED	Sent by control when item property has changed in control
W5EDITN_RCLICKSYMBOL	Sent by control when right click on instance
W5EDITN_RCLICKFB	Sent by control when right click on FB body
W5EDITN_COLCHANGE	Sent by control when selected column has changed

W5EDITN_DROP	Sent by control when drag n drop operation has been dropped in control
W5EDITN_ARRANGECOL	Sent by control when dictionary column has been changed by user
W5EDITN_EXPANDCOLLAPSE	Sent by control when a tree item has been expanded or collapsed
W5EDITN_CLICKONICON	Sent by control when a user click on image item in a treelist
W5EDITN_CONTROLDBG	Sent by control when user double clicks on variable during debug
W5EDITN_ADDITEM_FB	Sent by FFLD control when user add a new Function Block in diagram

3.4.2 Receive notifications from control in C/C++

The parent window will receive all notifications from control. This will have to be managed in message map in C++:

Example to manage right click notification from control:
This line must be added to your message map:

```
ON_CONTROL(W5EDITN_RCLICK, IDC_CTRL, OnPopupCtrl)
```

This line manages the notification W5EDITN_RCLICK sent by control IDC_CTRL to its parent. The function OnPopupCtrl will be called when user right clicks in control.

3.4.3 Receive notifications from control in C#

You will have to add the “WndProc” function in your parent form:

```
protected override void WndProc(ref Message m)
{
    switch (m.Msg)
    {
    case WinAPI.Const.WM_COMMAND: //event sent by editing control
    {
        UInt32 wp = (UInt32)m.WParam;
        UInt32 dwID = wp & 0xffff;   // ID of the control
        UInt32 dwNotif = wp >> 16;      //ID of the notification

        if (dwID == IDC_CTRL)
        {
            switch (dwNotif)
            {
            case W5.Const.W5EDITN_RCLICK:
                OnPopupCtrl();
                break;
            }
        }
    }
    break;
    }
}
```

These lines manage the notification W5EDITN_RCLICK sent by control IDC_CTRL to its parent. The function OnPopupCtrl will be called when user right clicks in control.

4 Control commands

In the following section, we describe all commands supported by controls.

General remarks about commands:

- When a command returns a const TCHAR* (LPCTSTR), the return **pointer MUST NOT be deleted** by the caller after use (the control uses the pointer internally). So use this pointer as temporary (the control can delete it whenever). If the pointer should be deleted by the caller, the exception will be indicated in function description.
- Controls don't support all commands. To have the list of supported commands for each control, please refer to the "***Control Interfaces.xls***" file (in this file crosses indicate that command is supported by control).
- The CTRxxxx in parenthesis is the test ID of the command. Do not use it when you call the command.

4.1 Common

Common status commands available for several controls.

4.1.1 SETENABLE (CTR0006)

This function enables or disables control, mouse events, drag n drop. The control can't get focus.
When control is disabled, no external event can be catch by control.

C/C++/C# definition:

```
void SetEnable(bool bEnable);
```

Return:
void

Arguments
bEnable, the enable/disable boolean.

See also
IsEnable

4.1.2 ISENABLE (CTR0007)

This function tests if control is enabled or disabled.

C/C++/C# definition:

```
bool IsEnable();
```

Return:
returns true if control is enable and false if control is disabled

Arguments
none

See also

SetEnable

4.1.3 SETREADONLY (CTR0008)

This function enables or disables control modification.
If the control is read only, no modification can be done. But Drag & Drop and mouse events are available.

C/C++/C# definition:

```
void SetReadOnly(bool bReadOnly);
```

Return:
none

Arguments
bReadOnly; indicates to the control if mode must be read only or not

See also
IsReadOnly

4.1.4 ISREADONLY (CTR0009)

This function tests if control is in read only mode.

C/C++/C# definition:

```
bool IsReadOnly();
```

Return:
true if control is in read only mode

Arguments
none

See also
SetReadOnly

4.1.5 SETMODIFIED (CTR0010)

This function sets the modified flag. The control is marked as modified.
This function is rarely used by integrators. Let the control manage its flag itself.

C/C++/C# definition:

```
void SetModified(bool bModified);
```

Return:
none

Arguments
bModified, indicates the status of the modified flag

See also

IsModified

4.1.6 ISMODIFIED (CTR0011)

This function returns the modified flag status.

C/C++/C# definition:

```
bool IsModified();
```

Return:
true if the content has been modified since last save.

Arguments
none

See also
SetModified

4.1.7 SETDEBUG (CTR0012)

This function sets edit/debug mode.

C/C++/C# definition:

```
void SetDebug(bool bDebug);
```

Return:
none

Arguments
bDebug, indicates to the control the status of the edit/debug mode

See also
IsDebug

Remarks
The project should have been connected to the middleware (K5MW.dll) before call this command (using K5MW_Connect and K5MW_OpenProject). You should implement a global middleware manager in your product/IDE/framework before connecting control to the middleware.
And then you should disconnect global middleware AFTER disconnect all controls.
Here is an example of a good way of working:

```
//open database
DWORD hClient = K5MW_Connect(…);
DWORD hProject = K5MW_OpenProject(hClient, …);

//connect control
TCHAR* argv[2];
argv[0] = _T("SetDebug");
argv[1] = _T("1");
```

```
        bool bSet = pfExecute (hWnd, 2, argv);
…
… do your job with control during debug
…
        //disconnect control
        argv[0] = _T("SetDebug");
        argv[1] = _T("0");
        bSet = pfExecute (hWnd, 2, argv);

        //close middleware after
        K5MW_CloseProject(0xffffffff, hProject); //close all projects
        K5MW_Disconnect(hClient);
```

4.1.8 ISDEBUG (CTR0013)

This function gets the edit/debug flag.

C/C++/C# definition:

```
bool IsDebug();
```

Return:
true if control is in debug mode

Arguments
none

See also
SetDebug

4.1.9 ISEMPTY (CTR0014)

This function tests if control is empty or has content.

C/C++/C# definition:

```
bool IsEmpty();
```

Return:
true if control is empty.

Arguments
none

4.2 Database connection

This section deals with database connection (the database is managed by K5DBSrv.dll).

4.2.1 SETPROJECTPATH (CTR0018)

This function connects the control to the specified straton project path.

C/C++ definition:

```
bool SetProjectPath(LPCTSTR szProjectPath);
```

C# definition:

```
bool SetProjectPath(string szProjectPath);
```

Return:
true if control is connected to the project

Arguments
szProjectPath: the path of the dabase project

Remarks
The project should have been connected to the straton project before call this command (using K5DB_Connect and K5DB_OpenProject). You should implement a global database manager in your product before connecting control to the database.
And then you should disconnect global database AFTER disconnect all controls.
Here is an example of a good way of working:

```
        //open database
        DWORD hClient = K5DB_Connect(…);
        DWORD hProject = K5DB_OpenProject(hClient, …);

        //connect control
        TCHAR* argv[2];
        argv[0] = _T("SetProjectPath");
        argv[1] = _T("myProjectPath");
        bool bSet = pfExecute (hWnd, 2, argv);
…
… do your job with control
…
        //disconnect control
        argv[0] = _T("SetProjectPath");
        argv[1] = _T("");
        bSet = pfExecute (hWnd, 2, argv);

        //close database after
        K5DB_CloseProject(hClient, hProject);
        K5DB_Disconnect(hClient);
```

See also
GetProjectPath

4.2.2 GETPROJECTPATH (CTR0019)

This function returns path of the current project in control.

C/C++ definition:

```
LPCTSTR GetProjectPath();
```

C# definition:

```
string GetProjectPath();
```

Return:
null terminated string (the pointer must not be deleted).

Arguments
none

See also
SetProjectPath, GetProjectID

4.2.3 GETPROJECTID (CTR0020)

This function returns database ID of the current project in control.

C/C++ definition:

```
DWORD GetProjectID();
```

C# definition:

```
UInt32 GetProjectID();
```

Return:
The project ID (32 bits)

Arguments
none

4.2.4 REMOVEPROJECT (CTR0021)

This function removes the current selected project from control.

C/C++ definition:

```
bool RemoveProject(DWORD dwPrj);
```

C# definition:

```
bool RemoveProject(UInt32 dwPrj);
```

Return:
true if project has been removed

Arguments
dwPrj, the project database ID to remove from control

Remarks
Specific command because all controls are single project editors.
This command is only supported by the “MultiTreePrj” control that is a multi-project control.

4.2.5 GETPROGRAMNAME (CTR0022)

This function returns the edited program name or retrieves the program context for the W5EditTL controls (like spylists, recipe, define tree…).

C/C++ definition:

```
LPCTSTR GetProgramName();
```

C# definition:

```
string GetProgramName();
```

Return:

null terminated string (the pointer must not be deleted).

Arguments

none

4.2.6 SETID (CTR0023)

Once project has been connected, this function is used to set a specific ID to the control.
Signification of the ID depends on control.
Gives to control the program database ID / or an object database ID to the tree.

C/C++ definition:

```
void SetID(DWORD dwID);
```

C# definition:

```
void SetID(UInt32 dwID);
```

Return:

none

Arguments

dwID, the ID of the program / object ID

4.3 Edition commands

4.3.1 CANCUT (CTR0024)

This function test if current selection can be cut (copy + clear)

C/C++/C# definition:

```
bool CanCut();
```

Return:
true if selection can be cut

Arguments
none

4.3.2 CUT (CTR0025)

This function cuts the current selection (copy + clear)

C/C++/C# definition:

```
void Cut();
```

Return:
none

Arguments
none

4.3.3 CANCLEAR (CTR0026)

This function test if current selection can be cleared

C/C++/C# definition:

```
bool CanClear();
```

Return:
bool, true if selection can be cleared

Arguments
none

4.3.4 CLEAR (CTR0027)

This function clears the current selection

C/C++/C# definition:

```
void Clear();
```

Return:
none

Arguments

none

4.3.5 CANCOPY (CTR0028)

This function test if current selection can be copied

C/C++/C# definition:

```
bool CanCopy();
```

Return:
bool, true if selection can be copied

Arguments
none

4.3.6 COPY (CTR0029)

This function copies the current selection

C/C++/C# definition:

```
void Copy();
```

Return:
none

Arguments
none

4.3.7 CANPASTE (CTR0030)

This function tests if previously data can be pasted in the control

C/C++/C# definition:

```
bool CanPaste();
```

Return:
bool, true if data can be pasted

Arguments
none

4.3.8 PASTE (CTR0031)

This function pastes previously copied data

C/C++/C# definition:

```
void Paste();
```

Return:
none

Arguments

none

4.3.9 CANUNDO (CTR0032)

This function tests if previously command can be undone

C/C++/C# definition:

```
bool CanUndo();
```

Return:
bool, true if data a command can be cancelled

Arguments
none

4.3.10 UNDO (CTR0033)

This function cancels previously command

C/C++/C# definition:

```
void Undo();
```

Return:
none

Arguments
none

4.3.11 CANREDO (CTR0034)

This function tests if previously undone command can be redone.

C/C++/C# definition:

```
bool CanRedo();
```

Return:
bool, true if undone command can be redone

Arguments
none

4.3.12 REDO (CTR0035)

This function redoes a previously undone command

C/C++/C# definition:

```
void Redo();
```

Return:
none

Arguments

none

4.3.13 EMPTYUNDOSTACK (CTR0046)

This function clears the undo/redo stack.
After this call no undo/redo can be done until a new edit command is executed.

C/C++/C# definition:

```
void EmptyUndoStack();
```

Return:
none

Arguments
none

4.3.14 CANSWAPITEMSTYLE (CTR0047)

This function tests if selected item(s) can be swap (swap style depends on control).

C/C++/C# definition:

```
bool CanSwapStyle();
```

Return:
bool, true if item style can be swapped

Arguments
none

4.3.15 SWAPITEMSTYLE (CTR0048)

This function swaps the style of the selected item(s)

C/C++/C# definition:

```
void SwapStyle();
```

Return:
none

Arguments
None

4.3.16 CANGROUPITEMS (CTR0589)

This function tests if selected items can be group or ungroup.

C/C++/C# definition:

```
bool CanGroupItems(bool bGroup);
```

Return:
bool, true if selected items can be group or ungroup

Arguments
bGroup specifies if selected items must be group or ungroup (true to group items, false to ungroup items)

4.3.17 GROUPITEMS (CTR0590)
This function groups or ungroups the selected items.

C/C++/C# definition:
```
bool GroupItems(bool bGroup);
```

Return:
bool, true if selected items has been group or ungroup

Arguments
bGroup specifies if selected items must be group or ungroup (true to group items, false to ungroup items)

4.4 Display commands

4.4.1 CANSETZOOM (CTR0039)
This function tests if zoom can be modified.

C/C++/C# definition:
```
bool CanSetZoom();
```

Return:
bool, true if zoom can be modified

Arguments
none

4.4.2 SETZOOM (CTR0040)
This function changes the zoom ratio.

C/C++/C# definition:
```
void SetZoom(int iDir, int iRatio);
```

Return:
none

Arguments
iDir specifies a zoom direction

W5ZOOM_IN	Increase zoom ratio
W5ZOOM_OUT	Decrease zoom ratio
W5ZOOM_AT	Set zoom ratio at value specified by iRatio
W5ZOOM_FIT	Adjust zoom to the whole control content
W5ZOOM_FITSEL	Adjust ratio to the whole selection

iRatio is used only when iDir value is W5ZOOM_AT (this is a % value).

Remarks
All controls don't support all iDir values.

4.4.3 GETZOOM (CTR0041)

This function gets the zoom ratio (in %)

C/C++/C# definition:

```
int GetZoom();
```

Return:
int, the zoom ratio (in %)

Arguments
none

4.4.4 CANSETGRID (CTR0042)

This function tests if grid can be displayed.

C/C++/C# definition:

```
bool CanSetGrid();
```

Return:
bool, true if grid can be displayed

Arguments
none

4.4.5 SETGRID (CTR0043)

This function displays/hides the grid.

C/C++/C# definition:

```
void SetGrid(bool bGrid);
```

Return:
none

Arguments
bGrid, if true displays grid

4.4.6 ISGRIDVISIBLE (CTR0044)

This function tests if grid is displayed.

C/C++/C# definition:

```
bool IsGridVisible();
```

Return:
bool, true if grid is displayed

Arguments
none

4.4.7 ISBOOKMARK (CTR0049)

This function tests if current selection has bookmark.

C/C++/C# definition:

```
bool IsBookmark();
```

Return:
bool, true selection has bookmark

Arguments
none

4.4.8 SETBOOKMARK (CTR0050)

This function set/unset bookmark at current position

C/C++/C# definition:

```
void SetBookmark(bool bSet);
```

Return:
none

Arguments
bSet, true to set bookmark, false to unset bookmark

4.4.9 GONEXTBOOKMARK (CTR0051)

This function selects the next bookmark in document.
If pass the end of document, search from the beginning.

C/C++/C# definition:

```
void GoNextBookmark();
```

Return:
none

Arguments
none

4.4.10 GOPREVBOOKMARK (CTR0052)

This function selects the previous bookmark in document.
If pass the beginning of document, search from the end.

C/C++/C# definition:

```
void GoPrevBookmark();
```

Return:
none

Arguments
none

4.4.11 DELETEALLBOOKMARK (CTR0053)

This function removes all bookmarks in document.

C/C++/C# definition:

```
void DeleteAllBookmark();
```

Return:
none

Arguments
none

4.4.12 GETBOOKMARKS (CTR0597)

This function retrieves the list of bookmarks in document. The string uses the compiler syntax for each bookmarks position and each of them are separated by ";".

C/C++ definition:

```
LPCTSTR GetBookmarks();
```

C# definition:

```
string GetBookmarks();
```

Return:
String, list of all bookmarks position in document

Arguments:
none

4.4.13 SETBOOKMARKS (CTR0598)

This function sets a list of bookmarks in document. The string uses the compiler syntax for each bookmarks position and each of them are separated by ";". This function has to be used with "GetBookmarks" command.

C/C++ definition:

```
int SetBookmarks(LPCTSTR szBookmarks);
```

C# definition:

```
int GetBookmarks(string sBookmarks);
```

Return:

int, returns the number of bookmarks set in document. Value “-1” means that no bookmark has been set in document. Value “0” means that all bookmarks have been deleted.

Arguments:
string, list of bookmarks to set.

4.4.14 USEOCCURST (CTR0599)

This function activates/deactivates the displaying of occurences found in ST editor editor when user uses Find/Replace feature or select a string using double click on it.

C/C++ definition:

```
BOOL UseOccurST(BOOL bSet);
```

C# definition:

```
bool UseOccurST (bool bSet);
```

Return:
bool, returns true is option has changed.

Arguments:
bool, Boolean for activating/de-activating option.

4.5 Selection

4.5.1 HASSELECTION (CTR0015)

This function tests if there is a current selection

C/C++/C# definition:

```
bool HasSelection();
```

Return:
bool, true if selection is not empty

Arguments
none

4.5.2 CANSELECTALL (CTR0036)

This function tests if user can select all items in control

C/C++/C# definition:

```
bool CanSelectAll();
```

Return:
bool, true if user can select all items

Arguments
none

4.5.3 SELECTALL (CTR0037)

This function selects all items in control

C/C++/C# definition:

```
void SelectAll();
```

Return:
none

Arguments
none

4.5.4 DESELECTALL (CTR0038)

This function deselects all items in control

C/C++/C# definition:

```
void DeselectAll();
```

Return:
none

Arguments

none

4.5.5 ENSURESELVISIBLE (CTR0045)

This function ensures the selection becomes visible (scroll control if needed)

C/C++/C# definition:

```
void EnsureSelVisible();
```

Return:
none

Arguments
none

4.5.6 SELECTITEM (CTR0534)

This function selects a specific item in the control.

C/C++ definition:

```
void SelectItem(DWORD_PTR dwData, bool bDeselectAll);
```

C# definition:

```
void SelectItem(IntPtr dwData, bool bDeselectAll);
```

Return:
none

Arguments
dwData, the object ID (can be also a pointer)
bDeselectAll, if true, deselect all items before select this one

4.5.7 SELECTITEMS (CTR0535)

This function selects many items in the control.

C/C++ definition:

```
void SelectItems(int nbItem, void* arrItem, bool bDeselectAll);
```

C# definition:

```
void SelectItems(int nbItem, IntPtr arrItem, bool bDeselectAll);
```

This function is not supported by C# wrapper!

Return:
none

Arguments
nbItem, the number of items contained in the array
arrItem, the item array (number of items is specified in nbItem)
bDeselectAll, if true deselect all items before select the new ones

4.6 File exchange commands

4.6.1 CANSAVE (CTR0054)

This function tests if control contents can be saved on disk

C/C++/C# definition:

```
bool CanSave();
```

Return:
bool, true if content can be saved

Arguments
none

4.6.2 SAVE (CTR0055)

This function saves the content of control on disk or in a database

C/C++ definition:

```
void Save(LPCTSTR szPath);
```

C# definition:

```
void Save(string szPath);
```

Return:
none

Arguments
szPath the path file. Can be “” for specific controls.

4.6.3 LOAD (CTR0056)

This function loads the content of disk/database into control

C/C++ definition:

```
void Load(LPCTSTR szPath);
```

C# definition:

```
void Load(string szPath);
```

Return:
none

Arguments
szPath the path file. Can be “” for specific controls.

4.6.4 INSERTFILE (CTR0057)

This function inserts the content of the file into the control at the current position.

C/C++ definition:

```
void InsertFile(LPCTSTR szPath);
```

C# definition:

```
void InsertFile(string szPath);
```

Return:
none

Arguments
szPath the path file

4.6.5 CANINSERTFILE (CTR0058)

This function test if the content of the file can be inserted at the current position in control.

C/C++/C# definition:

```
bool CanInsertFile();
```

Return:
bool true if content can be inserted

Arguments
none

4.6.6 CANEXPORTHTML (CTR0610)

This is a private use of controls

C/C++/C# definition:

```
bool CanExportHTML();
```

Return:
bool true if content can be exported

Arguments
none

4.6.7 EXPORTHTML (CTR0611)

This is a private use of controls

C/C++ definition:

```
bool ExportHTML(LPCTSTR szExportSettings);
```

C# definition:

```
bool ExportHTML(string szExportSettings);
```

Return:

bool: true if content has been exported as HTML

Arguments
szExportSettings: the settings used to export content as HTML file

This parameter has same syntax as an ini file.
Parameters are:

[SETTINGS]
DOWNLOAD_FOLDER=C:\\Temp\\tstHtml - this is the folder where the html page will be generated
FILE_DEST_HTML=C:\\Temp\\tstHtml\\export.html – this is the full page of the HTML page to be generated (and must be in same folder as above)
INTERNAL_NAME=gra_newgraphics – the internal name of the current page (must be unic)
TITLE=NewGraphics – the HTML page main title (will appear on tab of the browser)
REFRESH=1000 – the refresh rate of the web page (in milliseconds)
REQUEST_TYPE=0 – the request types (could be 0 = CGI, 1 = Fast CGI, 2 = Web Sockets)
APP_NAME=tstGra1 – the straton project name
SINGLECALL=ON –MUST be set to ON
COPY_HTMLFOLDER=ON – if you want to copy the full HTML folder from straton when generating HTML page, you should set it to ON. But if you want to generate several HTML pages using widget, the first generation should have ON as parameter and other should have OFF (avoid copying multiple times the same folder).

4.7 Text exchange commands

4.7.1 SETTEXT (CTR0059)

This function loads the content of string into control.

C/C++ definition:

```
void SetText(LPCTSTR szText);
```

C# definition:

```
void SetText(string szText);
```

Return:
none

Arguments
szText the string to set

4.7.2 GETTEXT (CTR0060)

This function puts the content of the control in a string.

C/C++ definition:

```
LPCTSTR GetText();
```

C# definition:

```
string GetText();
```

Return:
null terminated string (the pointer must not be deleted).

Arguments
none

4.7.3 GETTEXTLENGHT (CTR0061)

This function retrieves the text size of the control.

C/C++/C# definition:

```
int GetTextLenght();
```

Return:
the text size

Arguments
none

See also
GetText

4.7.4 GETSELTEXTLENGTH (CTR0062)

This function retrieves the selected text size in the control.

C/C++/C# definition:

```
int GetSelTextLength();
```

Return:
the selected text size

Arguments
none

See also
GetSelText

4.7.5 GETSELTEXT (CTR0063)

This function puts the content of the control's selection in a string.

C/C++ definition:

```
LPCTSTR GetSelText();
```

C# definition:

```
string GetSelText();
```

Return:
null terminated string (the pointer must not be deleted).

Arguments
none

4.7.6 CANINSERTTEXT (CTR0064)

This function tests if text can be inserted in control.

C/C++/C# definition:

```
bool CanInsertText();
```

Return:
bool, true if text can inserted

Arguments
none

4.7.7 INSERTTEXT (CTR0065)

This function copies the string content at given position.

C/C++ definition:

```
void InsertText(int iX, int iY, LPCTSTR szText);
```

C# definition:

```
void InsertText(int iX, int iY, string szText);
```

Return:
none

Arguments
iX, the x logical coordinate of position – position begins from left
iY, the y logical coordinate of position – position begins from top
szText, the text to insert at given position

4.8 Location and find/replace

4.8.1 LOCATEERROR (CTR0066)

This function set focus at the position given by the location string. This location string must be in compiler string format.

C/C++ definition:

```
void LocateError(LPCTSTR szError);
```

C# definition:

```
void LocateError(string szError);
```

Return:
none

Arguments
szError, the location string (this string must have the compiler error format)

4.8.2 FINDREPLACE (CTR0067)

This function finds or replaces specified string in the control.

C/C++ definition:

```
long FindReplace(LPCTSTR szFind, LPCTSTR szReplace, DWORD dwCommand, DWORD dwFlags, HWND
hWndListBox);
```

C# definition:

```
Int32 FindReplace(string szFind, string szReplace, UInt32 dwCommand, UInt32 dwFlags,
IntPtr hListBox);
```

Return:
long, 0 if FindReplace failed and positive value if FindReplace succeeded.
Can be a combination of:

W5RESULT_FIND	0x00001	find the occurence
W5RESULT_REPLACED	0x00002	replace the occurence

Arguments
szFind, the string to find
szReplace, the substitutive string (if command is a replace command)
dwCommand, the function FindReplace can be used in different modes. The command value can take one of these values:

W5FIND_NEXT	0x0001	find next
W5FIND_REPLACEANDNEXT	0x0002	replace and find next
W5FIND_REPLACEALL	0x0004	replace all found strings
W5FIND_ACTIVESTEP	0x0008	find the active step in SFC
W5FIND_FIRST	0x0011	find string including current selection

dwFlags, the search flags (for all control except the FFLD)
Can be a combination of the values:

W5FIND_WHOLEWORD	0x0001	only search replace a full word
W5REPLACE_ALL	0x0002	when all occurences of substring has to be replaced
W5ASSIGNATION	0x0004	search for assigned symbols(coil, := in ST...)
W5DONTLOOP	0x0008	do not loop at the end of search
W5NEXTITEM	0x0010	select next item
W5PREVITEM	0x0020	select previous item
W5FOCUS	0x0040	focus control when find string
W5ANYWHERE	0x0080	search anywhere in line, not only at beginning
W5FIND_MATCHCASE	0x0100	match case
W5FIND_VARPREFIX	0x0200	when searching in dictionary, search only variable by prefix

For FFLD, the dwFlags can be a combination of these values:

W5FINDFFLD_WHOLEWORD	0x0001	search the whole word(s) only
W5FINDFFLD_MATCHCASE	0x0002	search for the case sensitive string
W5FINDFFLD_ASSIGNATION	0x0004	search for assigned symbols(coil, := in ST...)
W5FINDFFLD_DONTLOOP	0x0008	do not loop at the end of document
W5FINDFFLD_BACKWARD	0x0010	find and replace go to previous found item
W5FINDFFLD_JUMP	0x0020	search in jump labels
W5FINDFFLD_FOCUS	0x0040	focus control when find string
W5FINDFFLD_NETLABEL	0x0080	search in network labels
W5FINDFFLD_DATAIN	0x0100	search in data in
W5FINDFFLD_DATAOUT	0x0200	search in data out
W5FINDFFLD_COIL	0x0400	search in coils
W5FINDFFLD_CONTACT	0x0800	search in contacts
W5FINDFFLD_FBTYPE	0x1000	search in function block types (names)
W5FINDFFLD_FBINSTANCE	0x2000	search in function block instances
W5FINDFFLD_NETTAG	0x4000	search in network tags
W5FINDFFLD_COMMENT	0x8000	search in comments

hWndListBox, the handle of the list box where to display the results of search/replace

Remarks:
for more details, please see the “W5EditApi.h” file.

4.8.3 SEARCHIN (CTR0068)

This function finds or replaces specified string in the control.
Remark: there is no UNICOD version for this function. Use command “FindReplace” instead.

C/C++ definition:

```
long SearchIn(str_W5FindReplace* pStrFR);
```

C# definition:

```
Int32 SearchIn(IntPtr pStrFR);
```

To use this function in C# please refer to Annexe “Using structures in C# wrappers”.
By the way, the best way of working is using “FindReplace” instead of “SearchIn”.

Return:
long, 0 if search failed and positive value if search succeeded.

Arguments
pStrFR is a pointer to structure “str_W5FindReplace” defined in “W5EditApi.h”

Remarks
For more information, please see the “FindReplace” function.

See also
FindReplace

4.8.4 GOTO (CTR0069)
Obsolete, see LocateError.

4.8.5 GETCURPOS (CTR0070)
This function gets the coordinates (in control units) of the mouse.

C/C++ definition:
```
long GetCurPos();
```

C# definition:
```
Int32 GetCurPos();
```

Return:
returns the current coordinates in a int32.
x coordinate can be retrieved in low word
y coordinate can be retrieved in high word

Arguments
none

4.8.6 GETSELSIZE (CTR0071)
This function gets the size of the current selection in control units.

C/C++ definition:
```
long GetSelSize();
```

C# definition:
```
Int32 GetSelSize();
```

Return:
returns the selection size
width can be retrieved in low word
height can be retrieved in high word

Arguments
none

4.8.7 GETSELPOS (CTR0072)

This function gets the coordinates (in control units) of the selection.

C/C++ definition:

```
long GetSelPos();
```

C# definition:

```
Int32 GetSelPos();
```

Return:
returns the current coordinates in a int32.
x coordinates can be retrieved in low word
y coordinate can be retrieved in high word

Arguments
none

4.8.8 GETSYMBOLPOS (CTR0073)

This function gets the coordinates (in control units) of the selected symbol (variable or instance) if any.

C/C++ definition:

```
long GetSymbolPos();
```

C# definition:

```
Int32 GetSymbolPos();
```

Return:
returns the current coordinates in a int32.
x coordinate can be retrieved in low word
y coordinate can be retrieved in high word

Arguments
none

4.8.9 GETCARET (CTR0074)

This function gets the coordinates (in control units) of the caret/selection.

C/C++ definition:

```
long GetCaret();
```

C# definition:

```
Int32 GetCaret();
```

Return:

returns the current coordinates in a int32.
x coordinates can be retrieved in low word
y coordinate can be retrieved in high word

Arguments
none

4.8.10 GOTOXY (CTR0075)

This function puts the caret/selection at the given position. The coordinates are in control units.

C/C++ definition:

```
void GotoXY(DWORD dwX, DWORD dwY);
```

C# definition:

```
void GotoXY(UInt32 dwX, UInt32 dwY);
```

Return:
none

Arguments
dwX, the x coordinate of the caret/selection (in control units)
dwY, the y coordinate of the caret/selection (in control units)

4.8.11 SAVESEL (CTR0076)

This function saves the current caret/selection.

C/C++/C# definition:

```
void SaveSel();
```

Return:
none

Arguments
none

4.8.12 RESTORESEL (CTR0077)

This function restores the previously saved caret/selection.

C/C++/C# definition:

```
void RestoreSel();
```

Return:
none

Arguments
none

4.8.13 FINDVARINPUT (CTR0078)

This function retrieves the symbol name connected to the selected input.

C/C++ definition:

```
LPCTSTR FindVarInput(LPCTSTR szInput);
```

C# definition:

```
string FindVarInput(string szInput);
```

Return:

symbol name: null terminated string (in C++ the pointer must not be deleted).
null in C++ if not found
“” in C# if not found

Arguments

szInput, the input name

4.9 Document properties

4.9.1 CANFORMATPROGRAM (CTR0079)

For future…

4.9.2 FORMATPROGRAM (CTR0080)

For future…

4.9.3 CANVIEWINFO (CTR0081)

This function tests if infos (depending to the control) can be displayed.

C/C++/C# definition:

```
bool CanViewInfo();
```

Return:

bool, true if infos can be displayed

Arguments

none

4.9.4 VIEWINFO (CTR0082)

This function displays infos (depending to the control).

C/C++/C# definition:

```
bool CanViewInfo();
```

Return:

none

Arguments

none

4.9.5 GETCELLHEIGHT (CTR0083)

This function returns the height of on cell in control (in pixels).

C/C++/C# definition:

```
int GetCellHeight();
```

Return:

the cell height in pixels

Arguments

none

4.9.6 GETCELLWIDTH (CTR0084)

This function returns the width of on cell in control (in pixels).

C/C++/C# definition:

```
int GetCellWidth();
```

Return:
the cell width in pixels

Arguments
none

4.9.7 SETCELLHEIGHT (CTR0085)

This function set the height of cells in control (in pixels).

C/C++/C# definition:

```
void SetCellHeight(int iHeight);
```

Return:
none

Arguments
the cell height in pixels

4.9.8 SETCELLWIDTH (CTR0086)

This function set the width of cells in control (in pixels).

C/C++/C# definition:

```
void SetCellWidth(int iWidth);
```

Return:
none

Arguments
the cell width in pixels

4.9.9 GETHEIGHT (CTR0087)

This function returns the height of document (in pixels).

C/C++/C# definition:

```
int GetHeight();
```

Return:
the document height in pixels

Arguments
none

4.9.10 GETWIDTH (CTR0088)

This function returns the width of document (in pixels).

C/C++/C# definition:

```
int GetWidth();
```

Return:
the document width in pixels

Arguments
none

4.9.11 TRACKCHANGES (CTRL0593)

This function activate the track changes in control. TrackChnages feature track all modifications in control (clear, move, add…) and display a mark for each one.

C/C++/C# definition:

```
bool TrackChanges(bool bTrack);
```

Return:
Returns true if feature is supported by control.

Arguments
Bool bTrack, true to activate track changes, false to desactivate track changes.

4.9.12 ISTRACKCHANGES (CTRL0594)

This function tests if track change is active in current control.

C/C++/C# definition:

```
bool IsTrackChanges();
```

Return:
Returns true if track change is active.

Arguments
none

4.9.13 GETTRACKCHANGES (CTRL0595)

This function returns the list of changes in document (internal text format to be used only with function SetTrackChanges).
C/C++ definition:

```
LPCTSTR GetTrackChanges ();
```

C# definition:

```
string GetTrackChanges ();
```

Return:
null terminated string (the pointer must not be deleted).

Arguments
none

4.9.14 SETTRACKCHANGES (CTRL0596)

This function set a new list of changes in document (internal text format to be used only with function GetTrackChanges).
C/C++ definition:

```
bool SetTrackChanges (LPCTSTR szTrackChanges);
```

C# definition:

```
bool SetTrackChanges (string szTrackChanges);
```

Return:
Returns true if function is supported by control.

Arguments
szTrackChanges is a string returned by a previously call to GetTrackChanges.

4.10 Paint

4.10.1 PAINTTODC (CTR0089)

This function paints the entire control in the Device Context.

C/C++ definition:

```
int PaintToDC(HDC hDC, int iLeft, int iTop, int iRight, int iBottom);
```

C# definition:

```
int PaintToDC(IntPtr hDC, int iLeft, int iTop, int iRight, int iBottom);
```

Return:

the height of the painted rectangle.

Arguments

hDC, the handle of the Device Context
iLeft, the left position of the rectangle in Device Context
iTop, the top position of the rectangle in Device Context
iRight, the right position of the rectangle in Device Context
iBottom, the bottom position of the rectangle in Device Context

4.10.2 RELOADBITMAP (CTR0090)

This function updates and redraws all bitmaps in controls.

C/C++/C# definition:

```
bool ReloadBitmap();
```

Return:

true if all bitmaps have been redrawn.

Arguments

none

4.11 Function blocks/symbols

4.11.1 SETCURRENTFB (CTR0091)

This function specifies the block to drag on the next drag n drop operation.

C/C++ definition:

```
bool SetCurrentFB(LPCTSTR szName, DWORD dwID, int nbIn, int nbOut, BOOL bInstanciable,
BOOL bDBObject);
```

C# definition:

```
bool SetCurrentFB(string szName, UInt32 dwID, int nbIn, int nbOut, bool bInstanciable,
bool bDBObject);
```

Return:
true if function is supported by control.

Arguments
szName, the name of the Function Block
dwID, the registry ID (if blocks comes from registry) or database ID (if blocks comes from database)
nbIn: number of inputs
nbOut: number of outputs
bInstanciable: specifies if block is instanciable (function block or function)
bDBObject: specifies if block comes from database or not

4.11.2 CANINSERTFB (CTR0092)

This function tests if Function Block can be inserted at current position.

C/C++/C# definition:

```
bool CanInsertFB();
```

Return:
true if FB can be inserted

Arguments
none

4.11.3 INSERTFB (CTR0093)

This function inserts block at current position.

C/C++ definition:

```
void InsertFB(LPCTSTR szName);
```

C# definition:

```
void InsertFB(string szName);
```

Return:

none

Arguments
szName, the name of the Function Block to insert

4.11.4 SETFB (CTR0094)
This function changes the selected block.

C/C++ definition:
```
void SetFB(LPCTSTR szName, DWORD dwID, int nbIn, int nbOut, BOOL bInstanciable, BOOL bDBObject);
```

C# definition:
```
void SetFB(string szName, UInt32 dwID, int nbIn, int nbOut, bool bInstanciable, bool bDBObject);
```

Return:
none

Arguments
szName, the name of the Function Block
dwID, the registry ID (if blocks comes from registry) or database ID (if blocks comes from database)
nbIn: number of inputs
nbOut: number of outputs
bInstanciable: specifies if block is instanciable (function block or function)
bDBObject: specifies if block comes from database or not

4.11.5 GETFB (CTR0095)
This function gets the name of the selected block.

C/C++ definition:
```
LPCTSTR GetFB();
```

C# definition:
```
string GetFB();
```

Return:
function name
null in C++ if no FB is selected
“” in C# if no FB selected

Arguments
none

4.11.6 CANINSERTSYMBOL (CTR0096)
This function tests if symbol can be inserted at current position.

C/C++/C# definition:

```
bool CanInsertSymbol();
```

Return:
true if symbol can be inserted

Arguments
none

4.11.7 INSERTSYMBOL (CTR0097)

This function inserts symbol at current position.

C/C++ definition:

```
void InsertSymbol(LPCTSTR szSymbol);
```

C# definition:

```
void InsertSymbol(string szSymbol);
```

Return:
none

Arguments
szSymbol, the symbol name to insert at current position

4.11.8 GETSYMBOLNAME (CTR0098)

This function gets symbol at current position.

C/C++ definition:

```
LPCTSTR GetSymbolName();
```

C# definition:

```
string GetSymbolName();
```

Return:
the symbol name

Arguments
none

4.11.9 GETTYPENAME (CTR0099)

This function gets symbol type at current position.

C/C++ definition:

```
LPCTSTR GetTypeName();
```

C# definition:

```
string GetTypeName();
```

Return:
the type name

Arguments
none

4.11.10 GETITEMTYPE (CTR0100)

This function gets type of the selected item.

C/C++ definition:

```
DWORD GetItemType();
```

C# definition:

```
UInt32 GetItemType();
```

Return:
the item type
Can be one of these values:

W5TYPE_NONE	0	W5TYPE_SECTIONPRG	33
W5TYPE_NETWORK	1	W5TYPE_PRG	34
W5TYPE_COIL	2	W5TYPE_FOLDER	35
W5TYPE_COIL_I	3	W5TYPE_SECTIONFIELDBUS	36
W5TYPE_COIL_R	4	W5TYPE_SECTIONPROFILES	37
W5TYPE_COIL_S	5	W5TYPE_SECTIONIOS	38
W5TYPE_COIL_P	6	W5TYPE_SECTIONGLOBALDEFINE	39
W5TYPE_COIL_N	7	W5TYPE_SECTIONVARS	40
W5TYPE_COMMENT	8	W5TYPE_SECTIONGRA	41
W5TYPE_CONTACT	9	W5TYPE_SECTIONRCP	42
W5TYPE_CONTACT_I	10	W5TYPE_SECTIONSPY	43
W5TYPE_CONTACT_P	11	W5TYPE_SECTIONCURVE	44
W5TYPE_CONTACT_N	12	W5TYPE_SECTIONSTRINGTABLE	45
W5TYPE_CONTACT_IP	13	W5TYPE_SECTIONSIGNAL	46
W5TYPE_CONTACT_IN	14	W5TYPE_FILE	47
W5TYPE_CORNER	15	W5TYPE_CONSTANT	48
W5TYPE_JUMP	16	W5TYPE_SUBPRG	49
W5TYPE_RETURN	17	W5TYPE_KEYWORD	50
W5TYPE_LABEL	18	W5TYPE_UDFB	51
W5TYPE_OR	19	W5TYPE_EXTERN	52
W5TYPE_POWERL	20	W5TYPE_SECTIONBINDING	53
W5TYPE_POWERR	21	W5TYPE_SECTIONHMI	54
W5TYPE_VARIABLE	22	W5TYPE_SECTIONHMISTRING	55
W5TYPE_BLOCK	23	W5TYPE_SECTIONHMIBMP	56
W5TYPE_BLOCK_IN	24	W5TYPE_SECTIONHMIFONT	57

W5TYPE_BLOCK_INI	25
W5TYPE_BLOCK_OUT	26
W5TYPE_BLOCK_OUTI	27
W5TYPE_LINE	28
W5TYPE_STEP	29
W5TYPE_MACRO	30
W5TYPE_TRANS	31
W5TYPE_PROJECT	32

W5TYPE_SECTIONHMISCREEN	58
W5TYPE_SCREEN	59
W5TYPE_SECTIONTYPES	60
W5TYPE_SCREENMASK	61
W5TYPE_SECTIONIEC850	62
W5TYPE_TOOLSMENU	63
W5TYPE_ISTEP	64

For description of each value see “W5EditApi.h"

Arguments
none

4.11.11 GETITEMTYPESEL (CTR0101)

This function gets selected text for context help.

C/C++ definition:

```
LPCTSTR GetItemTypeSel();
```

C# definition:

```
string GetItemTypeSel();
```

Return:
the item type string

Arguments
none

4.11.12 GETFIRSTCHAR (CTR0102)

This function gets the last char entered by user in control. This function should be called after parent window has received a notification “W5EDITN_CHAR” from the control.

C/C++ definition:

```
TCHAR GetFirstChar();
```

C# definition:

```
IntPtr GetFirstChar();
```

Return:
the char entered by user before the notification “W5EDITN_CHAR”.

Arguments
none

See also

Manage notifications from control

4.11.13 GETUSEDBLOCK (CTR0103)

This function gets the full list of blocks (function and FB) used in control. Each FB name is separated by a tab.

C/C++ definition:

```
LPCTSTR  GetUsedBlock();
```

C# definition:

```
string GetUsedBlock();
```

Return:
the list of used blcoks

Arguments
none

4.11.14 GETDBID (CTR0104)

This function gets the full array of database ID of all symbols used in control.

C/C++ definition:

```
int GetDBID(DWORD* pArrID);
```

C# definition:

```
int GetDBID(IntPtr pArrID);
```

Return:
the number of IDs used in control.

Arguments
an array of IDs (size is determinate by function return).
This can be null.

Remarks:
This function must be called in two times:

1. returns the size of the array (parameter pArrID is null). Function return determinates the array size of pArrID for the second call.
2. fill the array

4.11.15 GETINPUTBLOCKVAR (CTR0105)

This function gets the variable name connected on the input pin.

C/C++ definition:

```
LPCTSTR  GetInputBlockVar(int iPin);
```

C# definition:

```
string GetInputBlockVar(int iPin);
```

Return:
the input connected variable name
in C++ returns null if variable not found
in C# returns “” if variable not found

Arguments
the pin index (begins at 0)

4.11.16 CANEDITPINS (CTR0106)

This function tests if block pin names can be edited in control.

C/C++/C# definition:

```
bool CanEditPins();
```

Return:
true if pins can be edited on the current selection

Arguments
none

4.11.17 EDITPINS (CTR0107)

This function edits the pins of the current selection.

C/C++/C# definition:

```
bool EditPins();
```

Return:
true if pins have been edited

Arguments
none

4.12 Settings

4.12.1 ENABLEDRAGNDROP (CTR0108)

This function enables/disables all drag n drop features in the control.

C/C++/C# definition:

```
bool EnableDragnDrop(bool bEnable);
```

Return:
true if the command is supported by control.

Arguments
bEnable, true to enable drag n drop and false to disable

4.12.2 UNDOREDOSIZE (CTR0109)

This function changes the undo/redo stack size

C/C++/C# definition:

```
int UndoRedoSize(int iSize);
```

Return:
the new stack size (-1 if stack has not been changed).

Arguments
iSize, the new size to set

4.12.3 KEEPFBDSELECT (CTR0110)

If this property is set, when user puts a new item in control, the mode changes to select mode automatically.

C/C++/C# definition:

```
bool KeepFBDSelect(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
bSet, if true the select mode will be active after item insertion. If false, the insertion mode will be active after item insertion.

4.12.4 COPYBITMAP (CTR0111)

If this property is set, when control content is copied in clipboard, the bitmap of the copied part is saved in clipboard also.

C/C++/C# definition:

```
bool CopyBitmap(bool bSet);
```

Return:

returns true if this command is supported by control

Arguments
bSet, if true the bitmap is copied in clipboard also

4.12.5 PROMPTVARNAME (CTR0112)

If this property is set, when user inserts a new symbol in control:

1. If flag AUTOEDIT_SETVAR is activated, the control will display the declaration variable box.
2. If flag AUTOEDIT_SETVAR is not activated, the control will send a notification "W5EDITN_SETVAR" to its parent. The parent will have to declare a new symbol in database.
3.

C/C++/C# definition:

```
bool PromptVarname(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
bSet, if true the control will prompt to declare symbol in database

4.12.6 PROMPTINSTANCE (CTR0113)

If this property is set, when user inserts a new instance in control:

1. If flag AUTOEDIT_SETVAR is activated, the control will display the declaration variable box.
2. If flag AUTOEDIT_SETVAR is not activated, the control will send a notification "W5EDITN_SETVAR" to its parent. The parent will have to declare a new symbol in database.

C/C++/C# definition:

```
bool PromptInstance(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
bSet, if true the control will prompt to declare instance in database

Remarks
When this property is set to true, the property AutoDeclareInst cannot be set to true

4.12.7 ENABLEPULSE (CTR0114)

If this property is set, control allows to use the "pulse" style.
C/C++/C# definition:

```
bool EnablePulse(bool bSet);
```

Return:

returns true if this command is supported by control

Arguments
bEnable, if true enables the pulse style

4.12.8 SETVERTVARSIZE (CTR0115)

This function changes the default height of the variables.

C/C++/C# definition:

```
int SetVertVarSize(int iHeight);
```

Return:
returns the new default height for variables. 0 If command is not supported.

Arguments
iHeight, the new default height

4.12.9 SETHORZVARSIZE (CTR0116)

This function changes the default width of the variables.

C/C++/C# definition:

```
int SetHorzVarSize(int iWidth);
```

Return:
returns the new default width for variables. 0 If command is not supported.

Arguments
iWidth, the new default width

4.12.10 SETPAGEWIDTH (CTR0117)

This function changes the default page width (used for summary printing).

C/C++/C# definition:

```
int SetPageWidth(int iWidth);
```

Return:
returns the new page width

Arguments
iWidth, the new page width (in mm)

4.12.11 SETCONTENTS (CTR0118)

Not used…

4.12.12 AUTODECLAREINST (CTR0119)

This function enables/disables the auto-creation of a new instance in database when new instance is inserted in control.

C/C++/C# definition:

```
bool AutoDeclareInst(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true if want to autodeclare instance in database

Remarks
When this property is set to true, the property PromptInstance cannot be set to true

4.12.13 AUTODECLARESYMBOL (CTR0120)

This function enables/disables the auto-creation of a new variable in database when new variable is inserted in control.

C/C++/C# definition:

```
bool AutoDeclareSymbol(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true if want to autodeclare symbol in database

4.12.14 HIDEOPTIONDLGVAR (CTR0567)

This function enables to Show/Hide options in “SelectVar" dialog.

C/C++/C# definition:

```
bool HideOptionDlgVar(bool bHide);
```

Return:
returns true if this command is supported by control

Arguments
bHide, if true this hides the option in “variable selection” dialog box.

4.12.15 ENABLECOPY (CTR0121)

This function enables/disables copy feature in control.

C/C++/C# definition:

```
bool EnableCopy(bool bEnable);
```

Return:
returns true if this command is supported by control

Arguments
bEnable, if true this enables copy feature in control.

4.12.16 HIDESCROLL (CTR0122)

This function shows/hides control scroll bars.

C/C++/C# definition:

```
bool HideScroll(bool bHide);
```

Return:
returns true if this command is supported by control

Arguments
bHide, if true this hides the scroll bars.

4.12.17 SETTOOLTIPINFOS (CTR0123)

This function allows choosing the tooltip available information about a symbol in edit and debugging mode.

C/C++ definition:

```
bool SetTooltipInfos(BOOL bEdit, str_W5TooltipInfos* pStrTI);
```

C# definition:

```
bool SetTooltipInfos(bool bEdit, IntPtr pStrTI);
```

To use this function in C# please refer to Annexe “Using structures in C# wrappers”.
By the way, the best way of working is using “SetTooltipInfosEx” instead of “SetTooltipInfos”

Return:
returns true if this command is supported by control

Arguments
bEdit, if true specifies the tooltip for edit mode, if false specifies the tooltip for debug mode
pStrTP, the structure that specifies the tooltip content to display (see “W5EditApi.h” for more details about this structure)

4.12.18 SETTOOLTIPINFOSEX (CTR0583)

This function allows choosing the tooltip available information about a symbol in edit and debugging mode.

C/C++/C# definition:

```
bool SetTooltipInfos(bool bEdit, bool bSet, int iTooltipInformation);
```

Return:
returns true if this command is supported by control

Arguments
bEdit, if true specifies the tooltip for edit mode, if false specifies the tooltip for debug mode

bSet, if true the information will be displayed in tooltip, if false the information won't be displayed
iTooltipInformation: this is the information tooltip to show/hide:

Possible values are:

INFO_NAME	Show variable name
INFO_TYPE	Show variable type
INFO_DIM	Show variable dimensions
INFO_ATTRIB	Show variable attributes
INFO_PROPS	Show variable properties
INFO_SYB	Show symbol information
INFO_INITVALUE	Show variable initial value
INFO_TAG	Show variable tag
INFO_DESC	Show variable description

4.12.19 EDITPROPAFTERINSERTVAR (CTR0124)

This function enables/disables displaying the edit property box just after insert a new variable.

C/C++/C# definition:

```
bool EditPropAfterInsertVar(bool bInsert);
```

Return:
returns true if this command is supported by control

Arguments
bInsert, true want to display the property box

4.12.20 SETCHILDOFFSET (CTR0125)

This function set the offset of each child from its parent.

C/C++/C# definition:

```
int SetChildOffset(int iOffset);
```

Return:
returns the new specified child offset

Arguments
iOffset, the new child offset to set

4.12.21 INSERTVARAFTERINSERTFB (CTR0126)

This function is used to drag n drop variables during FB drag n drop from block list.

C/C++/C# definition:

```
bool InsertVarAfterInsertFB(bool bInsert);
```

Return:
returns true if this command is supported by control

Arguments

bDragWith, if true this allows to insert variable inputs and outputs with dragging Function Block.

4.12.22 DISPLAYIO (CTR0127)

This function is used to display embedded symbol in a different drawing method (using GDI_IO color and bold/italic)

C/C++/C# definition:

```
bool DisplayIO(bool bDisplay);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true if different drawing method must be used

4.12.23 SETAUTOEDIT (CTR0128)

This function allows changing the “autoedit” mode.
When an “autoedit” mode is active, the control is in charge to display its own dialog box.
When an “autoedit” mode is inactive, the control post a notification to its parent, and the prant is in charge to edit its own dialog box.

C/C++ definition:

```
DWORD SetAutoEdit(DWORD dwAutoEdit);
```

C# definition:

```
UInt32 SetAutoEdit(UInt32 dwAutoEdit);
```

Return:
returns the new value of AutoEdit flags.

Arguments
dwAutoEdit, is a combination of different flag values. See available values behind:

AUTOEDIT_DEFAULT	0x00000000	control will send all notifications and do not manage autoedit
AUTOEDIT_ALL	0xFFFFFFFF	control will manages all edits and won't send edit notifications
AUTOEDIT_SETVAR	0x00000001	for W5EDITN_SETVAR, W5EDITN_SETVARFIRSTCHAR and W5EDITN_COMPLETION
AUTOEDIT_SETFB	0x00000002	for W5EDITN_SETFB (use the "W5EditRes.dll")
AUTOEDIT_EDITPROPS	0x00000004	for W5EDITN_EDITPROPS (use the "W5EditRes.dll")
AUTOEDIT_INITARRAYVAR	0x00000008	for W5EDITN_INITARRAYVAR
AUTOEDIT_KEYWORD	0x00000010	for W5EDITN_LISTKEYWORD
AUTOEDIT_GRAPROPS	0x00000020	when double click on graphic item in W5EditACT.dll
AUTOEDIT_CONTROLDBG	0x00000040	for W5EDITN_CONTROLDBG

4.12.24 HIDEIONAME (CTR0129)

This function is used to hide/show the inputs and outputs names in extensible blocks.

C/C++/C# definition:

```
bool HideIOName(bool bHide);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true to hide IO names

4.12.25 UNDERLINEGLOBAL (CTR0130)

This function is used to enable/disable underline under global variables

C/C++/C# definition:

```
bool UnderlineGlobal(bool bUnderline);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true to underline global names in control

4.12.26 SETINFOS (CTR0131)

This function sets infos to control (infos have to be cast depending to dwInfo value).

C/C++ definition:

```
bool SetInfos(DWORD dwInfo, void* pInfos);
```

C# definition:

```
bool SetInfos(UInt32 dwInfo, IntPtr pInfos)
```

This function is not supported by C# wrapper!

Return:
returns true if this command is supported by control

Arguments
dwInfo, represents the type of structure passed in second argument
This can take value behind:

W5EDITINFOS_NONE	0
W5EDITINFOS_GRID	1
W5EDITINFOS_STLINENB	2

pInfos, a structure depending to the value of dwInfo (see “W5EditApi.h”).

W5EDITINFOS_NONE	none
W5EDITINFOS_GRID	str_W5InfosGrid
W5EDITINFOS_STLINENB	Boolean value (if >0 then display the line number, if 0 hide line numbers **in ST**)

4.12.27 GETINFOS (CTR0132)

This function gets informations from control (informations have to be cast depending to dwInfo value).

C/C++ definition:

```
bool GetInfos(DWORD dwInfo, void* pInfos);
```

C# definition:

```
bool GetInfos(UInt32 dwInfo, IntPtr pInfos);
```

This function is not supported by C# wrapper!

Return:
returns true if this command is supported by control

Arguments
dwInfo, represents the type of structure passed in second argument
This can take value behind:

W5EDITINFOS_NONE	0
W5EDITINFOS_GRID	1

pInfos, a structure depending to the value of dwInfo (see “W5EditApi.h”).

W5EDITINFOS_NONE	none
W5EDITINFOS_GRID	str_W5InfosGrid

4.12.28 SETBW (CTR0133)

This function is used to draw control in black and white.

C/C++/C# definition:

```
bool SetBW(bool bBackWhite);
```

Return:
returns true if this command is supported by control

Arguments
bSet, true to draw black & white, false to use color

4.12.29 SETCSVSEPARATORCOMA (CTR0577)

This function specifies the coma for CSV separator instead of semi colon.
C/C++/C# definition :

```
bool SetCsvSeparatorComa(bool bSetComa);
```

Return:
returns true if this command is supported by control

Arguments

bSetComa, true to specify coma as separator, false to specify semi colon as separator

Remarks:
By default the separator is semi colon (;) for csv files

4.12.30 AUTOREMOVEINST (CTR0588)

This function is used to delete unused instances of function blocks when saving the program.
C/C++/C# definition :

```
bool AutoRemoveInst(bool bAuto);
```

Return:
returns true if this command is supported by control

Arguments
`bAuto`, true to set the setting, false to unset the setting

4.12.31 USEDBLISTBOX (CTR0612)

This function is used to set the LISTBOX flag in variable dialog box instead of using the default tree organization (the aim of this feature is to decrease the open time of the dialog)
C/C++/C# definition :

```
bool UseDBListBox(bool bSet);
```

Return:
returns true if this command is supported by control

Arguments
`bSet`, true to set the setting, false to unset the setting

4.12.32 SETAUTORETYPEINST(CTR0613)

This function is used to retype instances in the database when the type in editor is not coherent with the type in database. For example using a type ALARM_M for inst1 in FBD and the type of inst1 is declared as “ALARM_A” in database -> the type of instance inst1 will be retyped in ALARM_M in database.
C/C++/C# definition :

```
bool SetAutoRetypeInst(int iSet);
```

Return:
returns true if this command is supported by control
Arguments
iSet, this indicates what control should do. This can take values below (see “W5EditApi.h”):

AUTORETYPE_UNSET	0	Do nothing (reset flag)
AUTORETYPE_SET	1	Flag is set. Now when saving, the database is updated
AUTORETYPE_NOW	2	Flag is not set, but the control update database now (not at saving)

4.12.33 SETDEFAULTFBWIDTH (CTR0615)

This function is used to set the default block width when user drags block from list blocks (using registered format "CF_K5FUNCBLOCK"). Parameter represents number of cell width.
If parameter is “-1”, then width is automatically calculated depending on block name and pins name to fit the best width.
C/C++/C# definition :

```
int SetDefaultFBWidth(int iWidth=-1);
```

Return:
returns new default width value (0 if command is not supported by control)
Arguments
iWidth, this indicates what width to set. -1 for automatic width.

4.13 Jpeg Print

4.13.1 PRINTSETPROPERTY (CTR0134)

This function sets a printing property.

C/C++ definition:

```
bool PrintSetProperty(int iProp, void* pValue);
```

C# definition:

```
bool PrintSetProperty(int iProp, IntPtr pValue);
```

Return:
returns true if property has been set

Arguments
iProp, this indicates which property has to be changed
This can take these values:

For print control only (“W5EditPrintApi.h” file):

PRINTPROP_STRUCT	0
PRINTPROP_PAGE	1
PRINTPROP_DOCUMENT	2
PRINTPROP_PROJECT	3
PRINTPROP_DOCUMENTDESCRIPTION	4
PRINTPROP_PROJECTDESCRIPTION	5
PRINTPROP_FOLIO	6
PRINTPROP_NPAGE	7
PRINTPROP_NBPAGE	8
PRINTPROP_DBID	9
custom property	>= 100

pValue, depends on value of iProp (it must be cast).
In case of Print control, cast are defined here:

PRINTPROP_STRUCT	str_W5Print

PRINTPROP_PAGE	const char*
PRINTPROP_DOCUMENT	const char*
PRINTPROP_PROJECT	const char*
PRINTPROP_DOCUMENTDESCRIPTION	const char*
PRINTPROP_PROJECTDESCRIPTION	const char*
PRINTPROP_FOLIO	const char*
PRINTPROP_NPAGE	const char*
PRINTPROP_NBPAGE	const char*
PRINTPROP_DBID	const char*
custom property	>= 100

See “W5EditPrintApi.h” file for description of str_W5Print structure.

Remarks:
In C# the value PRINTPROP_STRUCT can’t be used (Unable to declare the structure str_W5Print).

4.13.2 PRINTGETNBVERTFOLIO (CTR0135)

This function retrieves the number of vertical folios to print.

C/C++/C# definition:

```
int PrintGetNbVertFolio();
```

Return:
returns number of vertical folios to print

Arguments
none

4.13.3 PRINTGETNBHORZFOLIO (CTR0136)

This function retrieves the number of horizontal folios to print.

C/C++/C# definition:

```
int PrintGetNbHorzFolio();
```

Return:
returns number of horizontal folios to print

Arguments
none

4.13.4 PRINTGETFOLIO (CTR0137)

This function tests if folio contains items

C/C++/C# definition:

```
bool PrintGetFolio(int iX, int iY);
```

Return:

returns true if folio is not empty

Arguments
iX, the x coordinate of folio
iY, the y coordinate of folio

4.13.5 PRINTFOLIO (CTR0138)
This function prints jpeg of folio in file.

C/C++ definition:
```
void PrintFolio(int iX, int iY, LPCTSTR  szFile);
```

C# definition:
```
void PrintFolio(int iX, int iY, string szFile);
```

Return:
none

Arguments
iX, the x coordinate of folio
iY, the y coordinate of folio
szFile, the jpeg file path where folio will be printed

4.13.6 PRINTGETNBSYMBOLS (CTR0139)
This function gets number of symbols in folio.

C/C++/C# definition:
```
int PrintGetNbSymbols(int iX, int iY);
```

Return:
returns number of items in specified folio

Arguments
iX, the x coordinate of folio
iY, the y coordinate of folio

4.13.7 PRINTGETSYMBOLS (CTR0140)
This function gets list of symbols in folio (format: ID1,ID2,ID3….,IDn)

C/C++ definition:
```
LPCTSTR  PrintGetSymbols(int iX, int iY);
```

C# definition:
```
string PrintGetSymbols(int iX, int iY);
```

Return:
the list of item ID in folio (beware this are item ID, not database IDs).

Arguments

iX, the x coordinate of folio

iY, the y coordinate of folio

4.14 Draft print

4.14.1 SUMMARYPRINT (CTR0141)

This function prints the content of control in DC.

C/C++ definition:

```
bool SummaryPrint(BOOL bAll, HDC hDC);
```

C# definition:

```
bool SummaryPrint(bool bAll, IntPtr hDC);
```

Return:
returns true if something has been printed.

Arguments
bAll, true if print all content, false if print only selection
hDC, the Device Context handle where to print result

4.14.2 SETWIDTHSUMMARYPRINT (CTR0142)

This function sets the width of the rectangle to print.

C/C++/C# definition:

```
int SetWidthSummaryPrint(int iWidth);
```

Return:
returns new width

Arguments
iWidth, the new width to set

4.14.3 SETHEIGHTSUMMARYPRINT (CTR0143)

This function sets the height of the rectangle to print.

C/C++/C# definition:

```
int SetHeightSummaryPrint(int iHeight);
```

Return:
returns new height

Arguments
iHeight, the new height to set

4.14.4 CANPRINT (CTR0144)

This function tests if control can be printed.

C/C++/C# definition:

```
bool CanPrint();
```

Return:
returns true if control supports print feature.

Arguments
none

4.14.5 ISPRINTTEXT (CTR0145)

This function tests if control will be printed as a text

C/C++/C# definition:

```
bool IsPrintText();
```

Return:
returns true is printed as text.

Arguments
none

4.14.6 ISPRINTGRAPHIC (CTR0146)

This function tests if control will be printed graphically.

C/C++/C# definition:

```
bool IsPrintGraphic();
```

Return:
returns true is printed graphically.

Arguments
none

4.15 Extended print Folios

4.15.1 SETPRINTER (CTR0147)

This function initializes the print parameters. This have to be used before calling PrintFolioEx, GetNbVertFolioEx and GetNbHorzFolioEx.

C/C++/C# definition:

```
bool SetPrinter(int iDPIX, int iDPIY, int iFolioWidth, int iFolioHeight);
```

Return:
returns true if this command is supported by control

Arguments
iDPIX, is the horizontal DPI of the printer
iDPIY, is the vertical DPI of the printer
iFolioWidth, is the horizontal size of the printed rectangle in printer DC (in pixel)
iFolioHeight, is the vertical size of the printed rectangle in printer DC (in pixel)

4.15.2 GETNBVERTFOLIOEX (CTR0148)

This function retrieves the total number of vertical folios.

C/C++/C# definition:

```
int GetNbVertFolioEx();
```

Return:
returns the number of vertical folios

Arguments
none

Remark
To obtain the full number of folio to print, you will have to multiply result of GetNbVertFolioEx and GetNbHorzFolioEx.

4.15.3 GETNBHORZFOLIOEX (CTR0149)

This function retrieves the total number of horizontal folios.

C/C++/C# definition:

```
int GetNbHorzFolioEx();
```

Return:
returns the number of horizontal folios

Arguments
none

Remark

To obtain the full number of folio to print, you will have to multiply result of GetNbVertFolioEx and GetNbHorzFolioEx.

4.15.4 PRINTFOLIOEX (CTR0150)

This function prints one folio from control at the given position in the specified Device Context.

C/C++ definition:

```
LPCRECT PrintFolioEx(int iXFolio, int iYFolio, HDC hDC, int iLeftPos, int iTopPos);
```

C# definition:

```
IntPtr PrintFolioEx(int iFolioX, int iFolioY, IntPtr hDC, int iLeftPos, int iTopPos);
```

Return:
returns a const pointer to a RECT structure. Do not delete the returned pointer!

Arguments
iXFolio, the x coordinates of the folio (begins at 0)
iYFolio, the y coordinates of the folio (begins at 0)
hDC, the Device Context of the printer
iLeftPos, the x coordinate of the print area in Device Context
iTopPos, the y coordinate of the print are in Device Context

4.15.5 GETFOLIORECT (CTR0151)

This function is used to get logical coordinates of the selected folio.

C/C++ definition:

```
LPCRECT GetFolioRect(int iFolioX, int iFolioY);
```

C# definition:

```
IntPtr GetFolioRect(int iFolioX, int iFolioY);
```

Return:
returns a const pointer to a RECT structure. Do not delete the returned pointer!

Arguments
iXFolio, the x coordinates of the folio (begins at 0)
iYFolio, the y coordinates of the folio (begins at 0)

4.15.6 GETHEADERXY (CTR0152)

This function is used to get header coordinates (in text format LetterNumber) at the specified relative location.

C/C++ definition:

```
LPCTSTR  GetHeaderXY(int iX, int iY, LPCRECT rctLog);
```

C# definition:

```
string GetHeaderXY(int iX, int iY, IntPtr rctLog);
```

Return:
returns the coordinates of the specified point in control. The format of the return text is like: LetterNumber (ex.: B6)

Arguments
iX, the x coordinates of the point to convert
iY, the y coordinates of the point to convert
rctLog, a const RECT structure this represents the folio rectangle returned by GetFolioRect

4.15.7 GETSYMBOLINFOLIO (CTR0153)

This function gets list of symbols in folio (format: ID1,ID2,ID3….,IDn)

C/C++ definition:

```
LPCTSTR  GetSymbolInFolio(int iX, int iY);
```

C# definition:

```
string GetSymbolInFolio(int iX, int iY);
```

Return:
returns the list of item ID in folio (beware this are item ID, not database IDs).

Arguments
iX, the x coordinates of the folio (begins at 0)
iY, the y coordinates of the folio (begins at 0)

4.15.8 ADJUSTFOLIO (CTR0154)

This function assumes that print is stretch to not exceed max number of vertical and horizontal folios.

C/C++/C# definition:

```
BOOL AdjustFolio(int iMaxFolioX, int iMaxFolioY);
```

Return:
returns false if print can not adjust to the specified size. true if control can adjust to max FolioX and max Folio Y.

Arguments
iMaxFolioX, allows to specify maximum number of horizontal folio to print
iMaxFolioY, allows to specify maximum number of vertical folio to print

4.15.9 DISPLAYFOLIO (CTR0155)

This function shows/hides folio area in control.

C/C++/C# definition:

```
void DisplayFolio(BOOL bDisplay);
```

Return:
none

Arguments
bDisplay, true to display the folio lines in control.

4.15.10 ISDISPLAYFOLIO (CTR0156)

This function tests if folios are displayed in control.

C/C++/C# definition:

```
BOOL IsDisplayFolio();
```

Return:
true if folio lines are displayed in control

Arguments
none

4.15.11 FORCEPRINTZOOM (CTR0566)

This function forces the printing zoom (don't care of control zoom during printing).

C/C++/C# definition:

```
bool ForcePrintZoom(bool bForce, int iZoom);
```

Return:
true if the control supports this feature

Arguments
bForce, true if want to force zoom value (iZoom must be ranged in 0 – 100)
iZoom, the zoom ratio to force. This is a percentage (0-100). Don’t care if bForce is false.

4.16 Dictionary

4.16.1 FOCUSVAR (CTR0157)

This function selects and focuses the specified variable in specified group.

C/C++ definition:

```
bool FocusVar(LPCTSTR  szGroup, LPCTSTR  szVar);
```

C# definition:

```
bool FocusVar(string szGroup, string szVar);
```

Return:
true if variable has been found

Arguments
szGroup, the group name where to search variable
szVar, the variable to search

4.16.2 FOCUSGROUP (CTR0158)

This function selects and focuses the specified group.

C/C++ definition:

```
void FocusGroup(LPCTSTR  szGroup);
```

C# definition:

```
void FocusGroup(string szGroup);
```

Return:
none

Arguments
szGroup, the group name to select

4.16.3 FILTERGROUP (CTR0159)

This function filters the specified list of groups (all other groups are hidden).

C/C++ definition:

```
void FilterGroup(LPCTSTR  szGroupList);
```

C# definition:

```
void FilterGroup(string szGroupList);
```

Return:
none

Arguments
szGroupList, the list of group to display. This is a list of group IDs separate by comas (format: ID1,ID2,ID3….,IDn).

4.16.4 CANFILTER (CTR0574)

This function tests if control support the specified filter value

C/C++ definition:

```
bool CanFilter(DWORD dwFilter);
```

C# definition:

```
bool CanFilter(UInt32 dwFilter);
```

Return:

Bool, true if filter can be apply to the control

Arguments

For dictionary control, “dwFilter” this is a combination of values:

TLFILTER_ONLYSTRUCTURE	0x0001	dictionary only displays structures
TLFILTER_HIDESTRUCT	0x0002	dictionary hides structures
TLFILTER_HIDEINSTANCE	0x0004	hide the instances of FB and UDFB in dictionary
TLFILTER_ONLYGLOBAL	0x0008	shows only global groups
TLFILTER_ONLYSINGLETYPE	0x0010	shows only single types in dictionary
TLFILTER_HIDERETAIN	0x0020	hides the group of RETAIN variables

4.16.5 SETFILTER (CTR0160)

This function filters the control content.

C/C++ definition:

```
void SetFilter(DWORD dwFilter);
```

C# definition:

```
void SetFilter(UInt32 dwFilter);
```

Return:

none

Arguments

For dictionary control, “dwFilter” this is a combination of values:

TLFILTER_ONLYSTRUCTURE	0x0001	dictionary only displays structures
TLFILTER_HIDESTRUCT	0x0002	dictionary hides structures
TLFILTER_HIDEINSTANCE	0x0004	hide the instances of FB and UDFB in dictionary
TLFILTER_ONLYGLOBAL	0x0008	shows only global groups
TLFILTER_ONLYSINGLETYPE	0x0010	shows only single types in dictionary
TLFILTER_HIDERETAIN	0x0020	hides the group of RETAIN variables

4.16.6 GETSELGROUPNAME (CTR0161)

This function gets the selected group name if any.

C/C++ definition:
```
LPCTSTR  GetSelGroupName();
```

C# definition:
```
string GetSelGroupName();
```

Return:
returns the selected group name

Arguments
none

4.16.7 GETGROUPID (CTR0162)
This function gets the selected group database ID if any.

C/C++ definition:
```
DWORD GetGroupID();
```

C# definition:
```
UInt32 GetGroupID();
```

Return:
returns the selected group database ID

Arguments
none

4.16.8 LOADEXPAND (CTR0163)
This function loads the expanded groups from text file.

C/C++ definition:
```
void LoadExpand(LPCTSTR  szPath);
```

C# definition:
```
void LoadExpand(string szPath);
```

Return:
none

Arguments
szPath, the path from where to load expand serialization

4.16.9 SAVEEXPAND (CTR0164)
This function saves the expanded group to text file.

C/C++ definition:
```
void SaveExpand(LPCTSTR  szPath);
```

C# definition:

```
void SaveExpand(string szPath);
```

Return:
none

Arguments
szPath, the path where to save expand serialization

4.16.10 CANSWAPGLOBALRET (CTR0165)

This function tests if selected variable can be swap global/retain.

C/C++/C# definition:

```
bool CanSwapGlobalRet();
```

Return
true if selection can be swap between global and retain group

Arguments
none

4.16.11 SWAPGLOBALRET (CTR0166)

This function swaps global/retain the selected variable.

C/C++/C# definition:

```
void SwapGlobalRet();
```

Return:
none

Arguments
none

4.16.12 GETSERIALCOL (CTR0167)

This function is used to serialize shown columns and sizes.

C/C++ definition:

```
LPCTSTR  GetSerialCol(void);
```

C# definition:

```
string GetSerialCol();
```

Return:
the column serialization string

Arguments
none

4.16.13 SETSERIALCOL (CTR0168)

This function is used to load shown columns and column sizes.

C/C++ definition:

```
void SetSerialCol(LPCTSTR  szSerial);
```

C# definition:

```
void SetSerialCol(string szSerial);
```

Return:
none

Arguments
szSerial, the serialization string to set

4.16.14 GETSERIALEXPAND (CTR0169)

This function is used to serialize expanded items in tree.

C/C++ definition:

```
LPCTSTR  GetSerialExpand();
```

C# definition:

```
string GetSerialExpand();
```

Return:
the serialization of the item expand status

Arguments
none

4.16.15 SETSERIALEXPAND (CTR0170)

This function is used to expand items in tree.

C/C++ definition:

```
void SetSerialExpand(LPCTSTR  szSerial);
```

C# definition:

```
void SetSerialExpand(string szSerial);
```

Return:
none

Arguments
szSerial, applies expand serialization to the control

4.16.16 SETLOCALSEL (CTR0171)

This function sets the preferred group, dictionary is sorted with the preferred group in first.

C/C++ definition:
```
void SetLocalSel(DWORD dwGroup);
```

C# definition:
```
void SetLocalSel(UInt32 dwGroup);
```

Return:
none

Arguments
dwGroup, the contextual group database ID that will be display in first position in dictionary

4.16.17 GETEDITMODE (CTR0172)

This function gets the edition mode. See SetEditMode feature.

C/C++/C# definition:
```
bool GetEditMode() ;
```

Return:
true if control is in edit mode

Arguments
none

4.16.18 SETEDITMODE (CTR0173)

This function sets the edition mode. The edit mode is the one cell mode (when user double click in cell, it will display a “edit in place” dialog). When edit mode is inactive, full line can be selected and when user double clicks on variable dictionary display a dialog box to enter all variables properties in one time.

C/C++/C# definition:
```
void SetEditMode(bool bEdit);
```

Return:
none

Arguments
bEdit, true to activate edit mode

4.16.19 GETSELVARID (CTR0174)

This function gets the selected variable base ID.

C/C++ definition:
```
DWORD GetSelVarID();
```

C# definition:
```
UInt32 GetSelVarID();
```

Return:
returns the database ID of the selected variable (if one selected)

Arguments
none

4.16.20 ISDICOENABLE (CTR0175)

This function tests if dictionary is enable.

C/C++/C# definition:

```
bool IsDicoEnable(bool bWithSelVar, bool bWithSelGrp, bool bWithReadOnly);
```

Return:
returns true if the selection is enabled

Arguments
bWithSelVar, true if feature must be tested with a variable selection
bWithSelGrp, true if feature must be tested with a group selection
bWithReadOnly, true if the feature must be tested with read only status

4.16.21 ARRANGECOLUMNS (CTR0176)

This function calls dialog box for rearrange columns order.

C/C++/C# definition:

```
void ArrangeColumns();
```

Return:
none

Arguments
none

4.16.22 CANINSERTSTRUCTURE (CTR0177)

This function tests if structure can be inserted at current position.

C/C++/C# definition:

```
bool CanInsertStructure();
```

Return:
returns true if structure can be inserted at current position

Arguments
none

4.16.23 INSERTSTRUCTURE (CTR0178)

This function inserts structure at current position.

C/C++/C# definition:

```
void InsertStructure();
```

Return:
none

Arguments
none

4.16.24 CANMOVESTRUCTURE (CTR0179)

This function is obsolete.

4.16.25 MOVESTRUCTURE (CTR0180)

This function is obsolete.

4.16.26 EDITSTRUCTURE (CTR0181)

This function edits the current selected structure.

C/C++/C# definition:

```
bool EditStructure();
```

Return:
returns true if selected structure has been edited

Arguments
none

4.16.27 CANEDIT (CTR0182)

This function tests if variable can be edited.

C/C++/C# definition:

```
bool CanEdit();
```

Return:
returns true if selection can be edited

Arguments
none

4.16.28 GETSELVARNAME (CTR0183)

This function gets the selected var name.

C/C++ definition:

```
LPCTSTR  GetSelVarName();
```

C# definition:

```
string GetSelVarName();
```

Return:
return the name of the selected variable (if any)

in C++, if no variable is selected, this function returns null
in C#, if not variable selected, this function returns “”

Arguments
none

4.16.29 HIDEARRANGECOL (CTR0184)

This function hides the “to display rearrange column” button.

C/C++/C# definition:

```
void HideArrangeCol();
```

Return:
none

Arguments
none

4.16.30 SETCALLBACK (CTR0185)

This function sets one of callback functions.

C/C++ definition:

```
bool SetCallback(int iCallBack, void* pfCB, void* pData);
```

C# definition:

```
bool SetCallback(int iCallBack, IntPtr pfCB, IntPtr pData);
```

This function is not supported by C# wrapper!

Return:
true if the call back function has been set

Arguments
iCallback, the reference of the call back function.
Can be one of these values:

W5CB_NONE	0	
W5CB_CANCLEARVAR	1	this call back is called just before deleting a variable
W5CB_CLEARVARAFTERCANCELEDIT	2	this call back is called when user cancel edit on a variable
W5CB_CANMOVEVAR	3	this call back is called just before moving a variable
W5CB_DROPFB	4	this call back is called just before insert a new Function Block
W5CB_COMPARETREEITEM	5	this call back is called to compare two items during SortChildrenCB
W5CB_CHECKCELLVALUE	6	this call back is used to test if value is allowed in treelist item
W5CB_CANCHANGEVAR	7	this call back is used to test if var can be modified by user
W5CB_GETGROUPIMAGE	8	this call back is used to get a customized image index in dictionary (should be used in association with “SetTreeIcon” command)
W5CB_GETFILTEREDTYPES	9	This call back is used to get the available type list in a group (used

		only in dictionary)

pfCB, this is function pointer. Its type depends on the value of iCallback:

W5CB_NONE	
W5CB_CANCLEARVAR	bool (*K5CANCLEARVAR)(HWND hWnd, unsigned long dwVar, void* pData);
W5CB_CLEARVARAFTERCANCELEDIT	bool (*K5CLEARVARAFTEREDITCANCEL) (HWND hWnd, unsigned long dwVar, void* pData);
W5CB_CANMOVEVAR	bool (*K5CANMOVEVAR) (HWND hWnd, unsigned long dwVar, DWORD dwGroupFrom, DWORD dwGroupTo, void* pData);
W5CB_DROPFB	bool (*K5DROPFB)(HWND hWnd, LPCSTR szFB, void* pData);
W5CB_COMPARETREEITEM	int (*K5COMPARETREEITEM)(DWORD_PTR dwData1, DWORD_PTR dwData2);
W5CB_CHECKCELLVALUE	bool (*K5CHECKCELLVALUE)(HWND hWnd, DWORD_PTR hItem, int iCol, LPCSTR szValue, void* pData, BOOL *pbQuiet);
W5CB_CANCHANGEVAR	bool (*K5CANCHANGEVAR)(HWND hWnd, unsigned long dwVar, unsigned long dwInfoType, void* pData);
W5CB_GETGROUPIMAGE	int (*K5GETGROUPIMAGE)(LPCSTR szProjectPath, LPCSTR szGroup, void* pData);
W5CB_GETFILTEREDTYPES	LPCSTR (*K5GETFILTEREDTYPES) (DWORD dwGroup, void *pUserData) The callbcak returns list of supported types with this syntax : « Type1,Type2, Type3… »

pData, this is a user data called when call back is called

4.16.31 CANFINDVARIABLES (CTR0186)

This function tests if “FindVariable" feature is supported.

C/C++/C# definition:

```
bool CanFindVariables();
```

Return:

true if the “FindVariables” feature is supported

Arguments

none

4.16.32 FINDVARIABLES (CTR0187)

This function displays dialog to search a set of variables.

C/C++/C# definition:

```
int FindVariables();
```

Return:

retuns the number of variables found by the feature

Arguments

none

4.16.33 CANRENAMEVARIABLES (CTR0188)

This function tests if RenameVariable" feature is supported.

C/C++/C# definition:

```
bool CanRenameVariables();
```

Return:

this function returns true if feature “RenameVariables” is supported

Arguments

none

4.16.34 RENAMEVARIABLES (CTR0189)

This function displays dialog box to rename a set of variables.

C/C++/C# definition:

```
int RenameVariables();
```

Return:

retuns the number of variables renamed by the feature

Arguments

none

4.16.35 CANFORCEINITVALUE (CTR0190)

This function tests if user can force variable initial value during debug.

C/C++/C# definition:

```
bool CanForceInitValue();
```

Return:

true if variable init value can be changed during debug

Arguments

none

4.16.36 FORCEINITVALUE (CTR0191)

This function forces variable initial value during debug.

C/C++/C# definition:

```
void ForceInitValue();
```

Return:

none

Arguments

none

4.16.37 CANGROUPVAR (CTR0578)

This function tests if variable can be inserted in subgroup.

C/C++/C# definition:

```
bool CanGroupVar();
```

Return:

Returns true if selected variable can be put in a global sub-group

Arguments

none

4.16.38 GROUPVAR (CTR0579)

This function puts the selected variable(s) in a sub group or remove variable(s) from their sub-groups.

C/C++/C# definition:

```
bool GroupVar(bGroup);
```

Return:

Returns true if at least one variable has been put in a global sub-group.

Arguments

bGroup, if true, displays a dialog box to choose a sub-group name. If false, the selected variable(s) are removed from the sub-group and go to the global group.

4.17 ST control

4.17.1 CANSETTAB (CTR0192)

This function tests if can change tabulation size.

C/C++/C# definition:

```
bool CanSetTab();
```

Return:

returns true if tab can be set

Arguments

none

4.17.2 SETTAB (CTR0193)

This function sets tabulation size.

C/C++/C# definition:

```
int SetTab(int iTab);
```

Return:

returns the new value of tab

Arguments

iTab, the new value to set

4.17.3 SETSYNTAXCOLORING (CTR0194)

This function sets a new syntax to be colored in "keyword" color.

C/C++ definition:

```
bool SetSyntaxColoring(LPCTSTR  szSyntax);
```

C# definition:

```
bool SetSyntaxColoring(string szSyntax);
```

Return:

returns true if command is supported by control

Arguments

szSyntax, the new syntax to set (a list of keywords separate by comas)

4.17.4 ENABLESYNTAX (CTR0195)

This function enables/disables syntax coloring.

C/C++/C# definition:

```
bool EnableSyntax(bool bEnable);
```

Return:

returns true if command is supported by control

Arguments
bEnable, true to enable the syntax coloring, if false all text is displayed black

4.17.5 SETVALUEINTEXT (CTR0196)

This function enables/disables the debug tag in ST.

C/C++/C# definition:

```
void SetValueInText(bool bSet);
```

Return:
none

Arguments
bSet, true to display debug value in text during debug

4.17.6 GETVALUEINTEXT (CTR0197)

This function gets the state (enable/disable) of the debug tag in ST.

C/C++/C# definition:

```
int GetValueInText();
```

Return:
returns 1 if value are displayed in text during debug, 0 otherwise

Arguments
none

4.17.7 CANREMOVECOMMENT (CTR0198)

This function tests if comments can be deleted from selection.

C/C++/C# definition:

```
bool CanRemoveComment();
```

Return:
true if comments can be deleted

Arguments
none

4.17.8 REMOVECOMMENT (CTR0199)

This function removes comments from selection.

C/C++/C# definition:

```
void RemoveComment();
```

Return:

none

Arguments
none

4.17.9 SETBGCOLOR (CTR0200)

This function sets back color.

C/C++ definition:

```
bool SetBGColor(COLORREF rgbMain, COLORREF rgbUsed);
```

C# definition:

```
bool SetBGColor(IntPtr rgbMain, IntPtr rgbUsed);
```

Return:
true if back color has been set

Arguments
rgbMain, the background color (RGB convention) of the ST sections
rgbUsed, the background color (RGB convention) of the IL sections

4.17.10 SETPROMPT (CTR0201)

This function sets control in prompt mode, caret cannot be moved before prompt line.

C/C++ definition:

```
long SetPrompt(BOOL bAppend, LPCTSTR  szPrompt);
```

C# definition:

```
Int32 SetPrompt(bool bAppend, string szPrompt);
```

Return:
returns the current coordinates of the carte in a int32.
x coordinates can be retrieved in low word
y coordinate can be retrieved in high word

Arguments
bAppend, true to add lines to old text and set in prompt mode the old text
szPrompt, specifies the text to display in the prompt mode

4.17.11 GETCOMMANDLINE (CTR0202)

This function retrieves the command line after the prompt.

C/C++ definition:

```
LPCTSTR  GetCommandLine();
```

C# definition:

```
string GetCommandLine();
```

Return:
returns the command line

Arguments
none

4.17.12 AUTOCOMPLETE (CTR0203)

This function autocompletes the selected word. If flag AUTOEDIT_SETVAR (see command SetAutoEdit) is not set, the control sends notification “W5EDITN_SETVAR”.

C/C++/C# definition:

```
void AutoComplete();
```

Return:
none

Arguments
none

4.17.13 CANINDENT (CTR0570)

This function tests if selected text can be indented.

C/C++/C# definition:

```
bool CanIndent();
```

Return:
Returns true if text can be indented

Arguments
none

4.17.14 INDENT (CTR0571)

This function indents the selected ST text (using the current tab length value)

C/C++/C# definition:

```
bool Indent();
```

Return:
Returns true if text has been indented

Arguments
none

4.17.15 GETSTKEYWORDS (CTR0592)

This function retrieves:

- the list of keywords supported by ST control (like THEN, WHILE…)
- list of functions and operator from straton registry.

- List of sub-programs in straton database

C/C++ definition:

```
LPCTSTR  GetSTKeywords();
```

C# definition:

```
string GetSTKeywords();
```

Return:
Returns list of keywords (“\n” is the separator between keywords)

Arguments
none

4.18 FBD control

4.18.1 ISCHECK (CTR0204)

This function tests if specified mode is used.

C/C++/C# definition:

```
bool IsCheck(int iMode);
```

Return:

returns true if the specified mode is active

Arguments

iMode the mode to test.

In FBD can be one of these values:

FBD_SELECT	0	Selection mode, used to drag items, select multiple items…
FBD_ADD_FB	1	When active, if user drag n drop, it will be added a block at mouse position
FBD_ADD_VAR	2	When active, if user drag n drop, it will be added a variable at mouse position
FBD_ADD_COMMENT	3	When active, if user drag n drop, it will be added a comment at mouse position
FBD_ADD_ARC	4	When active, let the user draw arcs / line between other items
FBD_ADD_CORNER	5	When active, if user drag n drop, it will be added a corner at mouse position
FBD_ADD_BREAK	6	When active, if user drag n drop, it will be added a break at mouse position
FBD_ADD_LABEL	7	When active, if user drag n drop, it will be added a label at mouse position
FBD_ADD_JUMP	8	When active, if user drag n drop, it will be added a jump at mouse position
FBD_ADD_LEFTRAIL	9	When active, if user drag n drop, it will be added a left rail at mouse position
FBD_ADD_CONTACT	10	When active, if user drag n drop, it will be added a contact at mouse position
FBD_ADD_OR	11	When active, if user drag n drop, it will be added a OR vertical bar at mouse position
FBD_ADD_COIL	12	When active, if user drag n drop, it will be added a coil at mouse position
FBD_ADD_RIGHTRAIL	13	When active, if user drag n drop, it will be added a right rail at mouse position
FBD_PASTE	14	When active, the previous copied selection can be paste at mouse position
FBD_ADDRULE	15	When active, if user drag n drop, it will be added a rule (vertical line) at mouse position

In FFSFC and in SAMA, can be one of these values:

MODE_SELECT	0	Selection mode, used to drag items, select multiple items…
MODE_PASTE	1	When active, the previous copied selection can be paste at mouse position
MODE_ARC	2	When active, let the user draw arcs / line between other items
MODE_FB	3	When active, if user drag n drop, it will be added a block at mouse position
MODE_VAR	4	When active, if user drag n drop, it will be added a variable at mouse position
MODE_COMMENT	5	When active, if user drag n drop, it will be added a comment at mouse position
MODE_CORNER	6	When active, if user drag n drop, it will be added a corner at mouse position
MODE_BREAK	7	When active, if user drag n drop, it will be added a break at mouse position
MODE_LABEL	8	When active, if user drag n drop, it will be added a label at mouse position
MODE_JUMP	9	When active, if user drag n drop, it will be added a jump at mouse position
MODE_CONTACT	10	When active, if user drag n drop, it will be added a contact at mouse position
MODE_OR	11	When active, if user drag n drop, it will be added a OR vertical bar at mouse position

MODE_COIL	12	When active, if user drag n drop, it will be added a coil at mouse position
MODE_LEFTRAIL	13	When active, if user drag n drop, it will be added a left rail at mouse position
MODE_RIGHTRAIL	14	When active, if user drag n drop, it will be added a right rail at mouse position
MODE_ISTEP	15	When active, if user drag n drop, it will be added an Initial step at mouse position
MODE_STEP	16	When active, if user drag n drop, it will be added an step at mouse position
MODE_TRANS	17	When active, if user drag n drop, it will be added an transition at mouse position
MODE_MACRO	18	When active, if user drag n drop, it will be added an macro at mouse position
MODE_INOUT	19	When active, if user drag n drop, it will be added an in/out data at mouse position

In ACTX (graphics), can be one of these values:

ACT_SELECT	0	Selection mode, used to drag items, select multiple items…
ACT_PASTE	1	When active, the previous copied selection can be paste at mouse position

In FFLD (Free Form Logic Diagram) can be one of these values:

FFLD_SELECT	0	Selection mode, used to drag items, select multiple items…
FFLD_CONTACT	1	When active, if user drag n drop, it will be added a contact at mouse position
FFLD_CONTACT_INVERT	2	When active, if user drag n drop, it will be added a invert contact at mouse position
FFLD_CONTACT_P	3	When active, if user drag n drop, it will be added a pulse contact at mouse position
FFLD_CONTACT_PINVERT	4	When active, if user drag n drop, it will be added a pulse invert contact at mouse position
FFLD_CONTACT_N	5	When active, if user drag n drop, it will be added a N contact at mouse position
FFLD_CONTACT_NINVERT	6	When active, if user drag n drop, it will be added a invert N contact at mouse position
FFLD_COIL	7	When active, if user drag n drop, it will be added a coil at mouse position
FFLD_COIL_INVERT	8	When active, if user drag n drop, it will be added a invert coil at mouse position
FFLD_COIL_S	9	When active, if user drag n drop, it will be added a S coil at mouse position
FFLD_COIL_R	10	When active, if user drag n drop, it will be added a R coil at mouse position
FFLD_LINE_HORZ	11	When active, if user drag n drop, it will be added a horizontal line at mouse position
FFLD_LINE_VERT	12	When active, if user drag n drop, it will be added a vertical line at mouse position
FFLD_LINE_COMBO	13	When active, let the user draw arcs / line between other items
FFLD_FB	14	When active, if user drag n drop, it will be added a block at mouse position
FFLD_DATAIN	15	When active, if user drag n drop, it will be added a DataIn item at mouse position
FFLD_DATAIN_INVERT	16	When active, if user drag n drop, it will be added a invert DataIn item at mouse position
FFLD_DATAOUT	17	When active, if user drag n drop, it will be added a DataOut item at mouse position
FFLD_JUMP	18	When active, if user drag n drop, it will be added a jump at mouse position
FFLD_RETURN	19	When active, if user drag n drop, it will be added a return at mouse position
FFLD_NETWORK	20	When active, if user drag n drop, it will be added a jump at mouse position
FFLD_ROW	21	When active, if user drag n drop, it will be added an empty row at mouse position
FFLD_COMMENT	22	When active, if user drag n drop, it will be added a comment at mouse position
FFLD_LABEL	23	When active, if user drag n drop, it will be added a label at mouse position
FFLD_PRAGMA	24	When active, if user drag n drop, it will be added a pragma at mouse position
FFLD_LINE_POINT	25	When active, if user drag n drop, it will be added a point line at mouse position
FFLD_COIL_P	26	When active, if user drag n drop, it will be added a P coil at mouse position

FFLD_COIL_N	27	When active, if user drag n drop, it will be added a N coil at mouse position

4.18.2 CANCHECK (CTR0205)

This function tests if specified mode can be used.

C/C++/C# definition:

```
bool CanCheck(int iMode);
```

Return:
true if mode can be set

Arguments
iMode, for list of values, see “IsCheck” command.

See also
IsCheck, Check

4.18.3 CHECK (CTR0206)

This function sets the specified mode.

C/C++/C# definition:

```
void Check(int iMode);
```

Return:
none

Arguments
iMode, for list of values, see “IsCheck” command.

See also
IsCheck, CanCheck

4.18.4 CANDISPLAYFBDORDER (CTR0207)

This function tests if FBD order can be displayed.

C/C++/C# definition:

```
bool CanDisplayFBDOrder();
```

Return:
true if FBD order can be displayed in FBD order.

Arguments
none

4.18.5 DISPLAYFBDORDER (CTR0208)

This function shows the FBD order and return the number of item ordered.

C/C++ definition:

```
DWORD DisplayFBDOrder(void);
```

C# definition:

```
UInt32 DisplayFBDOrder();
```

Return:
the number or items ordered

Arguments
none

4.18.6 CANCREATEUDFB (CTR0209)

This function tests if User Defined Function Block can be created with selected code.

C/C++/C# definition:

```
bool CanCreateUDFB();
```

Return:
returns true if control can create a new UDFB with selected code

Arguments
none

4.18.7 CREATEUDFB (CTR0210)

This function creates User Defined Function Block with selected code.

C/C++/C# definition:

```
bool CreateUDFB();
```

Return:
returns true if control has created a new UDFB with selected source code

Arguments
none

4.18.8 CANCONNECT (CTR0211)

This function tests if two items can be connected using the “connection” dialog box.

C/C++/C# definition:

```
bool CanConnect();
```

Return:
true if can display connection dialog box

Arguments
none

4.18.9 CONNECT (CTR0212)

This function connects two items using the “connection” dialog box.

C/C++/C# definition:

```
bool Connect();
```

Return:

true if dialog box has been displayed

Arguments

none

4.18.10 CANROTATECORNERS (CTR0213)

This function tests if selected corners can be rotated.

C/C++/C# definition:

```
bool CanRotateCorners();
```

Return:

true if selection of corners can be rotate

Arguments

none

4.18.11 ROTATECORNERS (CTR0214)

This function uses symmetry to rotate corners.

C/C++/C# definition:

```
void RotateCorners(int iSym);
```

Return:

none

Arguments

iSym, determinates the symmetry axis

Can take one of these values:

SYM_VERT	1
SYM_HORZ	2

4.18.12 SETAUTOCONNECTVARIABLE (CTR0215)

This function sets the option to auto connect variables when drop near a function block input or output.

C/C++/C# definition:

```
bool SetAutoConnectVariable(bool bAuto);
```

Return:

true if command is supported by control

Arguments
bAuto, when true variables dropped near an input or output block will be automatically connected to this block

4.18.13 SETAUTOREMOVEVARIABLE (CTR0605)

This function sets the option to auto remove variables when they are connected to a function block that user is deleting.

C/C++/C# definition:

```
bool SetAutoRemoveVariable(bool bAuto);
```

Return:
true if command is supported by control

Arguments
bAuto, when true variables connected to block will be removed. NB: this does not remove variables from database.

4.18.14 SNAPFBDGRID (CTR0216)

This function snaps arcs to grid.

C/C++/C# definition:

```
bool SnapFBDGrid(bool bSnap);
```

Return:
true if command is supported by control

Arguments
bSnap, true if new lines have to be snap to grid step

4.18.15 DRAWFBDBRIDGE (CTR0217)

This function draws bridge when two arcs intersects.

C/C++/C# definition:

```
bool DrawFBDBridge(bool bDraw);
```

Return:
true if command is supported by control

Arguments
bDraw, true to draw little bridge when two unconnected lines cross

4.18.16 CANSHOWSPY (CTR0218)

This function tests if a spy box can be displayed near the selected item.

C/C++/C# definition:

```
bool CanShowSpy();
```

Return:
true if a spy box can be added near the selection
Arguments
none

4.18.17 ISSPYVISIBLE (CTR0219)

This function tests if a spy box is displayed near the selected item.

C/C++/C# definition:

```
bool IsSpyVisible();
```

Return:
true if spy box is visible near the selection

Arguments
none

4.18.18 SHOWSPY (CTR0220)

This function shows/hides the spy box near the selected item.

C/C++/C# definition:

```
void ShowSpy(bool bShow);
```

Return:
none

Arguments
bShow, true when spy box have to be displayed near the selection, false to hide spy box

4.18.19 SETRULES (CTR0016)

This function sets the rules for the control (positions of rules).

C/C++ definition:

```
void SetRules(LPCTSTR  szRules);
```

C# definition:

```
void SetRules(string szRules);
```

Return:
none

Arguments
szRules, the list of rules to display in control. These are a list of x grid coordinates separated by comas.

4.18.20 GETRULES (CTR0017)

This function gets the rule list used in the control.

C/C++ definition:

```
LPCTSTR  GetRules();
```

C# definition:

```
string GetRules();
```

Return:

the list of rules in control. These are a list of x grid coordinates separated by comas.

Arguments

none

4.18.21 SETFBDORDER (CTR0600)

This function sets the manual FBD order coming from FBD Order tree in straton framework. straton framework use only! Do not use it!

C/C++ definition:

```
bool SetFBDOrder(LPCTSTR  szOrder);
```

C# definition:

```
bool SetFBDOrder(string szOrder);
```

Return:

bool: true if the order has been modified

Arguments

szOrder, the list of items to set in control. This is a private format and support won't be assumed if OEM use it.

4.18.22 GETFBDORDER (CTR0601)

This function gets the manual FBD order. straton framework use only! Do not use it!

C/C++ definition:

```
LPCTSTR  GetFBDOrder();
```

C# definition:

```
string GetFBDOrder();
```

Return:

Returns the FBD manual order. This is a string representing list of ordered items. This is a private format and support won't be assumed if OEM use it.

Arguments

none

4.18.23 ACTIVATEFBDORDER (CTR0602)

This function activate/deactivate the manual FBD ordering. straton framework use only! Do not use it!

C/C++ definition:

```
bool ActivateFBDOrder(bool bActivate);
```

C# definition:

```
bool GetFBDOrder(bool bActivate);
```

Return:

Returns true if the manual ordering status has been changed.

Arguments

Bool: true to activate manual ordering

4.19 LD control

4.19.1 CANALIGNCOILS (CTR0221)

This function test if coils can be aligned on the same column.

C/C++/C# definition:

```
bool CanAlignCoils();
```

Return:
true if coil can be aligned

Arguments
none

4.19.2 ALIGNCOILS (CTR0222)

This function aligns coil on the same column.

C/C++/C# definition:

```
void AlignCoils();
```

Return:
none

Arguments
none

4.19.3 CANINSERTCONTACTBEFORE (CTR0223)

This function tests if contact can be inserted before the current selection.

C/C++/C# definition:

```
bool CanInsertContactBefore();
```

Return:
true if contact can be inserted

Arguments
none

4.19.4 INSERTCONTACTBEFORE (CTR0224)

This function inserts a contact before the current selection.

C/C++/C# definition:

```
void InsertContactBefore();
```

Return:
none

Arguments

none

4.19.5 CANINSERTCONTACTAFTER (CTR0225)

This function tests if contact can be inserted after the current selection.

C/C++ definition:

```
bool CanInsertContactAfter();
```

Return:
true if contact can be inserted

Arguments
none

4.19.6 INSERTCONTACTAFTER (CTR0226)

This function inserts a contact after the current selection..

C/C++/C# definition:

```
void InsertContactAfter();
```

Return:
none

Arguments
none

4.19.7 CANINSERTCONTACTPARALLEL (CTR0227)

This function tests if contact can be inserted parallel to the current selection.

C/C++/C# definition:

```
bool CanInsertContactParallel();
```

Return:
true if contact can be inserted

Arguments
none

4.19.8 INSERTCONTACTPARALLEL (CTR0228)

This function inserts a contact parallel to the current selection.

C/C++/C# definition:

```
void InsertContactParallel();
```

Return:
none

Arguments

none

4.19.9 CANINSERTCOIL (CTR0229)

This function tests if coil can be inserted at the current selection.

C/C++/C# definition:

```
bool CanInsertCoil();
```

Return:
true if coil can be inserted

Arguments
none

4.19.10 INSERTCOIL (CTR0230)

This function inserts a coil at current selection.

C/C++/C# definition:

```
void InsertCoil();
```

Return:
none

Arguments
None

4.19.11 CANINSERTCOILPARALLEL (CTR0603)

This function tests if coil can be inserted parallel to the current selection.

C/C++/C# definition:

```
bool CanInsertCoilParallel();
```

Return:
true if coil can be inserted

Arguments
none

4.19.12 INSERTCOILPARALLEL (CTR0604)

This function inserts a coil parallel to current selection.

C/C++/C# definition:

```
void InsertCoilParallel();
```

Return:
none

Arguments

None

4.19.13 CANINSERTCOILBEFORE (CTR0606)

This function tests if coil can be inserted before the current selection.

C/C++/C# definition:

```
bool CanInsertCoilBefore();
```

Return:
true if coil can be inserted

Arguments
none

4.19.14 INSERTCOILBEFORE (CTR0607)

This function inserts a coil before the current selection.

C/C++/C# definition:

```
void InsertCoilBefore();
```

Return:
none

Arguments
None

4.19.15 CANINSERTCOILAFTER (CTR0608)

This function tests if coil can be inserted after the current selection.

C/C++/C# definition:

```
bool CanInsertCoilAfter();
```

Return:
true if coil can be inserted

Arguments
none

4.19.16 INSERTCOILAFTER (CTR0609)

This function inserts a coil after the current selection.

C/C++/C# definition:

```
void InsertCoilBefore();
```

Return:
none

Arguments

None

4.19.17 CANINSERTFBBEFORE (CTR0231)

This function tests if Function Block can be inserted before the current selection.

C/C++/C# definition:

```
bool CanInsertFBBefore();
```

Return:
true if Function Block can be inserted

Arguments
none

4.19.18 INSERTFBBEFORE (CTR0232)

This function inserts a Function Block before the current selection.

C/C++ definition:

```
void InsertFBBefore();
```

Return:
none

Arguments
none

4.19.19 CANINSERTFBAFTER (CTR0233)

This function tests if Function Block can be inserted after the current selection.

C/C++/C# definition:

```
bool CanInsertFBAfter();
```

Return:
true if Function Block can be inserted

Arguments
none

4.19.20 INSERTFBAFTER (CTR0234)

This function inserts a Function Block after the current selection.

C/C++/C# definition:

```
void InsertFBAfter();
```

Return:
none

Arguments

none

4.19.21 CANINSERTFBPARALLEL (CTR0235)

This function tests if Function Block can be inserted parallel to the current selection.

C/C++/C# definition:

```
bool CanInsertFBParallel();
```

Return:
true if Function Block can be inserted

Arguments
none

4.19.22 INSERTFBPARALLEL (CTR0236)

This function inserts a Function Block parallel to the current selection.

C/C++/C# definition:

```
void InsertFBParallel();
```

Return:
none

Arguments
none

4.19.23 CANINSERTJUMP (CTR0237)

This function tests if jump can be inserted at current selection.

C/C++/C# definition:

```
bool CanInsertJump();
```

Return:
true if jump can be inserted

Arguments
none

4.19.24 INSERTJUMP (CTR0238)

This function inserts a jump at the current selection.

C/C++/C# definition:

```
void InsertJump();
```

Return:
none

Arguments

none

4.19.25 CANINSERTRUNG (CTR0239)

This function tests if a new rung can be inserted at current selection.

C/C++ definition:

```
bool CanInsertRung();
```

Return:
true if rung can be inserted

Arguments
none

4.19.26 INSERTRUNG (CTR0240)

This function inserts a rung at the current selection.

C/C++/C# definition:

```
void InsertRung();
```

Return:
none

Arguments
None

4.19.27 INSERTRUNGAFTER (CTR0591)

This function inserts a rung after the current selection.

C/C++/C# definition:

```
bool InsertRungAfter();
```

Return:
BOOL: true if a new rung has been inserted

Arguments
None

4.19.28 CANINSERTCOMMENT (CTR0241)

This function tests if a new comment can be inserted at current selection.

C/C++/C# definition:

```
bool CanInsertComment();
```

Return:
true if comment can be inserted

Arguments
none

4.19.29 INSERTCOMMENT (CTR0242)

This function inserts a comment at the current selection.

C/C++/C# definition:

```
void InsertComment();
```

Return:
none

Arguments
none

4.19.30 CANINSERTHORZ (CTR0243)

This function tests if a new horizontal line can be inserted at current selection.

C/C++/C# definition:

```
bool CanInsertHorz();
```

Return:
true if line can be inserted

Arguments
none

4.19.31 INSERTHORZ (CTR0244)

This function inserts a horizontal line at the current selection.

C/C++/C# definition:

```
void InsertHorz();
```

Return:
none

Arguments
none

4.19.32 WRAPRUNGS (CTR0245)

This function make all rung using the same width.

C/C++/C# definition:

```
void WrapRungs(bool bWrap);
```

Return:
none

Arguments
bWrap, should be true to wrap all rungs

4.20 FreeForm LD control

4.20.1 CANINSERTFFLDITEM (CTR0246)

This function tests if the freeform LD item can be inserted at current position.

C/C++/C# definition:

```
bool CanInsertFFLDItem(int iType);
```

Return:

returns true if an item can be inserted

Arguments

iType, the type of item to insert. Can be one of these values (see W5EditFFLDApi.h file):

FFLD_CONTACT	1	When active, if user drag n drop, it will be added a contact at mouse position
FFLD_CONTACT_INVERT	2	When active, if user drag n drop, it will be added a invert contact at mouse position
FFLD_CONTACT_P	3	When active, if user drag n drop, it will be added a pulse contact at mouse position
FFLD_CONTACT_PINVERT	4	When active, if user drag n drop, it will be added a pulse invert contact at mouse position
FFLD_CONTACT_N	5	When active, if user drag n drop, it will be added a N contact at mouse position
FFLD_CONTACT_NINVERT	6	When active, if user drag n drop, it will be added a invert N contact at mouse position
FFLD_COIL	7	When active, if user drag n drop, it will be added a coil at mouse position
FFLD_COIL_INVERT	8	When active, if user drag n drop, it will be added a invert coil at mouse position
FFLD_COIL_S	9	When active, if user drag n drop, it will be added a S coil at mouse position
FFLD_COIL_R	10	When active, if user drag n drop, it will be added a R coil at mouse position
FFLD_LINE_HORZ	11	When active, if user drag n drop, it will be added a horizontal line at mouse position
FFLD_LINE_VERT	12	When active, if user drag n drop, it will be added a vertical line at mouse position
FFLD_FB	14	When active, if user drag n drop, it will be added a block at mouse position
FFLD_DATAIN	15	When active, if user drag n drop, it will be added a DataIn item at mouse position
FFLD_DATAIN_INVERT	16	When active, if user drag n drop, it will be added a invert DataIn item at mouse position
FFLD_DATAOUT	17	When active, if user drag n drop, it will be added a DataOut item at mouse position
FFLD_JUMP	18	When active, if user drag n drop, it will be added a jump at mouse position
FFLD_RETURN	19	When active, if user drag n drop, it will be added a return at mouse position
FFLD_NETWORK	20	When active, if user drag n drop, it will be added a jump at mouse position
FFLD_ROW	21	When active, if user drag n drop, it will be added an empty row at mouse position
FFLD_COMMENT	22	When active, if user drag n drop, it will be added a comment at mouse position
FFLD_LABEL	23	When active, if user drag n drop, it will be added a label at mouse position
FFLD_PRAGMA	24	When active, if user drag n drop, it will be added a pragma at mouse position
FFLD_COIL_P	26	When active, if user drag n drop, it will be added a P coil at mouse position
FFLD_COIL_N	27	When active, if user drag n drop, it will be added a N coil at mouse position

4.20.2 INSERTFFLDITEM (CTR0247)

This function inserts a freeform LD item at current position.

C/C++/C# definition:

```
bool InsertFFLDItem(int iType);
```

Return:
true if the item has been inserted

Arguments
iType, the type of item to insert. For possible values, see chapter “CanInsertFFLDItem”.

4.20.3 SETFIRSTNETWORKTAG (CTR0248)

This function sets the first tag of FFLD control.

C/C++/C# definition:

```
void SetFirstNetworkTag(int iFirstTag);
```

Return:
none

Arguments
iFirstTag, the tag of the first network in control

4.20.4 GETNETSEL (CTR0249)

This function returns the current network tag.

C/C++/C# definition:

```
int GetNetSel();
```

Return:
returns the tag of the selected network

Arguments
none

4.20.5 GETNETITEMSEL (CTR0250)

This function returns the coordinates of caret relative to current network.

C/C++ definition:

```
long GetNetItemSel();
```

C# definition:

```
Int32 GetNetItemSel();
```

Return:
returns the current coordinates in a int32.
x coordinates can be retrieved in low word
y coordinate can be retrieved in high word

Arguments

none

4.20.6 UPDATEUDFB (CTR0251)

This function updates size of the selected UDFB if selected UDFB size does not correspond to the data base definition.

C/C++/C# definition:

```
void UpdateUDFB();
```

Return:
none

Arguments
none

4.20.7 NEEDUPDATEUDFB (CTR0252)

This function checks if selected UDFB and database definition have same height and return true if the current selection should be updated.

C/C++/C# definition:

```
bool NeedUpdateUDFB();
```

Return:
true if the UDFB size needs to be updated

Arguments
none

4.20.8 LISTUPDATEUDFB (CTR0253)

This function returns the list of UDFB that have to be updated (compiler format).

C/C++ definition:

```
LPCTSTR  ListUpdateUDFB();
```

C# definition:

```
string ListUpdateUDFB();
```

Return:
returns a list of compiler location strings (same format as used in function “LocateError”) separate by “\n”

Arguments
none

4.20.9 SETLINEPERCOMMENT (CTR0254)

This function sets the number of visible lines in a comment (multiple of 2 or -1 to display all lines).

C/C++/C# definition:

```
void SetLinePerComment(int nbLine);
```

Return:
none

Arguments
nbLine, the number of lines to display

4.20.10 IMPORTCOMMENT (CTR0255)

This function displays dialog box for importing comments.

C/C++/C# definition:

```
void ImportComment();
```

Return:
none

Arguments
none

4.20.11 EXPORTCOMMENT (CTR0256)

This function displays dialog box for exporting comments.

C/C++/C# definition:

```
void ExportComment();
```

Return:
none

Arguments
none

4.20.12 DISPLAYCONFIRMATION (CTR0257)

This function is used to display or not confirmation boxes in control.

C/C++/C# definition:

```
void DisplayConfirmation(bool bDisplay);
```

Return:
none

Arguments
bDisplay, true to display confirmations boxes

4.20.13 CANCLEARROW (CTR0258)

This function test if entire row can be deleted.

C/C++/C# definition:

```
bool CanClearRow();
```

Return:
true if row can be deleted

Arguments
none

4.20.14 CLEARROW (CTR0259)

This function clears the focused row.

C/C++/C# definition:

```
bool ClearRow();
```

Return:
true if row has been deleted

Arguments
none

4.20.15 CANCLEARNETWORK (CTR0260)

This function tests if entire network can be deleted.

C/C++/C# definition:

```
bool CanClearNetwork();
```

Return:
true if network can be deleted

Arguments
none

4.20.16 CLEARNETWORK (CTR0261)

This function clears the entire network (true if success).

C/C++ definition:

```
bool ClearNetwork();
```

Return:
true if network has been deleted

Arguments
none

4.20.17 CANEXPAND (CTR0262)

This function tests if selected networks can be expanded.

C/C++/C# definition:

```
bool CanExpand();
```

Return:
true if current networks can be expanded

Arguments
none

4.20.18 EXPAND (CTR0263)

This function expands selected networks.

C/C++/C# definition:

```
void Expand();
```

Return:
none

Arguments
none

4.20.19 CANEXPANDALL (CTR0264)

This function tests if all networks can be expanded.

C/C++/C# definition:

```
bool CanExpandAll();
```

Return:
true if all networks can be expanded

Arguments
none

4.20.20 EXPANDALL (CTR0265)

This function expands all networks.

C/C++/C# definition:

```
void ExpandAll();
```

Return:
none

Arguments
none

4.20.21 CANCOLLAPSE (CTR0266)

This function tests if selected networks can be collapsed.

C/C++/C# definition:

```
bool CanCollapse();
```

Return:
true if selected networks can be collapsed

Arguments
none

4.20.22 COLLAPSE (CTR0267)

This function collapses selected networks.

C/C++/C# definition:

```
void Collapse();
```

Return:
none

Arguments
none

4.20.23 CANCOLLAPSEALL (CTR0268)

This function tests if all networks can be collapsed.

C/C++/C# definition:

```
bool CanCollapseAll();
```

Return:
true if all networks can be collapsed

Arguments
none

4.20.24 COLLAPSEALL (CTR0269)

This function collapses all networks.

C/C++/C# definition:

```
void CollapseAll();
```

Return:
none

Arguments
none

4.20.25 SWAPCOLLAPSE (CTR0270)

This function toggles expanded/collapsed selected networks.

C/C++/C# definition:

```
void SwapCollapse();
```

Return:
none

Arguments
none

4.21 SFC control

4.21.1 LOCK (CTR0271)

This function is obsolet.

4.21.2 CANINSERT (CTR0272)

This function tests if a new item can be inserted at current position.

C/C++/C# definition:

```
bool CanInsert();
```

Return:

true if item can be inserted

Arguments

none

4.21.3 INSERTSTEP (CTR0273)

This function inserts STEP at current position.

C/C++/C# definition:

```
void InsertStep();
```

Return:

none

Arguments

none

4.21.4 INSERTTRANS (CTR0274)

This function inserts TRANSITION at current position.

C/C++/C# definition:

```
void InsertTrans();
```

Return:

none

Arguments

none

4.21.5 INSERTJUMP (CTR0275)

This function inserts JUMP at current position.

C/C++/C# definition:

```
void InsertJump();
```

Return:
none

Arguments
none

4.21.6 INSERTMAINDIV (CTR0276)

This function inserts a new divergence at current position.

C/C++ definition:

```
void InsertMainDiv();
```

Return:
none

Arguments
none

4.21.7 INSERTDIV (CTR0277)

This function inserts a divergence branch in the current divergence.

C/C++/C# definition:

```
void InsertDiv();
```

Return:
none

Arguments
none

4.21.8 INSERTCNV (CTR0278)

This function inserts a convergence at current position.

C/C++/C# definition:

```
void InsertCnv();
```

Return:
none

Arguments
none

4.21.9 INSERTMACRO (CTR0279)

DEPRECATED!

This function inserts a new MACRO at current position.

C/C++/C# definition:

```
void InsertMacro();
```

Return:
none

Arguments
none

4.21.10 INSERTMACROBODY (CTR0280)

DEPRECATED!
This function inserts a MACRO body at current position.

C/C++/C# definition:

```
void InsertMacroBody();
```

Return:
none

Arguments
none

4.21.11 INSERTINITSTEP (CTR0281)

This function inserts an initial step at current position.

C/C++/C# definition:

```
void InsertInitStep();
```

Return:
none

Arguments
none

4.21.12 CANSWAPSTYLE (CTR0282)

This function test if item style can be swapped.

C/C++/C# definition:

```
bool CanSwapStyle();
```

Return:

Arguments

4.21.13 SWAPSTYLE (CTR0283)

This function swaps the current item style.

C/C++/C# definition:

```
void SwapStyle();
```

Return:
none

Arguments
none

4.21.14 GETTRANSCODE (CTR0284)

This function gets the transition source code (iRef is the transition reference).

C/C++ definition:

```
LPCTSTR GetTransCode(int iRef);
```

C# definition:

```
string GetTransCode(int iRef);
```

Return:
returns the source code of the specified transition, null is no transition found

Arguments
iRef, a transition reference

4.21.15 GETTRANSNOTE (CTR0285)

This function retrieves the transition note text.

C/C++ definition:

```
LPCTSTR GetTransNote(int iRef);
```

C# definition:

```
string GetTransNote(int iRef);
```

Return:
returns the note/comment of the specified transition, null is no transition found

Arguments
iRef, a transition reference

4.21.16 SETTRANSNOTE (CTR0286)

This function sets the transition source code.

C/C++ definition:

```
void SetTransNote(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetTransNote(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a transition reference
szCode, the note text to set

4.21.17 SETTRANSCODE (CTR0287)

This function sets the transition code.

C/C++ definition:

```
void SetTransCode(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetTransCode(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a transition reference
szCode, the transition code to set

4.21.18 GETSTEPCODE_DEF (CTR0288)

This function gets the step qualifier default source code.

C/C++ definition:

```
LPCTSTR GetStepCode_Def(int iRef);
```

C# definition:

```
string GetStepCode_Def(int iRef);
```

Return:
returns the code of the specified step, null is no step found

Arguments
iRef, a step reference

4.21.19 SETSTEPCODE_DEF (CTR0289)

This function sets the step qualifier default source code.

C/C++ definition:

```
void SetStepCode_Def(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetStepCode_Def(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a step reference
szCode, the step code to set

4.21.20 GETSTEPCODE_P1 (CTR0290)

This function gets the step qualifier P1 source code.

C/C++ definition:

```
LPCTSTR GetStepCode_P1(int iRef);
```

C# definition:

```
string GetStepCode_P1(int iRef);
```

Return:
returns the code of the specified P1 qualifier step, null is no step found

Arguments
iRef, a step reference

4.21.21 SETSTEPCODE_P1 (CTR0291)

This function sets the step qualifier P1 source code.

C/C++ definition:

```
void SetStepCode_P1(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetStepCode_P1(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a step reference
szCode, the step P1 code to set

4.21.22 GETSTEPCODE_N (CTR0292)

This function gets the step qualifier N source code.

C/C++ definition:

```
LPCTSTR GetStepCode_N(int iRef);
```

C# definition:

```
string GetStepCode_N(int iRef);
```

Return:
returns the code of the specified N qualifier step, null is no step found

Arguments

iRef, a step reference

4.21.23 SETSTEPCODE_N (CTR0293)

This function sets the step qualifier N source code.

C/C++ definition:

```
void SetStepCode_N(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetStepCode_N(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a step reference
szCode, the step N code to set

4.21.24 GETSTEPCODE_P0 (CTR0294)

This function gets the step qualifier P0 source code.

C/C++ definition:

```
LPCTSTR GetStepCode_P0(int iRef);
```

C# definition:

```
string GetStepCode_P0(int iRef);
```

Return:
returns the code of the specified P0 qualifier step, null is no step found

Arguments
iRef, a step reference

4.21.25 SETSTEPCODE_P0 (CTR0295)

This function sets the step qualifier P0 source code.

C/C++ definition:

```
void SetStepCode_P0(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetStepCode_P0(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a step reference
szCode, the step P0 code to set

4.21.26 GETSTEPNOTE (CTR0296)

This function gets the step note text.

C/C++ definition:

```
LPCTSTR GetStepNote(int iRef);
```

C# definition:

```
string GetStepNote(int iRef);
```

Return:

returns the note/comment of the specified step, null is no step found

Arguments

iRef, a step reference

4.21.27 SETSTEPNOTE (CTR0297)

This function sets the step note text.

C/C++ definition:

```
void SetStepNote(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetStepNote(int iRef, string szCode);
```

Return:

none

Arguments

iRef, a step reference

szCode, the step note/comment to set

4.21.28 GETMACRONOTE (CTR0298)

This function gets the macro note text.

C/C++ definition:

```
LPCTSTR GetMacroNote(int iRef);
```

C# definition:

```
string GetMacroNote(int iRef);
```

Return:

returns the note/comment of the specified macro, null is no macro found

Arguments

iRef, a macro reference

4.21.29 SETMACRONOTE (CTR0299)

This function sets the macro note text.

C/C++ definition:

```
void SetMacroNote(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetMacroNote(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a macro reference
szCode, the macro note/comment to set

4.21.30 ISSELSTEP (CTR0300)

This function gets the selected step reference.

C/C++ definition:

```
long IsSelStep();
```

C# definition:

```
Int32 IsSelStep();
```

Return:
the step reference found, 0 if not found

Arguments
none

4.21.31 ISSELTRANS (CTR0301)

This function gets the selected transition reference.

C/C++ definition:

```
long IsSelTrans();
```

C# definition:

```
Int32 IsSelTrans();
```

Return:
the transition reference found, 0 if not found

Arguments
none

4.21.32 ISSELMACRO (CTR0302)

This function gets the selected macro reference.

C/C++ definition:

```
long IsSelMacro();
```

C# definition:

```
Int32 IsSelMacro();
```

Return:
the macro reference found, 0 if not found

Arguments
none

4.21.33 GETSELREFNUM (CTR0303)

This function gets the selected item reference.

C/C++ definition:

```
long GetSelRefNum();
```

C# definition:

```
Int32 GetSelRefNum();
```

Return:
the item reference found, 0 if not found

Arguments
none

4.21.34 LOCKSTEP (CTR0304)

This function locks/unlocks the selected step (if any).

C/C++/C# definition:

```
void LockStep(bool bLock, int iRef);
```

Return:
none

Arguments
bLock, specifies if step should be locked or unlocked
iRef, a step reference

4.21.35 LOCKTRANS (CTR0305)

This function locks/unlocks the selected transition (if any).

C/C++/C# definition:

```
void LockTrans(bool bLock, int iRef);
```

Return:
none

Arguments
bLock, specifies if transition should be locked or unlocked
iRef, a transition reference

4.21.36 LOCKMACRO (CTR0306)

This function locks/unlocks the macro.

C/C++/C# definition:

```
void LockMacro(bool bLock, int iRef);
```

Return:
none

Arguments
bLock, specifies is macro should be locked or unlocked
iRef, a macro reference

4.21.37 ISSTEPLOCK (CTR0307)

This function tests if selected step is locked.

C/C++/C# definition:

```
bool IsStepLock(int iRef);
```

Return:
true is specified step is lock

Arguments
iRef, a step reference

4.21.38 ISTRANSLOCK (CTR0308)

This function tests if selected transition is locked.

C/C++/C# definition:

```
bool IsTransLock(int iRef);
```

Return:
true is specified transition is lock

Arguments
iRef, a transition reference

4.21.39 ENTERSELREF (CTR0309)

This function displays box to change the reference of the current item.

C/C++/C# definition:

```
void EnterSelRef();
```

Return:

none

Arguments
none

4.21.40 CANENTERSELREF (CTR0310)

This function tests if reference of the selected item can be changed.

C/C++/C# definition:

```
bool CanEnterSelRef();
```

Return:
true if reference of the selection can be edited

Arguments
none

4.21.41 CANEDITSELCODE (CTR0311)

This function tests if level 2 can be edited.

C/C++/C# definition:

```
bool CanEditSelCode();
```

Return:
true if level 2 of the selection can be edited

Arguments
none

4.21.42 GETSTEPLANGUAGEP1 (CTR0312)

This function gets the language of P1 qualifier.

C/C++/C# definition:

```
int GetStepLanguageP1(int iRef);
```

Return:
returns the language value of the specified P1 step reference
Can be one of these values:

W5EDITSFCLV2_NONE	0
W5EDITSFCLV2_STIL	1
W5EDITSFCLV2_LD	2
W5EDITSFCLV2_FBD	3
W5EDITSFCLV2_FFLD	4

This function returns W5EDITSFCLV2_NONE if specified reference is not found or not a step

Arguments

iRef, a step reference

4.21.43 GETSTEPLANGUAGEN (CTR0313)

This function gets the language of N qualifier.

C/C++/C# definition:

```
int GetStepLanguageN(int iRef);
```

Return:

returns the language value of the specified N step reference
Can be one of these values:

W5EDITSFCLV2_NONE	0
W5EDITSFCLV2_STIL	1
W5EDITSFCLV2_LD	2
W5EDITSFCLV2_FBD	3
W5EDITSFCLV2_FFLD	4

This function returns W5EDITSFCLV2_NONE if specified reference is not found or not a step

Arguments

iRef, a step reference

4.21.44 GETSTEPLANGUAGEP0 (CTR0314)

This function gets the language of P0 qualifier.

C/C++/C# definition:

```
int GetStepLanguageP0(int iRef);
```

Return:

returns the language value of the specified P0 step reference
Can be one of these values:

W5EDITSFCLV2_NONE	0
W5EDITSFCLV2_STIL	1
W5EDITSFCLV2_LD	2
W5EDITSFCLV2_FBD	3
W5EDITSFCLV2_FFLD	4

This function returns W5EDITSFCLV2_NONE if specified reference is not found or not a step

Arguments

iRef, a step reference

4.21.45 GETTRANSLANGUAGE (CTR0315)

This function gets the language of transition.

C/C++/C# definition:

```
int GetTransLanguage(int iRef);
```

Return:
returns the language value of the specified transition reference
Can be one of these values:

W5EDITSFCLV2_NONE	0
W5EDITSFCLV2_STIL	1
W5EDITSFCLV2_LD	2
W5EDITSFCLV2_FBD	3
W5EDITSFCLV2_FFLD	4

This function returns W5EDITSFCLV2_NONE if specified reference is not found or not a transition

Arguments
iRef, a transition reference

4.21.46 SETNOTEDISPLAY (CTR0316)

This function sets/unsets the note displaying.

C/C++/C# definition:

```
void SetNoteDisplay(bool bSet);
```

Return:
none

Arguments
bSet, if true the only notes are displayed

4.21.47 GETNOTEDISPLAY (CTR0317)

This function gets displaying note mode.

C/C++/C# definition:

```
bool GetNoteDisplay();
```

Return:
true if only notes are displayed

Arguments
none

4.21.48 GETDISPLAY (CTR0568)

This function returns the displaying mode of the steps and transitions in level1

C/C++ definition:

```
WORD GetDisplay();
```

C# definition:

```
UInt16 GetDisplay();
```

Return:
returns value can take one of these values (from W5EditSFCApi.h):

DISPLAY_ALL	All qualifiers are displayed in the SFC item
DISPLAY_NOTE	Only notes are displayed in the SFC item
DISPLAY_ACTION	Only action / transition are displayed in SFC item

Arguments
none

4.21.49 SETDISPLAY (CTR0569)

This function changes the displaying mode of the steps and transitions in level1

C/C++ definition:

```
void SetDisplay(WORD wDisplay);
```

C# definition:

```
void SetDisplay(UInt16 wDisplay);
```

Return:
none

Arguments
wDisplay is the mode to set. It cantake one of these values:

DISPLAY_ALL	All qualifiers are displayed in the SFC item
DISPLAY_NOTE	Only notes are displayed in the SFC item
DISPLAY_ACTION	Only action / transition are displayed in SFC item

4.21.50 CANRENUMBER (CTR0318)

This function tests if references of items can be renumbered.

C/C++/C# definition:

```
bool CanRenumber();
```

Return:
true is all items can be renumbered (if at least one item is lock, this returns false)

Arguments
none

4.21.51 RENUMBER (CTR0319)

This function renumbers all references of items.

C/C++/C# definition:

```
void Renumber();
```

Return:
none

Arguments
none

4.21.52 NEXTPOSITEM (CTR0320)

This function selects the next item (bLoop to loop after passed the end of file).

C/C++ definition:

```
long NextPosItem(BOOL bLoop);
```

C# definition:

```
Int32 NextPosItem(bool bLoop);
```

Return:
the reference of the selected item in document

Arguments
bLoop, true if search should pass the end of file

4.21.53 ACTIVESTEP (CTR0321)

This function has been replace by the “FindReplace” function used with command “W5FIND_ACTIVESTEP”.

4.21.54 ISSTEP (CTR0322)

This function tests if item at position is step.

C/C++ definition:

```
long IsStep(int iX, int iY);
```

C# definition:

```
Int32 IsStep(int iX, int iY);
```

Return:
the reference of the found step (0 if not found)

Arguments
iX, the x grid coordinate
iY, the y grid coordinate

4.21.55 ISTRANS (CTR0323)

This function tests if item at position is transition.

C/C++ definition:

```
long IsTrans(int iX, int iY);
```

C# definition:

```
Int32 IsTrans(int iX, int iY);
```

Return:
the reference of the found transition (0 if not found)

Arguments
iX, the x grid coordinate
iY, the y grid coordinate

4.21.56 ISCOMMENT (CTR0564)

This function tests if item at position is comment.

C/C++ definition:

```
long IsComment(int iX, int iY);
```

C# definition:

```
Int32 IsComment(int iX, int iY);
```

Return:
the reference of the found comment (0 if not found)

Arguments
iX, the x grid coordinate
iY, the y grid coordinate

4.21.57 ISMACRO (CTR0565)

This function tests if item at position is macro.

C/C++ definition:

```
long IsMacro(int iX, int iY);
```

C# definition:

```
Int32 IsMacro(int iX, int iY);
```

Return:
the reference of the found macro (0 if not found)

Arguments
iX, the x grid coordinate
iY, the y grid coordinate

4.21.58 ISSELCOMMENT (CTR0324)

This function gets the selected comment reference.

C/C++ definition:

```
DWORD IsSelComment();
```

C# definition:

```
UInt32 IsSelComment();
```

Return:
the selected comment reference (0 if no comment selected)

Arguments
none

4.21.59 LOCKCOMMENT (CTR0325)

This function locks/unlocks the referenced comment.

C/C++/C# definition:

```
void LockComment(bool bLock, int iRef);
```

Return:
none

Arguments
bLock, specifies is macro should be locked or unlocked
iRef, a comment reference

4.21.60 GETCOMMENTNOTE (CTR0326)

This function gets the comment note text.

C/C++ definition:

```
LPCTSTR GetCommentNote(int iRef);
```

C# definition:

```
string GetCommentNote(int iRef);
```

Return:
returns the text of the specified comment, null is no comment found

Arguments
iRef, a comment reference

4.21.61 SETCOMMENTNOTE (CTR0327)

This function sets the comment note text.

C/C++ definition:

```
void SetCommentNote(int iRef, LPCTSTR szCode);
```

C# definition:

```
void SetCommentNote(int iRef, string szCode);
```

Return:
none

Arguments
iRef, a comment reference
szCode, the text to set

4.21.62 GETITEMGUID (CTR0328)

This function retrieves the GUID of an item at given position.

C/C++ definition:

```
DWORD GetItemGUID(int iX, int iY);
```

C# definition:

```
UInt32 GetItemGUID(int iX, int iY);
```

Return:
the GUID (unique ID) of the item at position iX,iY

Arguments
iX, the x grid coordinate
iY, the y grid coordinate

4.21.63 SETTIMER (CTR0329)

This function inserts/sets a timer transition at current position.

C/C++/C# definition:

```
void SetTimer();
```

Return:
none

Arguments
none

4.21.64 SETSFCSETTINGS2 (CTR0330)

This function sets the SFC settings (same as SETSFCSETTINGS, but with flags instead of structure).

C/C++ definition:

```
bool SetSFCSettings2(DWORD dwFlags);
```

C# definition:

```
bool SetSFCSettings2(UInt32 dwFlags);
```

Return:
returns true if SFC settings have been set

Arguments
dwFlags, the setting flags to set. Can be a combination of these values (see W5EditApi.h):

SFC_NOFIFLVL2	0x00000001

4.21.65 SETSFCSETTINGS (CTR0331)

This function sets the SFC settings (as SetSFCSettings2 but using a structure).

C/C++ definition:

```
bool SetSFCSettings(str_W5SFCSettings* pStrSettings);
```

C# definition:

```
bool SetSFCSettings(IntPtr pStrSettings) ;
```

To use this function in C# please refer to Annexe “Using structures in C# wrappers”.
By the way, the best way of working is using “SetSFCSettings2” instead of “SetSFCSettings”.

Return:
returns true if SFC settings have been set

Arguments
pStrSettings, the structure (defined in W5EditApi.h) used to change SFC settings

4.22 ActiveX control

4.22.1 CANALIGN (CTR0332)

This function tests if selected items can be aligned.

C/C++/C# definition:

```
bool CanAlign() ;
```

Return:
true if selected items can be aligned

Arguments
none

4.22.2 ALIGN (CTR0333)

This function aligns item with the main selected item.

C/C++/C# definition:

```
void Align(int iHorz, int iVert);
```

Return:
none

Arguments
iHorz: ACT_ALIGNLEFT to left align, ACT_ALIGNRIGHT to right align and ACT_ALIGNHORZCENTER for an horizontal centered alignment
iVert: ACT_ALIGNTOP to top align, ACT_ALIGNBOTTOM to bottom align and ACT_ALIGNVERTCENTER for a vertical centered alignment

4.22.3 GETNBITEM (CTR0334)

This function gets number of items in control.

C/C++/C# definition:

```
int GetNbItem();
```

Return:
the number of items

Arguments
none

4.22.4 GETZORDERSTRUCT (CTR0335)

This function is obsolete.

4.22.5 CANMOVETOP (CTR0336)

This function tests if item can be moved top in z order.

C/C++/C# definition:

```
bool CanMoveTop();
```

Return:
true if item can move

Arguments
none

4.22.6 MOVETOP (CTR0337)

This function moves item to top of z order.

C/C++/C# definition:

```
void MoveTop();
```

Return:
none

Arguments
none

4.22.7 CANMOVEBOTTOM (CTR0338)

This function tests if item can be moved bottom in z order.

C/C++/C# definition:

```
bool CanMoveBottom();
```

Return:
true if item can move

Arguments
none

4.22.8 MOVEBOTTOM (CTR0339)

This function moves item to bottom of z order.

C/C++ definition:

```
void MoveBottom();
```

Return:
none

Arguments
none

4.22.9 CANRESIZE (CTR0340)

This function tests if selection can be resized.

C/C++/C# definition:

```
bool CanResize();
```

Return:
true if selection can be resized

Arguments
none

4.22.10 RESIZE (CTR0341)

This function resizes the current selection.

C/C++/C# definition:

```
void Resize(bool bX, bool bY);
```

Return:
none

Arguments
bX, if true, all selected items will have same width as main selected item
bY, if true, all selected items will have same height as main selected item

4.22.11 CANCHANGEBKCOLOR (CTR0342)

This function tests if background color can be changed.

C/C++/C# definition:

```
bool CanChangeBkColor();
```

Return:
true is background color can be changed

Arguments
none

4.22.12 CHANGEBKCOLOR (CTR0343)

This function changes the current background color.

C/C++/C# definition:

```
void ChangeBkColor();
```

Return:
none

Arguments
none

4.22.13 NEWFILE (CTR0344)

This function creates a new empty control.

C/C++/C# definition:

```
void NewFile();
```

Return:
none

Arguments
none

4.22.14 SELECTITEM (CTR0345)

This function selects the item (eventually, deselect all others).

C/C++ definition:

```
void SelectItem(DWORD_PTR dwData, BOOL bDeselectAll)
```

C# definition:

```
void SelectItem(IntPtr dwData, bool bDeselectAll);
```

Return:
none

Arguments
dwData, the graphic object ID
bDeselectAll, if true deselect all items before select this one

4.22.15 GETITEMID (CTR0346)

This function gets the ID of the selected item (file ID).

C/C++ definition:

```
DWORD GetItemID();
```

C# definition:

```
UInt32 GetItemID();
```

Return:
the selected item graphic ID

Arguments
none

4.22.16 MOVEBEFORE (CTR0347)

This function moves item top of another item in z order.

C/C++ definition:

```
void MoveBefore(DWORD dwMove, DWORD dwBefore);
```

C# definition:

```
void MoveBefore(UInt32 dwMove, UInt32 dwBefore);
```

Return:
none

Arguments
dwMove, the graphic item ID to move before other one
dwBefore, the reference graphic item ID

4.22.17 MOVEAFTER (CTR0348)

This function moves item bottom to another in z order.

C/C++ definition:

```
void MoveAfter(DWORD dwMove, DWORD dwAfter);
```

C# definition:

```
void MoveAfter(UInt32 dwMove, UInt32 dwAfter);
```

Return:
none

Arguments
dwMove, the graphic item ID to move after other one
dwAfter, the reference graphic item ID

4.22.18 CANACTIVATE (CTR0349)

This function tests if control can be activated.

C/C++/C# definition:

```
bool CanActivate();
```

Return:
true if the control can be activated

Arguments
none

4.22.19 GETACTIVATION (CTR0350)

This function gets the activation status.

C/C++/C# definition:

```
bool GetActivation();
```

Return:
true if the control is activated

Arguments
none

4.22.20 SETACTIVATION (CTR0351)

This function enables/disables the activation. When graphic control is activated during debug, the value are displayed at screen but items cannot be modified (moved or property changed).

C/C++/C# definition:

```
void SetActivation(bool bActive);
```

Return:
none

Arguments
bActive, true to activate the control

4.22.21 SETUSERID (CTR0352)

This function changes the name of the selected item.

C/C++ definition:

```
void SetUserID(LPCTSTR szUserID);
```

C# definition:

```
void SetUserID(string szUserID);
```

Return:
none

Arguments
szUserID, change the user ID (identifier, not graphic ID) of the selected item

4.22.22 GETLINK (CTR0353)

When parent window receives the notification “W5EDITN_LINKTO”, call this function to retrieve the link value of the clicked link object.

C/C++ definition:

```
LPCTSTR GetLink();
```

C# definition:

```
string GetLink();
```

Return:
The link path of the last link graphic object, empty if no link selected

Arguments
none

4.23 Current step and breakpoints

4.23.1 RESETSTEPPOS (CTR0354)

This function resets the current step position of step by step debugging.

C/C++/C# definition:

```
bool ResetStepPos();
```

Return:
returns true if step position has been reset

Arguments
none

4.23.2 SETSTEPPOS (CTR0355)

This function sets the current step position of the step by step debugging.

C/C++ definition:

```
bool SetStepPos(LPCTSTR szLocation);
```

C# definition:

```
bool SetStepPos(string szLocation);
```

Return:
true if step position has been changed

Arguments
szLocation, the new position of current step in control (this is a compiler string format – see LocateError function)

4.23.3 REMOVEALLBKP (CTR0356)

This function removes all breakpoints.

C/C++/C# definition:

```
void RemoveAllBkp();
```

Return:
none

Arguments
none

4.23.4 SETBKP (CTR0357)

This function sets breakpoint at position given by szLocation.

C/C++ definition:

```
void SetBkp(LPCTSTR szLocation);
```

C# definition:

```
void SetBkp(string szLocation);
```

Return:
none

Arguments
szLocation, the location of the breakpoint in control (compiler format string)

4.23.5 HASBREAKPOINT (CTR0358)

This function tests there is breakpoint at current position.

C/C++/C# definition:

```
bool HasBreakpoint();
```

Return:
true if selection contains a breakpoint

Arguments
none

4.23.6 SETTRACEPOINT (CTR0359)

This function sets tracepoint at position given by "szLocation".

C/C++ definition:

```
void SetTracePoint(LPCTSTR szLocation);
```

C# definition:

```
void SetTracePoint(string szLocation);
```

Return:
none

Arguments
szLocation, the location of the trace point in control (compiler format string)

4.23.7 HASTRACEPOINT (CTR0360)

This function tests if caret has tracepoint.

C/C++/C# definition:

```
bool HasTracepoint();
```

Return:
true if selection contains a trace point

Arguments
none

4.23.8 SETBKPEX (CTR0361)

This function sets/unsets a breakpoint/tracepoint at position.

C/C++ definition:

```
bool SetBkpEx(LPCTSTR szLocation, DWORD dwType);
```

C# definition:

```
bool SetBkpEx(string szLocation, UInt32 dwType);
```

Return:

true if breakpoint or trace point at been set at location

Arguments

szLocation, the location of the breakpoint/tracepoint to add/remove in control (compiler format string)

dwType, the type of point to add/remove. Can take one of these values:

BKP_NONE	0	removes the current breakpoint/tracepoint
BKP_BREAKACTIVE	1	adds an active breakpoint
BKP_BREAKINACTIVE	2	adds an inactive breakpoint
BKP_TRACEACTIVE	3	adds an active tracepoint
BKP_TRACEINACTIVE	4	adds an inactive tracepoint

4.23.9 GETBKPLIST (CTR0362)

This function retrieves the breakpoint list in control.

C/C++ definition:

```
LPCTSTR GetBkpList();
```

C# definition:

```
string GetBkpList();
```

Return:

returns the list of breakpoints/tracepoints in compiler format string separate by ";"

Arguments

4.23.10 SETINSTANCE (CTR0363)

This function sets instance name for Function Block debugging. Used when controls is debugging an instance used in another program.

C/C++ definition:

```
void SetInstance(LPCTSTR szInstance);
```

C# definition:

```
void SetInstance(string szInstance);
```

Return:
none

Arguments
szInstance the new instance name

4.23.11 GETINSTANCE (CTR0364)

This function gets the current instance debugged of the step by step debugging.

C/C++ definition:

```
LPCTSTR GetInstance();
```

C# definition:

```
string GetInstance();
```

Return:
returns the instance name

Arguments
none

4.23.12 SETPARENT (CTR0365)

This function sets program caller for FB debugging. Used when controls is debugging an instance used in another parent program.

C/C++ definition:

```
void SetParent(LPCTSTR szParent);
```

C# definition:

```
void SetParent(string szParent);
```

Return:
none

Arguments
szParent, the name of the parent program

4.23.13 GETPARENT (CTR0366)

This function gets the current parent of the step by step debugging.

C/C++ definition:

```
LPCTSTR GetParent();
```

C# definition:

```
string GetParent();
```

Return:
the program parent name used in debug

Arguments
none

4.24 Item properties

4.24.1 SETPROPERTIES (CTR0367)

This function sets the properties of the selected item: name,value for activeX editor..

C/C++ definition:

```
void SetProperties(LPCTSTR szProp);
```

C# definition:

```
void SetProperties(string szProp);
```

Return:
none

Arguments
szProp, the new properties to set to the selected item

4.24.2 GETPROPERTIES (CTR0368)

This function gets the properties of the selected item:
name,value(type)@enum1|enum2|enum3|.....@ for activeX editor.

C/C++ definition:

```
LPCTSTR GetProperties();
```

C# definition:

```
string GetProperties();
```

Return:
returns the list of properties of the selected item

Arguments
none

4.24.3 SETGRAPHICPROPERTIES (CTR0369)

This function sets the properties of the selected graphic object : use the syntax for graphics.

C/C++ definition:

```
void SetGraphicProperties(LPCTSTR szGraObject, LPCTSTR szProperties);
```

C# definition:

```
void SetGraphicProperties(string szGraObject, string szProperties);
```

Return:
none

Arguments
szGraObject, this is the graphic object reference. This can take one of these values (see file X5ScriptApi.h, the values are X5_TYPExxx):

X5_TYPEMETER	"X5METER"
X5_TYPECLOUDS	"X5CLOUDS"
X5_TYPEBOOLBITMAP	"X5BITMAPBOOL"
X5_TYPEBITMAP	"X5BITMAP"
X5_TYPETEXTDISP	"X5TEXTDISP"
X5_TYPETEXTIN	"X5TEXTIN"
X5_TYPEPUSHBUTTON	"X5PUSHBUTTON"
X5_TYPEBARGRAPH	"X5BARGRAPH"
X5_TYPEANALOGBUTTON	"X5ANALOGBUTTON"
X5_TYPESCALE	"X5SCALE"
X5_TYPEDIGITALMETER	"X5DIGITALMETER"
X5_TYPEBUTTON	"X5BUTTON"
X5_TYPESPYCONNECTION	"X5SPYCONNECTION"
X5_TYPECHART	"X5CHART"
X5_TYPESHAPE	"X5SHAPE"
X5_TYPELINK	"X5LINK"
X5_TYPESLIDER	"X5SLIDER"
X5_TYPEROTATEBUTTON	"X5ROTATEBUTTON"
X5_TYPEEDITOR	"X5EDITOR"
X5_TYPESPYSYMBOL	"X5SPYSYMBOL"
X5_TYPEBITS	"X5BITS"
X5_TYPECOMBO	"X5COMBO"
X5_TYPEPANEL	"X5PANEL"

szProperties, the properties list (these list is returned by ACTX control when call “GETPROPERTIES”)

4.24.4 CANCHANGEPROPERTIES (CTR0370)

This function tests if selected item properties can be changed.

C/C++/C# definition:

```
bool CanChangeProperties();
```

Return:

true if the properties of the current selected item can be changed

Arguments

none

4.25 Sampling

4.25.1 NEEDSETUP (CTR0371)

This function tests if sampling need to be setup before start.

C/C++/C# definition:

```
bool NeedSetup();
```

Return:
true if needs to setup before start

Arguments
none

4.25.2 CANSTARTSAMPLING (CTR0372)

This function tests if sampling can start.

C/C++/C# definition:

```
bool CanStartSampling();
```

Return:
true if sampling can be started

Arguments
none

4.25.3 STARTSAMPLING (CTR0373)

This function starts the sampling.

C/C++/C# definition:

```
bool StartSampling();
```

Return:
true if sampling has been started

Arguments
none

4.25.4 CANSTOPSAMPLING (CTR0374)

This function tests if sampling can be stopped.

C/C++/C# definition:

```
bool CanStopSampling();
```

Return:
true if sampling can be stopped

Arguments

none

4.25.5 STOPSAMPLING (CTR0375)

This function stops the current sampling.

C/C++/C# definition:

```
bool StopSampling();
```

Return:
true if sampling has been stopped

Arguments
none

4.25.6 CANSETUPSAMPLING (CTR0376)

This function tests if sampling can be setup.

C/C++/C# definition:

```
bool CanSetupSampling();
```

Return:
true setup can be done on sampling

Arguments
none

4.25.7 ISAUTOSCROLL (CTR0377)

This function tests if autoscroll is active.

C/C++/C# definition:

```
bool IsAutoScroll();
```

Return:
true if auto-scroll is active

Arguments
none

4.25.8 ENABLEAUTOSCROLL (CTR0378)

This function enables/disables autoscroll.

C/C++/C# definition:

```
void EnableAutoScroll(bool bEnable);
```

Return:
none

Arguments

bEnable, true to enable the auto-scroll

4.26 Tree Lists

4.26.1 GETTREEPARENT (CTR0379)

This function retrieves the root parent of the current item.

C/C++ definition:

```
LPCTSTR GetTreeParent();
```

C# definition:

```
string GetTreeParent();
```

Return:
the parent string of the selected item

Arguments
none

4.26.2 FOCUSTREE (CTR0380)

This function forces the tree of TLcontrol to gain the focus.

C/C++/C# definition:

```
bool FocusTree();
```

Return:
true if control support this feature

Arguments
none

4.26.3 CANCOLLAPSEALL (CTR0381)

This function tests if all items can be collapsed.

C/C++/C# definition:

```
bool CanCollapseAll();
```

Return:
true if all items can be collapsed

Arguments
none

4.26.4 COLLAPSEALL (CTR0382)

This function collapses all items in tree.

C/C++/C# definition:

```
void CollapseAll();
```

Return:

none

Arguments
none

4.26.5 CANEXPANDALL (CTR0383)

This function tests if all items can be expanded.

C/C++/C# definition:

```
bool CanExpandAll();
```

Return:
true if all items can be expanded

Arguments
none

4.26.6 EXPANDALL (CTR0384)

This function expands all items in tree.

C/C++ definition:

```
void ExpandAll();
```

Return:
none

Arguments
none

4.26.7 CANEXPAND (CTR0385)

This function tests if selected item(s) can be expanded.

C/C++/C# definition:

```
bool CanExpand();
```

Return:
true if selected item(s) can be expanded

Arguments
none

4.26.8 EXPAND (CTR0386)

This function expands the selected item(s).

C/C++/C# definition:

```
void Expand();
```

Return:

none

Arguments
none

4.26.9 COLLAPSE (CTR0387)

This function collapses the selected item(s).

C/C++/C# definition:

```
void Collapse();
```

Return:
none

Arguments
none

4.26.10 SWAPCOLLAPSE (CTR0388)

This function toggle (expand <-> collapse) expand status of the selected item(s).

C/C++/C# definition:

```
void SwapCollapse();
```

Return:
none

Arguments
none

4.26.11 ISUNDEF (CTR0389)

This function gets the status of the section of code, true if undefined section.

C/C++/C# definition:

```
bool IsUndef();
```

Return:
true if selected item(s) are in a undef section

Arguments
none

4.26.12 SETUNDEF (CTR0390)

This function sets the undefined status to a section of code.

C/C++/C# definition:

```
void SetUndef(bool bUndef);
```

Return:

none

Arguments
bIgnore, true to undefined the selected section

4.26.13 CANUNDEF (CTR0614)

This function tests if the selected fieldbus node can be undefined.

C/C++/C# definition:

```
bool CanUndef();
```

Return:
true if selected item can be undefined (de-activated)

Arguments
none

4.26.14 SETPROGRAMNAME (CTR0391)

This function sets the program context for the tree.

C/C++ definition:

```
void SetProgramName(LPCTSTR szProgram);
```

C# definition:

```
void SetProgramName(string szProgram);
```

Return:
none

Arguments
szProgram, the context program name

4.26.15 CANMOVEDOWN (CTR0392)

This function test if selected item(s) can move down.

C/C++/C# definition:

```
bool CanMoveDown();
```

Return:
true if selected item(s) can be moved down

Arguments
none

4.26.16 MOVEDOWN (CTR0393)

This function moves down the selected item(s).

C/C++ definition:

```
HTREEITEM MoveDown();
```

C# definition:

```
IntPtr MoveDown();
```

Return:
the handle of the moved item

Arguments
none

4.26.17 CANMOVEUP (CTR0394)

This function tests if selected item(s) can move up.

C/C++/C# definition:

```
bool CanMoveUp();
```

Return:
true if selected item(s) can be moved up

Arguments
none

4.26.18 MOVEUP (CTR0395)

This function moves up the selected item.

C/C++ definition:

```
HTREEITEM MoveUp();
```

C# definition:

```
IntPtr MoveUp();
```

Return:
the handle of the moved item

Arguments
none

4.26.19 CANHEXDISPLAY (CTR0396)

This function tests if hexadecimal display can be set for this control.

C/C++/C# definition:

```
bool CanHexDisplay();
```

Return:
true if hexadecimal display can be set

Arguments
none

4.26.20 ISHEXDISPLAY (CTR0397)

This function tests if hexadecimal display is set.

C/C++/C# definition:

```
bool IsHexDisplay();
```

Return:
true if hexadecimal display is active

Arguments
none

4.26.21 SWAPHEXDISPLAY (CTR0398)

This function swaps hexadecimal/decimal display.

C/C++/C# definition:

```
void SwapHexDisplay();
```

Return:
none

Arguments
none

4.26.22 CANSORT (CTR0399)

This function tests if items can be sorted.

C/C++/C# definition:

```
bool CanSort(int iCol);
```

Return:
true is column can be sorted

Arguments
iCol, the column to sort

4.26.23 SORT (CTR0400)

This function sorts all items.

C/C++/C# definition:

```
void Sort(int iCol);
```

Return:
none

Arguments
iCol, the column to sort

4.26.24 INSERTITEM (CTR0401)

This function inserts a new tree item at current position or at the end.

C/C++ definition:

```
HTREEITEM InsertItem(BOOL bLast, LPCTSTR szItem);
```

C# definition:

```
IntPtr InsertItem(bool bLast, string szItem);
```

Return:
the handle of the new inserted item (null if insert failed)

Arguments
bLast, true if insert at the end of the tree, false if insert at current position
szItem, the item text to insert

4.26.25 CANSAVEVALUES (CTR0402)

This function tests if values can be saved in a new column.

C/C++/C# definition:

```
bool CanSaveValues();
```

Return:
true if values can be saved

Arguments
none

4.26.26 SAVEVALUES (CTR0403)

This function saves values in a new column.

C/C++ definition:

```
bool SaveValues(int iCol, LPCTSTR szCol);
```

C# definition:

```
bool SaveValues(int iCol, string szCol);
```

Return:
true if column has been saved

Arguments
iCol, the index of the new column to insert
szCol, the name of the new column

4.26.27 CANSENDRECEIPE (CTR0404)

This function tests if column recipe can be sent to target.

C/C++/C# definition:

```
bool CanSendReceipe(int iCol);
```

Return:

returns true if recipe can be sent

Arguments

iCol, the column index to send

4.26.28 SENDRECEIPE (CTR0405)

This function sends column recipe to target.

C/C++/C# definition:

```
bool SendReceipe(int iCol);
```

Return:

true if the recipe has been sent

Arguments

iCol, the column index to send

4.26.29 CANREMOVECOL (CTR0406)

This function tests if column can be removed.

C/C++/C# definition:

```
bool CanRemoveCol(int iCol);
```

Return:

true if column can be removed

Arguments

iCol, the column to remove

4.26.30 REMOVECOL (CTR0407)

This function removes column.

C/C++/C# definition:

```
bool RemoveCol(int iCol);
```

Return:

true if column has been removed

Arguments

iCol, the column to remove

4.26.31 CANINSERTCOL (CTR0408)

This function tests if a new column can be inserted.

C/C++/C# definition:

```
bool CanInsertCol(int iCol);
```

Return:

true if column can be inserted at position

Arguments

iCol, the index where to insert the new column

Can be one of these values:

W5F_NOCOL	-1	Insert new column at the end
W5F_CURRENTCOL	-2	Insert new colmun at current position
Other	>= 0	Insert colmun at specified index

4.26.32 INSERTCOL (CTR0409)

This function inserts a new column.

C/C++ definition:

```
int InsertCol(int iCol, LPCTSTR szCol, WORD wColType, WORD wColTypeEx);
```

C# definition:

```
int InsertCol(int iCol, string szCol, UInt16 wColType, UInt16 wColTypeEx);
```

Return:

return the index of the new inserted column (< 0 if insert failed)

Arguments

iCol, the index where to insert the new column (see possible values in “CanInsertCol”)

szCol, the column header text

wColType, describes the type of column to insert. These can be one of these values (see “W5EditTLApi.h”):

NO_TYPE	0	column type is not defined
TYPE_NAME	1	database, symbol ID must be set in the item data
TYPE_TYPE	2	database, symbol ID must be set in the item data
TYPE_DIM	3	database, symbol ID must be set in the item data
TYPE_ATTR	4	database, symbol ID must be set in the item data
TYPE_INIT	5	database, symbol ID must be set in the item data
TYPE_ALIAS	6	database, symbol ID must be set in the item data
TYPE_COMM	7	database, symbol ID must be set in the item data
TYPE_VALUE	8	middleware value
TYPE_SYB	9	database, symbol ID must be set in the item data
TYPE_PROPS	10	database, need N° property, symbol ID must be set in the item data
TYPE_CUSTOM	11	need IDCustom (see TC_xxx defines)

TYPE_ADD	12	need N° of added col (0 based!)
TYPE_ITEM	13	type depends of item (see TC_xxx defines)
TYPE_VARUSERGROUP	14	database, symbol ID must be set in the item data
TYPE_CONTENT	15	database, symbol ID must be set in the item data (for a program display content)
TYPE_CELL	16	the content of each cell has its own edit type

Types TYPE_CUSTOM, TYPE_ITEM and TYPE_CELL are different:

- TYPE_CUSTOM is used to set the entire column with one edit type
- TYPE_ITEM is used to set only one line with one edit type (when using “PropertyGrid” control)
- TYPE_CELL is used to set only one cell with one edit type, the caller has to set each edit type for each cell by default all cells are TC_EDIT

wColTypEx, describes the sub type of the column (in completion of wColType values TYPE_CUSTOM, TYPE_ITEM and TYPE_CELL). This can be one of these values (see “W5EditTLApi.h”):

TC_NONE	0	Item can not be edited
TC_EDIT	1	Item uses an simple edit box
TC_COLOR	2	Item uses a color box
TC_FILEPATH	3	Item uses a File dialog box
TC_ENUM	4	Item uses a list
TC_SELECTVAR	5	Item uses the database "select variable" box
TC_FONTNAME	6	Item uses the "Font Name" box
TC_FONTSIZE	7	Item uses the "Font size" box
TC_CHECK	8	Item displays a check box
TC_SPIN	9	Item uses a spin box
TC_BITMAP	10	Item display a bitmap
TC_PROGRESS	11	Item displays a progress bar
TC_ENUMEDIT	12	Item displays a editable enum list
TC_MAGNETO	13	is a mix of flags values: see TC_MAGNETO_xxx for possible values
TC_GRID2	14	used to edit 2 column values in same cell
TC_PASSWORD	15	Item uses a password box (and display **** instead of text)
TC_BASEFILE	16	used with mask (high word: see TLFILE_xxx for possible values)
TC_OEM	17	used to edit user cell
TC_NOTIF	18	used to test double click, editcell notif parent only
TC_GRID1	19	used to edit a list of values in same cell
TC_EDITMULTI	20	Special feature

4.26.33 CANRENAMECOL (CTR0410)

This function tests if column can be renamed.

C/C++/C# definition:

```
bool CanRenameCol(int iCol);
```

Return:
true if specified column can be renamed

Arguments
iCol, the column to renamed

4.26.34 RENAMECOL (CTR0411)

This function renames column.

C/C++/C# definition:

```
bool RenameCol(int iCol);
```

Return:
true if column has been renamed

Arguments
iCol, the column to renamed

4.26.35 CANCOPYCOL (CTR0412)

This function tests if column can be copied.

C/C++/C# definition:

```
bool CanCopyCol(int iCol);
```

Return:
true if column can be copied

Arguments
iCol, the column to copy

4.26.36 COPYCOL (CTR0413)

This function copies the column.

C/C++/C# definition:

```
bool CopyCol(int iCol);
```

Return:
true if column has been copied

Arguments
iCol, the column to copy

4.26.37 CANDECREASE (CTR0414)

This function tests if cell value can be decreased.

C/C++/C# definition:

```
bool CanDecrease();
```

Return:
true if selection can be decrease

Arguments
none

4.26.38 DECREASE (CTR0415)

This function decreases the selected cell value.

C/C++/C# definition:

```
void Decrease();
```

Return:
none

Arguments
none

4.26.39 CANINCREASE (CTR0416)

This function tests if cell value can be increased.

C/C++/C# definition:

```
bool CanIncrease();
```

Return:
true if selection can be increase

Arguments
none

4.26.40 INCREASE (CTR0417)

This function increases the selected cell value.

C/C++/C# definition:

```
void Increase();
```

Return:
none

Arguments
none

4.26.41 CANUNCHECK (CTR0418)

This function tests if the selected cell can be unchecked.

C/C++/C# definition:

```
bool CanUncheck();
```

Return:
true if selection can be unchecked

Arguments
none

4.26.42 UNCHECK (CTR0419)

This function unchecks the selected cell.

C/C++/C# definition:

```
void Uncheck();
```

Return:
none

Arguments
none

4.26.43 CANCHECKTREE (CTR0420)

This function tests if the selected cell can be checked.

C/C++/C# definition:

```
bool CanCheckTree();
```

Return:
true if selected item(s) can be check

Arguments
none

4.26.44 CHECKTREE (CTR0421)

This function checks the selected cell.

C/C++ definition:

```
LPCTSTR CheckTree();
```

C# definition:

```
string CheckTree();
```

Return:
returns the result of the check

Arguments
none

4.26.45 RESETVALUES (CTR0422)

This function applies the default values for all cells.

C/C++/C# definition:

```
void ResetValues();
```

Return:
none

Arguments
none

4.26.46 GETSORTEDCOL (CTR0423)

This function retrieves the sorted column (-1 if no sort).

C/C++/C# definition:

```
int GetSortedCol();
```

Return:
the index of the sorted column (-1 if no sort active)

Arguments
none

4.26.47 ISSORTASCENDING (CTR0424)

This function retrieves if sort is ascending or descending.

C/C++/C# definition:

```
bool IsSortAscending();
```

Return:
true if sorted column is sorted ascending

Arguments
none

4.26.48 SETITEMTEXT (CTR0425)

This function sets the text of the specified cell.

C/C++ definition:

```
bool SetItemText(HTREEITEM hItem, int iCol, LPCTSTR szText);
```

C# definition:

```
bool SetItemText(IntPtr hItem, int iCol, string szText);
```

Return:
true if set was succeeded

Arguments
hItem, the handle of the item to set text

iCol, the column index to set (0 based)
szText, the new text value to set

4.26.49 GETITEMTEXT (CTR0426)

This function gets the text in cell.

C/C++ definition:

```
LPCTSTR GetItemText(HTREEITEM hItem, int iCol);
```

C# definition:

```
string GetItemText(IntPtr hItem, int iCol);
```

Return:
the text in selected cell

Arguments
hItem, the handle of the item to get text
iCol, the column index to get (0 based)

4.26.50 SETCOLWIDTH (CTR0427)

This function sets the column width in pixel.

C/C++/C# definition:

```
void SetColWidth(int iCol, int iWidth);
```

Return:
none

Arguments
iCol, the column index to set (0 based)
iWidth, the new width of the column (in pixels)

4.26.51 GETCOLWIDTH (CTR0448)

This function retrieves the column width.

C/C++/C# definition:

```
int GetColWidth(int iCol);
```

Return:
the column width (in pixels)

Arguments
iCol, the column index

4.26.52 GETFIRSTITEM (CTR0428)

This function gets the root item in tree.

C/C++ definition:

```
HTREEITEM GetFirstItem();
```

C# definition:

```
IntPtr GetFirstItem();
```

Return:
the handle of the first item in tree (root item)

Arguments
none

4.26.53 GETNEXTLINEITEM (CTR0429)

This function gets the next line item in tree.

C/C++ definition:

```
HTREEITEM GetNextLineItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetNextLineItem(IntPtr hItem);
```

Return:
the handle of the next line item

Arguments
hItem, the handle of the tree item

4.26.54 GETPREVLINEITEM (CTR0457)

This function retrieves the previous line item in tree.

C/C++ definition:

```
HTREEITEM GetPrevLineItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetPrevLineItem(IntPtr hItem);
```

Return:
the handle of the previous line item of hItem

Arguments
hItem, the handle of the item tree

4.26.55 GETNEXTVISIBLEITEM (CTR0430)

This function retrieves next visible item of hItem.

C/C++ definition:

```
HTREEITEM GetNextVisibleItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetNextVisibleItem(IntPtr hItem);
```

Return:
the handle of the next visible item

Arguments
hItem, the handle of the item

4.26.56 EVTCHANGED (CTR0431)

This function sends message indicated that binding has changed.

C/C++/C# definition:

```
void EVTChanged();
```

Return:
none

Arguments
none

4.26.57 ISLOADED (CTR0432)

This function returns true if project is already loaded.

C/C++/C# definition:

```
bool IsLoaded();
```

Return:
true if project is loaded

Arguments
none

4.26.58 ISOUTOFDATE (CTR0433)

This function returns true if information displayed in control is not updated.

C/C++/C# definition:

```
bool IsOutOfDate();
```

Return:
true if out of date

Arguments
none

4.26.59 LOCKBINDING (CTR0434)

This function locks/unlocks binding for avoid changes.

C/C++/C# definition:

```
void LockBinding(bool bLock);
```

Return:
none

Arguments
bLock, true to lock the binding

4.26.60 REFRESH (CTR0435)

This function forces the redraw of tree.

C/C++/C# definition:

```
void Refresh();
```

Return:
none

Arguments
none

4.26.61 GETPRGID (CTR0436)

This function retrieves the program ID in program list.

C/C++ definition:

```
DWORD GetPrgID();
```

C# definition:

```
UInt32 GetPrgID();
```

Return:
the ID of the program

Arguments
none

4.26.62 GETPRGNAME (CTR0437)

This function retrieves the program name in program list.

C/C++ definition:

```
LPCTSTR GetPrgName();
```

C# definition:

```
string GetPrgName();
```

Return:
the program name

Arguments

none

4.26.63 CANMOVEPRG (CTR0438)

This function tests if program can be moved.
C/C++ definition:

```
bool CanMovePrg(WORD wMove);
```

C# definition:

```
bool CanMovePrg(UInt16 wMove);
```

Return:
true if the selected program can be moved

Arguments
wMove, can take one of these values (see TLMOVE_xxx in W5EditTLApi.h file for constants):

TLMOVE_UP	1	Move program up
TLMOVE_DOWN	2	Move program down
TLMOVE_LEFT	3	Move program to parent brother (for SFC children only)
TLMOVE_RIGHT	4	Move program child of the previous program (for SFC only)
TLMOVE_BEGIN	5	Move program to begin section
TLMOVE_END	6	Move program to end section

4.26.64 MOVEPRG (CTR0439)

This function moves program.

C/C++ definition:

```
bool MovePrg(WORD wMove);
```

C# definition:

```
bool MovePrg(UInt16 wMove);
```

Return:
true if program has been moved

Arguments
wMove, for possible value, see “CanMovePrg”.

4.26.65 CANADDPRG (CTR0440)

This function tests if program can be added.

C/C++ definition:

```
bool CanAddPrg(WORD wPrg);
```

C# definition:

```
bool CanAddPrg(UInt16 wPrg);
```

Return:

true if a new program can be added

Arguments
wPrg, can be one of these values (see W5EditTLApi.h):

TLPRG_PRGSFC	1
TLPRG_PRGSFCCHILD	2
TLPRG_PRGFBD	3
TLPRG_PRGLD	4
TLPRG_PRGST	5
TLPRG_PRGIL	6
TLPRG_UDFBFBD	7
TLPRG_UDFBLD	8
TLPRG_UDFBST	9
TLPRG_UDFBIL	10
TLPRG_SUBFBD	11
TLPRG_SUBLD	12
TLPRG_SUBST	13
TLPRG_SUBIL	14

4.26.66 ADDPRG (CTR0441)

This function inserts a new program in tree.

C/C++ definition:

```
bool AddPrg(WORD wPrg);
```

C# definition:

```
bool AddPrg(UInt16 wPrg);
```

Return:
true if program has been added

Arguments
wPrg, see “CanAddPrg” for possible values

4.26.67 CANRENAMEPRG (CTR0442)

This function tests if selected program can be renamed (only the container can rename program, not the control).

C/C++/C# definition:

```
bool CanRenamePrg();
```

Return:
true if selected program can be renamed

Arguments

none

4.26.68 CANCMDPRG (CTR0443)

This function tests if middleware command can be sent to selected program.

C/C++ definition:

```
bool CanCmdPrg(WORD wCommand);
```

C# definition:

```
bool CanCmdPrg(UInt16 wCommand);
```

Return:

true if command can be executed

Arguments

wCommand, is one of these values (see “W5EditTLApi.h”):

TLPRGCMD_START	1
TLPRGCMD_STOP	2
TLPRGCMD_PAUSE	3
TLPRGCMD_RESUME	4

4.26.69 CMDPRG (CTR0444)

This function sends command to selected program.

C/C++ definition:

```
bool CmdPrg(WORD wCommand);
```

C# definition:

```
bool CmdPrg(UInt16 wCommand);
```

Return:

true if command has been executed

Arguments

wCommand, see CanCmdPrg for possible values

4.26.70 CANOPENPRG (CTR0445)

This function tests if selected program can be opened.

C/C++/C# definition:

```
bool CanOpenPrg();
```

Return:

true if selected program can be open

Arguments

none

4.26.71 GETNBPRG (CTR0446)

This function retrieves the number of programs (low word)/UDFB (high word) in program list.

C/C++ definition:

```
long GetNbPrg();
```

C# definition:

```
Int32 GetNbPrg();
```

Return:
number of programs
number of main programs can be retrieved in low word
number of UDFB can be retrieved in high word

Arguments
none

4.26.72 CANCOPYPRG (CTR0447)

This function tests if program can be copied (only the container can copy program, not the control).

C/C++/C# definition:

```
bool CanCopyPrg();
```

Return:
retuns true if selected program can be copied

Arguments
none

4.26.73 GETCOLTYPE (CTR0449)

This function retrieves the type assigned to column (see W5EditTLApi.h for defines).

C/C++ definition:

```
WORD GetColType(int iCol);
```

C# definition:

```
UInt16 GetColType(int iCol);
```

Return:
the type of the specified column (see “InsertCol” for possible values)

Arguments
iCol, the column index

4.26.74 GETEXCOLTYPE (CTR0450)

This function retrieves the extended type assigned to column.

C/C++ definition:

```
WORD GetExColType(int iCol);
```

C# definition:

```
UInt16 GetExColType(int iCol);
```

Return:
the extended type of the specified column (see “InsertCol” for possible values)

Arguments
iCol, the column index

4.26.75 GETSELECTEDITEM (CTR0451)

This function retrieves the current selected item.

C/C++ definition:

```
HTREEITEM GetSelectedItem();
```

C# definition:

```
IntPtr GetSelectedItem();
```

Return:
the handle of the selected item (if any, null if not)

Arguments
none

4.26.76 GETCURRENTCOL (CTR0452)

This function retrieves the current selected column.

C/C++/C# definition:

```
int GetCurrentCol();
```

Return:
the index of the selected column

Arguments
none

4.26.77 GETITEMDATA (CTR0453)

This function gets the data associated to item.

C/C++ definition:

```
DWORD_PTR GetItemData(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetItemData(IntPtr hItem);
```

Return:
the item data contained in tree item

Arguments
hItem, the handle of the item

4.26.78 SETITEMDATA (CTR0454)

This function associates data to item.

C/C++ definition:

```
void SetItemData(HTREEITEM hItem, DWORD_PTR dwData);
```

C# definition:

```
void SetItemData(IntPtr hItem, IntPtr dwData);
```

Return:
none

Arguments
hItem, the handle of the item to specify
dwData, the data to set in hItem

4.26.79 GETNBCOL (CTR0455)

This function retrieves the number of columns.

C/C++/C# definition:

```
int GetNbCol();
```

Return:
the number of columns

Arguments
none

4.26.80 SETITEMCOLOR (CTR0456)

This function affects a preferred color to item.

C/C++ definition:

```
bool SetItemColor(HTREEITEM hItem, COLORREF rgb);
```

C# definition:

```
bool SetItemColor(IntPtr hItem, IntPtr rgb);
```

Return:
true if item has been set

Arguments
hItem, the handle of item to set color
rgb, the color to set (RGB value)

4.26.81 SETCURRENTCOL (CTR0458)

This function selects the column.

C/C++/C# definition:

```
void SetCurrentCol(int iCol);
```

Return:
none

Arguments
iCol, the index of the column to select

4.26.82 SETSORTCOL (CTR0459)

This function sorts the column ascending or descending.

C/C++/C# definition:

```
void SetSortCol(int iCol, bool bAscending);
```

Return:
none

Arguments
iCol, the column index of the sorted
bAscending, if true sorts column ascending

4.26.83 GETFIRSTSEL (CTR0460)

This function retrieves the first selected item in tree.

C/C++ definition:

```
HTREEITEM GetFirstSel();
```

C# definition:

```
IntPtr GetFirstSel();
```

Return:
the handle of the first selected item in tree

Arguments
none

4.26.84 GETNEXTSEL (CTR0461)

This function retrieves the next selected item in tree.

C/C++ definition:

```
HTREEITEM GetNextSel(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetNextSel(IntPtr hItem);
```

Return:
the handle of the next selected item in tree after hItem

Arguments
hItem, a selected item

4.26.85 INSERTITEMEX (CTR0462)

This function inserts a new item. The previous item and parent item can be specified.

C/C++ definition:

```
HTREEITEM InsertItemEx(LPCTSTR szItem, HTREEITEM hParent, HTREEITEM hInsertAfter);
```

C# definition:

```
IntPtr InsertItemEx(string szItem, IntPtr hParent, IntPtr hInsertAfter);
```

Return:
the new inserted item handle (null if insert failed)

Arguments
szItem, the name of the new inserted item
hParent, the handle of the parent item
hInsertAfter, the handle of the item where insert new item tree

4.26.86 SETITEMBOLD (CTR0463)

This function sets item bold/not bold.

C/C++ definition:

```
bool SetItemBold(HTREEITEM hItem, BOOL bBold);
```

C# definition:

```
bool SetItemBold(IntPtr hItem, bool bBold);
```

Return:
true if item can be set bold

Arguments
hItem, the handle of the item to set
bBold, true if item displayed bold

4.26.87 SETCOLITEMTEXT (CTR0464)

This function changes column text header.

C/C++ definition:

```
void SetColItemText(int iCol, LPCTSTR szCol);
```

C# definition:

```
void SetColItemText(int iCol, string szCol);
```

Return:
none

Arguments
iCol, the column to set
szCol, the header text to set

4.26.88 SETCOLITEMPARAM (CTR0465)

This function sets item data to column header.

C/C++ definition:

```
void SetColItemParam(int iCol, DWORD_PTR dwData);
```

C# definition:

```
void SetColItemParam(int iCol, IntPtr dwData);
```

Return:
none

Arguments
iCol, the index column
dwData, the data to associate with the column

4.26.89 GETCOLITEMPARAM (CTR0466)

This function gets item data associated to column header item.

C/C++ definition:

```
DWORD_PTR GetColItemParam(int iCol);
```

C# definition:

```
IntPtr GetColItemParam(int iCol);
```

Return:
the data associated with colmun

Arguments
iCol, the column to get

4.26.90 GETCOLITEMTEXT (CTR0467)

This function retrieves header column item text.

C/C++ definition:

```
LPCTSTR GetColItemText(int iCol);
```

C# definition:

```
string GetColItemText(int iCol);
```

Return:
the text of the specified column

Arguments
iCol, the column index

4.26.91 ENSUREVISIBLE (CTR0468)

This function ensures that item is visible.

C/C++ definition:

```
bool EnsureVisible(HTREEITEM hItem);
```

C# definition:

```
bool EnsureVisible(IntPtr hItem);
```

Return:
true if item is visible

Arguments
hItem, the item to set visible

4.26.92 GETPARENTITEM (CTR0469)

This function retrieves tree parent item.

C/C++ definition:

```
HTREEITEM GetParentItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetParentItem(IntPtr hItem);
```

Return:
handle of the parent item of hItem

Arguments
hItem, the handle of item tree

4.26.93 GETFIRSTVISIBLEITEM (CTR0572)

This function gets the first visible item in tree.

C/C++ definition:

```
HTREEITEM GetFirstVisibleItem();
```

C# definition:

```
IntPtr GetFirstVisibleItem();
```

Return:
handle of the first item in tree

Arguments
none

4.26.94 GETPREVSIBLINGITEM (CTR0470)

This function retrieve previous sibling tree item.

C/C++ definition:

```
HTREEITEM GetPrevSiblingItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetPrevSiblingItem(IntPtr hItem);
```

Return:
the handle of the previous sibling item of hItem (same level item)

Arguments
hItem, the handle of the tree item

4.26.95 GETNEXTSIBLINGITEM (CTR0471)

This function retrieves next sibling tree item.

C/C++ definition:

```
HTREEITEM GetNextSiblingItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetNextSiblingItem(IntPtr hItem);
```

Return:
the handle of the next sibling item of hItem (same level)

Arguments
hItem, the handle of the tree item

4.26.96 GETCHILDITEM (CTR0472)

This function retrieves first child tree item.

C/C++ definition:

```
HTREEITEM GetChildItem(HTREEITEM hItem);
```

C# definition:

```
IntPtr GetChildItem(IntPtr hItem);
```

Return:

the handle of the first child item of hItem (null if no child)

Arguments
hItem, the handle of the tree item

4.26.97 SETPROPGROUP (CTR0473)

This function sets group name of property.

C/C++ definition:

```
bool SetPropGroup(int iProp, LPCTSTR szGroup);
```

C# definition:

```
bool SetPropGroup(int iProp, string szGroup);
```

Return:
true if property group has been set

Arguments
iProp, the index of property
szGroup, the name of the property group

4.26.98 SETPROPVALUE (CTR0474)

This function sets the value of the property.

C/C++ definition:

```
bool SetPropValue(int iProp, LPCTSTR szValue);
```

C# definition:

```
bool SetPropValue(int iProp, string szValue);
```

Return:
true if value has been set

Arguments
iProp, the index of the property to set
szValue, the new value of the property

4.26.99 SETPROPNAME (CTR0475)

This function sets name of property.

C/C++ definition:

```
bool SetPropName(int iProp, LPCTSTR szName);
```

C# definition:

```
bool SetPropName(int iProp, string szName);
```

Return:
true if name has been set

Arguments
iProp, the index of the property to set
szName, the new name of the property

4.26.100 SETPROPREADONLY (CTR0476)

This function sets the read only status for property.

C/C++/C# definition:

```
bool SetPropReadOnly(int iProp, bool bReadOnly);
```

Return:
true if read only status has been set

Arguments
iProp, the index of the property to set
bReadOnly, true to set property read only

4.26.101 ISPROPREADONLY (CTR0477)

This function gets the read only status for property.

C/C++/C# definition:

```
bool IsPropReadOnly(int iProp);
```

Return:
true if property is in read only mode

Arguments
iProp, the index of the property to get

4.26.102 SETPROPHEADER (CTR0478)

This function sets the header name (internal name, not translated).

C/C++ definition:

```
bool SetPropHeader(int iProp, LPCTSTR szHeader);
```

C# definition:

```
bool SetPropHeader(int iProp, string szHeader);
```

Return:
true if property has been set

Arguments
iProp, the index of the property to set
szHeader, the text value of the header of property

4.26.103 GETPROPHEADER (CTR0479)

This function retrieves the internal name of property.

C/C++ definition:

```
LPCTSTR GetPropHeader(int iProp);
```

C# definition:

```
string GetPropHeader(int iProp);
```

Return:

returns the property header

Arguments

iProp, the index of the property to get

4.26.104 SETPROPTYPE (CTR0480)

This function sets the property type.

C/C++/C# definition:

```
bool SetPropType(int iProp, int iType);
```

Return:

true if type has been set

Arguments

iProp, the index of the property to set

iType, the type of property (method to edit property). Can be on of these values:

TC_NONE	0	Item can not be edited
TC_EDIT	1	Item uses an simple edit box
TC_COLOR	2	Item uses a color box
TC_FILEPATH	3	Item uses a File dialog box
TC_ENUM	4	Item uses a list
TC_SELECTVAR	5	Item uses the database "select variable" box
TC_FONTNAME	6	Item uses the "Font Name" box
TC_FONTSIZE	7	Item uses the "Font size" box
TC_CHECK	8	Item displays a check box
TC_SPIN	9	Item uses a spin box
TC_BITMAP	10	Item display a bitmap
TC_PROGRESS	11	Item displays a progress bar
TC_ENUMEDIT	12	Item displays a editable enum list
TC_MAGNETO	13	is a mix of flags values: see TC_MAGNETO_xxx for possible values
TC_GRID2	14	used to edit 2 column values in same cell
TC_PASSWORD	15	Item uses a password box (and display **** instead of text)
TC_BASEFILE	16	used with mask (high word: see TLFILE_xxx for possible values)
TC_OEM	17	used to edit user cell

TC_NOTIF	18	used to test double click, editcell notif parent only
TC_GRID1	19	used to edit a list of values in same cell
TC_EDITMULTI	20	Special feature

4.26.105 GETPROPTYPE (CTR0481)

This function retrieves the property type.

C/C++/C# definition:

```
int GetPropType(int iProp);
```

Return:

the edit type of the property (see SetPropType for possible values)

Arguments

iProp, the index of the property to get

4.26.106 SETPROPDESC (CTR0482)

This function retrieves the property description.

C/C++ definition:

```
bool SetPropDesc(int iProp, LPCTSTR szDesc);
```

C# definition:

```
bool SetPropDesc(int iProp, string szDesc);
```

Return:

return true if description has been set

Arguments

iProp, the index of the property to set
szDesc, the new description of the property

4.26.107 SETPROPMAXLEN (CTR0483)

This function sets the available length for property.

C/C++/C# definition:

```
bool SetPropMaxLen(int iProp, int iMaxLen);
```

Return:

true if max len has been set

Arguments

iProp, the index of the property to set
iMaxLen the maximum value text lenght

4.26.108 SETPROPMININT (CTR0484)

This function sets the available minimum value for spin properties.

C/C++/C# definition:

```
bool SetPropMinInt(int iProp, int iMinInt);
```

Return:
return true if minimum value has been set

Arguments
iProp, the index of the property to set
iMinInt the minimum value allowed

4.26.109 SETPROPMAXINT (CTR0485)

This function sets the available maximum value for spin properties.

C/C++/C# definition:

```
bool SetPropMaxInt(int iProp, int iMaxInt);
```

Return:
return true if maximum value has been set

Arguments
iProp, the index of the property to set
iMaxInt the maximum value allowed

4.26.110 SETPROPENUM (CTR0486)

This function sets the list of choice for the enumerated property.

C/C++ definition:

```
bool SetPropEnum(int iProp, LPCTSTR szEnum);
```

C# definition:

```
bool SetPropEnum(int iProp, string szEnum);
```

Return:
return true if enumerate values have been set

Arguments
iProp, the index of the property to set
szEnum, the enumerate values (list of values separate by ASCII character 0x02.

4.26.111 SETPROPFILTER (CTR0487)

This function sets the list of file types supported by the property. Available only when property has type TC_FILEPATH (set with “SetPropType”).

C/C++ definition:

```
bool SetPropFilter(int iProp, LPCTSTR szFilter);
```

C# definition:

```
bool SetPropFilter(int iProp, string szFilter);
```

Return:
true if the filter has been set

Arguments
iProp, the index of the property to set
szFilter, the list of filters, the format is: “text1 (*.ext1)|*.ext1|text2 (*.ext2)|*.ext2||””

4.26.112 GETPROPVALUE (CTR0488)

This function retrieves property value.

C/C++ definition:

```
LPCTSTR GetPropValue(int iProp);
```

C# definition:

```
string GetPropValue(int iProp);
```

Return:
get the value of the specified property

Arguments
iProp, the index of the property to get

4.26.113 GETPROPNAME (CTR0489)

This function retrieves property name.

C/C++ definition:

```
LPCTSTR GetPropName(int iProp);
```

C# definition:

```
string GetPropName(int iProp);
```

Return:
returns the name of the specified property

Arguments
iProp, the index of the property to get

4.26.114 SETUSED (CTR0490)

This function fills list of used blocks.

C/C++ definition:

```
void SetUsed(LPCTSTR szUsed);
```

C# definition:

```
void SetUsed(string szUsed);
```

Return:
none

Arguments
list of blocks (each block is separated by tab - \t)

4.26.115 SORTTREE (CTR0491)

This function sorts the tree with sorting method and reset the init flag.

C/C++ definition:

```
void SortTree(WORD wSortMethod, bool bInit);
```

C# definition:

```
void SortTree(UInt16 wSortMethod, bool bInit);
```

Return:
none

Arguments
wSortMethod, the sort method. Can be one of these values:

TLSORT_NONE	0	No sort method
TLSORT_ALPHABETIC	1	Alphabetic method
TLSORT_EXEC	2	Execution order (for program lists)

bInit, no longer used

4.26.116 GETSORTMETHOD (CTR0492)

This function retrieves the sorting method.

C/C++ definition:

```
WORD GetSortMethod();
```

C# definition:

```
UInt16 GetSortMethod();
```

Return:
returns the sorting method used by tree (see SortTree for available values)

Arguments
none

4.26.117 GETPROGRAMID (CTR0493)

This function retrieves the selected program ID.

C/C++ definition:

```
DWORD GetProgramID();
```

C# definition:

```
UInt32 GetProgramID();
```

Return:
the database selected program handle

Arguments
none

4.26.118 GETFOLDERID (CTR0494)

This function retrieves the selected folder ID.

C/C++ definition:

```
DWORD GetFolderID();
```

C# definition:

```
UInt32 GetFolderID();
```

Return:
the database selected folder handle

Arguments
none

4.26.119 GETSCREENID (CTR0495)

This function retrieves the selected screen ID (embedded HMI).

C/C++ definition:

```
DWORD GetScreenID();
```

C# definition:

```
UInt32 GetScreenID();
```

Return:
the database selected screen handle

Arguments
none

4.26.120 SETSCREENID (CTR0496)

This function sets the screen ID.

C/C++ definition:

```
void SetScreenID(DWORD dwID);
```

C# definition:

```
void SetScreenID(UInt32 dwID);
```

Return:

none

Arguments
dwID, the screen ID to set

4.26.121 CANEDITPROPERTIES (CTR0497)

This function tests if properties can be edited on selected item.

C/C++/C# definition:

```
bool CanEditProperties();
```

Return:
true if properties of the selected item can be edited

Arguments
none

4.26.122 EDITPROPERTIES (CTR0498)

This function edits properties of the selected item.

C/C++/C# definition:

```
bool EditProperties();
```

Return:
true if properties have been edited

Arguments
none

4.26.123 CANEDITPARAMETERS (CTR0499)

This function tests if parameters can be edited on the selected item.

C/C++/C# definition:

```
bool CanEditParameters();
```

Return:
true if parameters of the selected item can be edited

Arguments
none

4.26.124 EDITPARAMETERS (CTR0500)

This function edits parameters of the selected item.

C/C++/C# definition:

```
void EditParameters();
```

Return:

none

Arguments
none

4.26.125 CANCREATEFILE (CTR0501)

This function tests if a new file same as selected item can be created.

C/C++ definition:

```
bool CanCreateFile(DWORD dwFileType);
```

C# definition:

```
bool CanCreateFile(UInt32 dwFileType);
```

Return:
returns true if a new file can be created in tree

Arguments
dwFileType, the type of file to be added. Can be one of these values:

TLFILE_NONE	0x0000
TLFILE_SPY	0x0001
TLFILE_RECIPE	0x0002
TLFILE_GRAPHICS	0x0004
TLFILE_SIGNAL	0x0008
TLFILE_STRINGTABLE	0x0010
TLFILE_CURVE	0x0020
TLFILE_IEC850	0x0040

4.26.126 CREATEFILE (CTR0502)

This function creates a new file of the same type as selected item.

C/C++ definition:

```
bool CreateFile(DWORD dwFileType);
```

C# definition:

```
bool CreateFile(UInt32 dwFileType);
```

Return:
true if new file has been created

Arguments
dwFileType, the type of file to be added (see “CanCreateFile” for possible values).

4.26.127 ADDFILETYPES (CTR0503)

This function creates the folders to get the files types.

C/C++ definition:

```
void AddFileTypes(DWORD dwFileTypes);
```

C# definition:

```
void AddFileTypes(UInt32 dwFileTypes);
```

Return:
none

Arguments
dwFileTypes, type of files supported by folders. Can be a **combination** of values in "CanCreateFile").

4.26.128 GETFILEID (CTR0504)

This function retrieves the selected file ID.

C/C++ definition:

```
DWORD GetFileID();
```

C# definition:

```
UInt32 GetFileID();
```

Return:
return the database handle of the selected file

Arguments
none

4.26.129 GETFILESECTION (CTR0505)

This function retrieves the type of section/file selected.

C/C++ definition:

```
DWORD GetFileSection();
```

C# definition:

```
UInt32 GetFileSection();
```

Return:
returns one of these values:

TLFILE_NONE	0x0000
TLFILE_SPY	0x0001
TLFILE_RECIPE	0x0002
TLFILE_GRAPHICS	0x0004
TLFILE_SIGNAL	0x0008
TLFILE_STRINGTABLE	0x0010
TLFILE_CURVE	0x0020
TLFILE_IEC850	0x0040

Arguments
none

4.26.130 SETREDRAW (CTR0506)

This function avoids redraw of each item for a lot of sequential modifications.

C/C++/C# definition:

```
bool SetRedraw(bool bRedraw);
```

Return:
returns true if function is supported by control

Arguments
bRedraw, true if redraw is set

4.26.131 SETERRORMODE (CTR0507)

This function displays errors in red.

C/C++/C# definition:

```
void SetErrorMode(bool bErrorMode);
```

Return:
none

Arguments
bErrorMode, true to set the error mode

4.26.132 GETPRINTABLETEXT (CTR0508)

This function retrieves the printable text for text print.

C/C++ definition:

```
LPCTSTR GetPrintableText();
```

C# definition:

```
string GetPrintableText();
```

Return:
returns the text formatted ready to be printed

Arguments
none

4.26.133 NOTIFCHANGES (CTR0509)

This function blocks the notification W5EDITN_MODIFIED when control content changed.

C/C++/C# definition:

```
bool NotifChanges(bool bNotif);
```

Return:
true if function is supported by control

Arguments
bNotif, if true, parent window will be notified by modifications in control

4.26.134 CANINSERTNETWORK (CTR0510)

This function tests if a network can be inserted at the current place.

C/C++/C# definition:

```
bool CanInsertNetwork();
```

Return:
true if network can be inserted

Arguments
none

4.26.135 INSERTNETWORK (CTR0511)

This function inserts a new network at the current place.

C/C++/C# definition:

```
void InsertNetwork();
```

Return:
none

Arguments
none

4.26.136 CANINSERTMASTERPORT (CTR0512)

This function tests if a master or port can be inserted at the current place.

C/C++/C# definition:

```
bool CanInsertMasterPort();
```

Return:
true if master or port can be inserted

Arguments
none

4.26.137 INSERTMASTERPORT (CTR0513)

This function inserts a new master/port at the current place.

C/C++/C# definition:

```
void InsertMasterPort();
```

Return:
none

Arguments
none

4.26.138 CANINSERTSLAVEREQUEST (CTR0514)

This function tests if a slave or group can be inserted at the current place.

C/C++/C# definition:

```
bool CanInsertSlaveRequest();
```

Return:
true if can insert

Arguments
none

4.26.139 INSERTSLAVEREQUEST (CTR0515)

This function inserts a slave/group at the current place.

C/C++/C# definition:

```
void InsertSlaveRequest();
```

Return:
none

Arguments
none

4.26.140 LOCKCONFIG (CTR0516)

This function locks/unlocks the current fieldbus configuration.

C/C++/C# definition:

```
void LockConfig(bool bLock);
```

Return:
none

Arguments
bLock, true to lock configuration

4.26.141 GETKEY (CTR0517)

This function retrieves the key for the help section.

C/C++ definition:

```
LPCTSTR GetKey();
```

C# definition:

```
string GetKey();
```

Return:
the key value for the help section

Arguments
none

4.26.142 CANEXECCOMMAND (CTR0518)

This function tests if added command can be executed.

C/C++/C# definition:

```
bool CanExecCommand(int iCommand);
```

Return:
true if command can be executed

Arguments
iCommand, the command index to test

4.26.143 EXECCOMMAND (CTR0519)

This function executes the added command.

C/C++/C# definition:

```
bool ExecCommand(int iCommand);
```

Return:
true if command has modified the control content

Arguments
iCommand, the index of command to execute

4.26.144 GETNBCOMMAND (CTR0520)

This function gets the number of available added commands.

C/C++/C# definition:

```
int GetNbCommand();
```

Return:
the number of added commands

Arguments
none

4.26.145 GETCOMMAND (CTR0521)

This function retrieves the added command text.

C/C++ definition:

```
LPCTSTR GetCommand(int iCommand);
```

C# definition:

```
string GetCommand(int iCommand);
```

Return:
the text of the command

Arguments
iCommand, the index of command to get

4.26.146 CANEXPORT (CTR0522)

This function tests if tree can be exported.

C/C++/C# definition:

```
bool CanExport();
```

Return:
true if tree can be exported

Arguments
none

4.26.147 EXPORT (CTR0523)

This function exports the current tree.

C/C++/C# definition:

```
void Export();
```

Return:
none

Arguments
none

4.26.148 CANIMPORT (CTR0524)

This function tests if tree can be imported.

C/C++/C# definition:

```
bool CanImport();
```

Return:
true if tree can be imported

Arguments
none

4.26.149 IMPORT (CTR0525)

This function imports new tree content.

C/C++/C# definition:

```
void Import();
```

Return:
none

Arguments
none

4.26.150 CANEXPORTTREE (CTR0526)

This function test if tree can be exported as 'fb5' file.

C/C++/C# definition:

```
bool CanExportTree();
```

Return:
true if tree can be exported

Arguments
none

4.26.151 EXPORTTREE (CTR0527)

This function exports tree in 'fb5' file.

C/C++/C# definition:

```
void ExportTree();
```

Return:
none

Arguments
none

4.26.152 CANIMPORTTREE (CTR0528)

This function tests if tree can be imported from a 'fb5' file.

C/C++/C# definition:

```
bool CanImportTree();
```

Return:
true if tree can be imported

Arguments
none

4.26.153 IMPORTTREE (CTR0529)

This function imports FB from 'fb5' file.

C/C++/C# definition:

```
void ImportTree();
```

Return:
none

Arguments
none

4.26.154 GETERROR (CTR0530)

This function gets the current error from selected node.

C/C++ definition:

```
LPCTSTR GetError();
```

C# definition:

```
string GetError();
```

Return:
returns the error string of the selected node

Arguments
none

4.26.155 GETLASTCHAR (CTR0531)

This function retrieves the last char hit in control.

C/C++/C# definition:

```
int GetLastChar();
```

Return:
returns the ascii code of the last char

Arguments
none

4.26.156 GETLASTKEYDOWN (CTR0532)

This function retrieves the last key hit in control.

C/C++/C# definition:

```
int GetLastKeyDown();
```

Return:
returns the ascii of the last key down (not char)

Arguments
none

4.26.157 SETSTYLE (CTR0533)

This function changes the grid style.

C/C++ definition:

```
void SetStyle(DWORD dwStyle);
```

C# definition:

```
void SetStyle(UInt32 dwStyle);
```

Return:
none

Arguments
can be a combination of these values (defines are in W5EditTLApi.h):

TL_HIDEHEADER	0x0001	hide the header line
TL_DISABLEDND	0x0002	disable drag n drop feature
TL_NOTOOLTIP	0x0004	hide tooltips
TL_NOHORZSCROLL	0x0008	no horizontal scroll bar
TL_COMMENTWITHNAME	0x0010	display comments with name
TL_MULTISEL	0x0020	allows multiselection
TL_ONESELSELECTION	0x0040	allows one cell selection (not full line)
TL_HASLINES	0x0080	show lines in tree
TL_LINESATROOT	0x0100	show item as child of root item
TL_HASBUTTONS	0x0200	displays cross to expand/collapse
TL_SHOWSELALWAYS	0x0400	display selection even if tree haven't focus
TL_DISABLEHEADERRESIZE	0x0800	can not resize the header columns with mouse
TL_AUTOEDIT	0x1000	if this style is set, the standard treelist open its own dialog boxes to enter values
TL_HOTHEADER	0x2000	indicates that header is hot paint when user can change col selected
TL_DISABLEHEADERVERTRESIZE	0x4000	can not resize vertical column to zoom content
TL_HIDELINESEL	0x8000	do not display selection in tree
TL_SHOWMARGIN	0x00010000	Used to show left margin

4.26.158 REMOVESTYLE (CTR0573)

This function removes the specified grid styles (defines are in W5EditTLApi.h)

C/C++ definition:

```
void RemoveStyle(DWORD dwStyle);
```

C# definition:

```
void RemoveStyle(UInt32 dwStyle);
```

Return:

None

Arguments
can be a combination of these values (defines are in W5EditTLApi.h):

TL_HIDEHEADER	0x0001	hide the header line
TL_DISABLEDND	0x0002	disable drag n drop feature
TL_NOTOOLTIP	0x0004	hide tooltips
TL_NOHORZSCROLL	0x0008	no horizontal scroll bar
TL_COMMENTWITHNAME	0x0010	display comments with name
TL_MULTISEL	0x0020	allows multiselection
TL_ONESELSELECTION	0x0040	allows one cell selection (not full line)
TL_HASLINES	0x0080	show lines in tree
TL_LINESATROOT	0x0100	show item as child of root item
TL_HASBUTTONS	0x0200	displays cross to expand/collapse
TL_SHOWSELALWAYS	0x0400	display selection even if tree haven't focus
TL_DISABLEHEADERRESIZE	0x0800	cannot resize the header columns with mouse
TL_AUTOEDIT	0x1000	if this style is set, the standard treelist open its own dialog boxes to enter values
TL_HOTHEADER	0x2000	indicates that header is hot paint when user can change col selected
TL_DISABLEHEADERVERTRESIZE	0x4000	cannot resize vertical column to zoom content
TL_HIDELINESEL	0x8000	do not display selection in tree

4.26.159 GETCONTENTWIDTH (CTR0536)

This function retrieves the width of the document in pixel size (including zoom).

C/C++/C# definition:

```
int GetContentWidth();
```

Return:
returns the document width

Arguments
none

4.26.160 GETCONTENTHEIGHT (CTR0537)

This function retrieves the height of the document in pixel size (including zoom).

C/C++/C# definition:

```
int GetContentHeight();
```

Return:
returns the document height

Arguments
none

4.26.161 SETPROPENUMEDIT (CTR0538)

This function sets the list of choice for the enumerated property, enum list is editable.

C/C++ definition:

```
bool SetPropEnumEdit(int iProp, LPCTSTR szEnum);
```

C# definition:

```
bool SetPropEnumEdit(int iProp, string szEnum);
```

Return:
true if property has been set

Arguments
iProp, the index of the property to set
szEnum, the enumerate values (list of values separate by ASCII character 0x02

4.26.162 GETTARGETMARK (CTR0539)

This function gets the target mark of the drag n drop operation.

C/C++ definition:

```
bool GetTargetMark(void* pTargetMark);
```

C# definition:

```
bool GetTargetMark(IntPtr pTargetMark);
```

This function is not supported by C# wrapper!

Return:
true if a target mark exists during drag n drop operation

Arguments
pTargetMark, the pointer to the structure str_W5TLTargetMark (see file “W5EditTLApi.h” for definition)

4.26.163 CANCREATEITEM (CTR0540)

This function tests if a new item can be inserted at current position

C/C++/C# definition:

```
bool CanCreateItem(int iType, int iSubType);
```

Return:
true if a new item can be created

Arguments
iType is the type of item to create see “GetItemType” function (see W5TYPE_xxx for types in file “W5EditTLApi.h”).
iSubType, if the sub type of the item. If item is W5TYPE_EXTERN, the value can be one of these values:

W5SUBTYPE_NONE	0	No sub type
W5SUBTYPE_WEB	1	A WEB type
W5SUBTYPE_OEM	2	A types determinate by oem
W5SUBTYPE_OTHER	3	All other types

4.26.164 CREATEITEM (CTR0541)

This function inserts a new item at current position.

C/C++ definition:

```
bool CreateItem(int iType, int iSubType, LPCTSTR szDll);
```

C# definition:

```
bool CreateItem(int iType, int iSubType, string szDll);
```

Return:

true if new item has been create

Arguments

iType is the type of item to create see “GetItemType” function (see W5TYPE_xxx for types in file “W5EditTLApi.h”).
iSubType, if the sub type of the item. See function “CanCreateItem” for possible values
szDll, is the oem DLL used to create new item (can be empty). This dll is used only when iType = W5TYPE_EXTERN and iSubType = W5SUBTYPE_OEM.

4.26.165 EDITINPLACE (CTR0542)

This function sets the control edit in place mode.

C/C++/C# definition:

```
void EditInPlace(bool bEdit);
```

Return:

none

Arguments

bEdit, true to set edit in place mode

4.26.166 MOVEITEMAFTER (CTR0543)

This function moves a tree item after another, returned the new item.

C/C++ definition:

```
HTREEITEM MoveItemAfter(HTREEITEM hTarget, HTREEITEM hItem);
```

C# definition:

```
IntPtr MoveItemAfter(IntPtr hTarget, IntPtr hItem);
```

Return:

the handle of the move item

Arguments
hTarget, the hItem will be moved after hTarget
hItem, this hItem will be moved after hTarget

4.26.167 ISITEMEXPANDED (CTR0544)

This function tests if an item is expanded.

C/C++ definition:

```
bool IsItemExpanded(HTREEITEM hItem);
```

C# definition:

```
bool IsItemExpanded(IntPtr hItem);
```

Return:
true if item hItem is expanded

Arguments
hItem, the item handle to test

4.26.168 EXPANDITEM (CTR0545)

This function expands or collapses a specific item.

C/C++ definition:

```
void ExpandItem(HTREEITEM hItem, bool bExpand);
```

C# definition:

```
void ExpandItem(IntPtr hItem, bool bExpand);
```

Return:
none

Arguments
hItem, the item to expand or collapse
bExpand, true to expand, false to collapse

4.26.169 SETTREEICON (CTR0546)

This function sets a new image list of bitmaps for tree.

C/C++ definition:

```
bool SetTreeIcon(HBITMAP hBmp, int cx);
```

C# definition:

```
bool SetTreeIcon(IntPtr hBmp, int cx);
```

Return:
true if the image list has been set

Arguments
hBmp, the handle of the bitmap (beware this handle must be alive as long as the tree uses it)
cx, width of one image

Remarks:
Once this feature has been called, the function “SetItemImage” can be used to specify offset of one image in image list

4.26.170 LOCKITEMTYPE (CTR0547)

This function locks/unlocks the item type in control.

C/C++ definition:

```
bool LockItemType(BOOL bLock, DWORD dwPrj, DWORD dwItemType, LPCTSTR szPath);
```

C# definition:

```
BOOL LockItemType(bool bLock, UInt32 dwPrj, UInt32 dwItemType, string szPath);
```

Return:
Bool, true if item status has changed

Arguments
bLock, true to lock item
dwPrj, the handle of the item project
dwItemType, the type of object to lock/unlock (see “GetItemType” for possible values)
szPath, the full path of the item if extern item (used only when dwItemType = W5TYPE_EXTERN)

4.26.171 GETITEMSUBTYPE (CTR0548)

This function returns subtype of selected item (used in “multiprj” control for extern items).

C/C++ definition:

```
DWORD GetItemSubType();
```

C# definition:

```
UInt32 GetItemSubType();
```

Return:
returns the sub type of the selected item. Can be one of these values:

W5SUBTYPE_NONE	0	No sub type
W5SUBTYPE_WEB	1	A WEB type
W5SUBTYPE_OEM	2	A types determinate by oem
W5SUBTYPE_OTHER	3	All other types

Arguments
none

4.26.172 SETITEMIMAGE (CTR0549)

This function sets the image offset for the current item (see SetTreeIcon).

C/C++ definition:

```
bool SetItemImage(HTREEITEM hItem, int iImage);
```

C# definition:

```
bool SetItemImage(IntPtr hItem, int iImage);
```

Return:
true if image has been set

Arguments
hItem, the item to set image
iImage, the image offset in image list (set by function “SetTreeIcon”).

4.26.173 SETITEMTOOLTIP (CTR0550)

This function sets the tooltip for the specified item.

C/C++ definition:

```
bool SetItemTooltip(HTREEITEM hItem, LPCTSTR szTooltip);
```

C# definition:

```
bool SetItemTooltip(IntPtr hItem, string szTooltip);
```

Return:
true if tooltip has been set

Arguments
hItem, the item to set
szTooltip, the tooltip to set (for full line only)

4.26.174 SETITEMBITMAP (CTR0551)

This function sets bitmap to item (caller is responsible of creation and destruction of the bitmap object).

C/C++ definition:

```
bool SetItemBitmap(HTREEITEM hItem, HBITMAP hBmp);
```

C# definition:

```
bool SetItemBitmap(IntPtr hItem, IntPtr hBmp);
```

Return:
true if bitmap has been set

Arguments
hItem, the item to set bitmap
hBmp, the handle of the bitmap to set (must be alive as long as the tree uses it)

4.26.175 DELETEITEM (CTR0552)

This function deletes the item in tree.

C/C++ definition:

```
bool DeleteItem(HTREEITEM hItem);
```

C# definition:

```
bool DeleteItem(IntPtr hItem);
```

Return:

true if item has been deleted

Arguments

hItem, the item to delete

4.26.176 DLGCREATEVAR (CTR0553)

This function creates a new variable in dictionary with dialog (can set name and input/output flags).

C/C++ definition:

```
bool DlgCreateVar(LPCTSTR szVar, DWORD dwFlags);
```

C# definition:

```
bool DlgCreateVar(string szVar, UInt32 dwFlags);
```

Return:

true if new var has been created in control

Arguments

szVar, the variable name (can be empty, the control will display dialog box to choose name)
dwFlags, the variable creation flags. Can be a combination of values (see K5DBApi.h file: K5DBVAR_xxx values)

4.26.177 CANGENERATESHARED (CTR0554)

This function tests if shared memory can be generate from fieldbus.

C/C++/C# definition:

```
bool CanGenerateShared();
```

Return:

true if can generate

Arguments

none

4.26.178 GENERATESHARED (CTR0555)

This function generates shared memory from selected fieldbus.

C/C++/C# definition:

```
int GenerateShared();
```

Return:

int, can be one of these results (see “W5EditApi.h” file for defines):

K5GENSHARED_ERROR	0	error during generation
K5GENSHARED_OK	1	success
K5GENSHARED_CANCEL	2	cancel by user during generation

Arguments

None

4.26.179 SETITEMBREAKPOINT (CTR0575)

This function sets a breakpoint on the specified item (see W5EditAPi.h for dwBkp values BKP_xxx)

C/C++ definition:

```
bool SetItemBreakpoint(HTREEITEM hItem, DWORD dwBkpType) ;
```

C# definition:

```
bool SetItemBreakpoint(IntPtr hItem, UInt32 dwBkpType);
```

Return:

true breakpoint has been set

Arguments

hItem: the item to set. If the item is null, the breakpoint is apply to all items.
dwBkpType: the type of point to add/remove. Can take one of these values:

BKP_NONE	0	removes the current breakpoint/tracepoint
BKP_BREAKACTIVE	1	adds an active breakpoint
BKP_BREAKINACTIVE	2	adds an inactive breakpoint
BKP_TRACEACTIVE	3	adds an active tracepoint
BKP_TRACEINACTIVE	4	adds an inactive tracepoint

Remarks

To display breakpoint in left margin, the tree list must have TL_SHOWMARGIN style when created.

4.26.180 SETITEMCURPOS (CTR0576)

This function sets a breakpoint on the specified item (see W5EditAPi.h for dwBkp values BKP_xxx)

C/C++ definition:

```
bool SetItemCurPos(HTREEITEM hItem);
```

C# definition:

```
bool SetItemCurPos(IntPtr hItem);
```

Return:
True if current step position has been set (old position is clear)

Arguments
hItem: the item to set. If item is null, step position is cleared, not set.

Remarks
To display current step position in left margin, the tree list must have TL_SHOWMARGIN style when created.

4.26.181 SELECTTREEITEM (CTR0580)

This function selects the specified tree item.

C/C++ definition:

```
bool SelectTreeItem(HTREEITEM hItem, BOOL bDeselectAll);
```

C# definition:

```
bool SelectTreeItem(IntPtr hItem, bool bDeselectAll) ;
```

Return:
true if the selection has been set

Arguments
hItem: the item to set. If the item is null, the selection is removed.
bDeselectAll: if true, old selection is remove before select the new item

4.26.182 DEBUGPROJECT (CTR0581)

This function starts or stop debug mode of the specified project project in tree (the tree must support several projects).

C/C++ definition:

```
void DebugProject(DWORD dwPrj, BOOL bDebug);
```

C# definition:

```
void DebugProject(Uint32 dwPrj, bool bDebug);
```

Return:
void

Arguments:
dwPrj: this is the project handle. This handle is used to start/stop the debug mode of the project in tree
bDebug: the debug mode (true for actived, false for desactived)

Remarks:
Only tree list that support several projects can use this feature.

4.26.183 ENABLENOTIF (CTR0582)

This function enables/disables notifications from control (see K5N_xxx constants in W5EditTLApi.h include file)

C/C++ definition:

```
bool EnableNotif(WORD wNotif, bool bEnable);
```

C# definition:

```
bool EnableNotif(Uint16 wNotif, bool bEnable);
```

Return:
bool, true if at least one notification has been enabled/disabled

Arguments:
wNotif: this is the notification to enable/disable (see W5EDITN_xxx for possible values)
bEnable: this enable or disable the notification

Ex:
To enable all notifications, use EnableNotif(0, TRUE).
To disable al notification, use EnableNotif(0, FALSE).
To enable W5EDITN_SELCHANGE notification, use EnableNotif(W5EDITN_SELCHANGE, TRUE).

4.26.184 SELECTITEMTYPE (CTR0584)

This function selects a specific item in tree list.

C/C++ definition:

```
HTREEITEM SelectItemType(DWORD dwPrj, DWORD dwItemType, DWORD dwDBObject, LPCTSTR
szPath);
```

C# definition:

```
IntPtr EnableNotif(Uint32 dwPrj, Uint32 dwItemType, Uint32 dwDBObject, string szPath);
```

Return:
HTREEITEM, not NULL if a item has been found and selected

Arguments:
dwPrj: the project handle (database handle). Can be 0 for global items
dwItemType: the object type (see the possible values: W5TYPE_xxx in function GetItemType)
dwDBObject: the object handle (database handle). Can be 0 for items not in database
szPath: the full path of item. This should be empty if dwDBObject is valid

4.26.185 SETITEMIMAGESTATUS (CTR0585)

This function set a mask status over item image. This status indicates that item is open or lock.

C/C++ definition:

```
bool SetItemImageStatus(HTREEITEM hItem, DWORD dwStatus);
```

C# definition:

```
bool SetItemImageStatus (IntPtr hItem, Uint32 dwStatus);
```

Return:

bool, true if image status has been set

Arguments:

hItem, the item to set image status

dwStatus, the mask status

dwStatus is a mask of values behind(from W5EditApi.h):

W5ITEMMASKBMP_NONE	No mask used
W5ITEMMASKBMP_LOCK	The item is open (key)
W5ITEMMASKBMP_SC	The item is lock by source control

4.27 SAMA

4.27.1 CANPREVIEW (CTR0556)

This function tests if preview box can be displayed.

C/C++/C# definition:

```
bool CanPreview();
```

Return:
true if preview box can be displayed

Arguments
none

4.27.2 ISPREVIEW (CTR0557)

This function tests if preview box is displayed.

C/C++/C# definition:

```
bool IsPreview();
```

Return:
true if preview box is displayed

Arguments
none

4.27.3 TOGGLEPREVIEW (CTR0558)

This function toggles (show/hide) preview box.

C/C++/C# definition:

```
void TogglePreview();
```

Return:
none

Arguments
none

4.27.4 SETMAXWIDTH (CTR0559)

This function sets the maximum width that document can contain.

C/C++/C# definition:

```
void SetMaxWidth(int iMaxWidth);
```

Return:
none

Arguments

iMaxWidth, the new maximum width allowed by document

4.27.5 SETMAXHEIGHT (CTR0560)

This function sets the maximum height that document can contain.

C/C++/C# definition:

```
void SetMaxHeight(int iMaxHeight);
```

Return:
none

Arguments
iMaxHeight, the new maximum height allowed by document

4.28 HELP

4.28.1 HELP (CTR0561)

This function retrieves list of all commands supported by DLL.

C/C++ definition:

```
LPCTSTR Help(LPCTSTR szCsvFile);
```

C# definition:

```
string Help(string szCsvFile);
```

Return:
the help string about all commands

Arguments
szCsvFile, is the path of file where to save result (if szCsvFile is not null, save result in file)

4.28.2 HELPON (CTR0562)

This function returns the help on the selected command.

C/C++ definition:

```
LPCTSTR HelpOn(LPCTSTR szCommand);
```

C# definition:

```
string HelpOn(string szCommand);
```

Return:
the help about the command “szCommand”

Arguments
szCommand, the command to retrieve help about

4.28.3 GETINTERFACES (CTR0563)

This function retrieves list of all commands supported by DLL, return in wType format (see W5EditApi.h).

C/C++ definition:

```
LPCTSTR GetInterfaces(WORD wStyle);
```

C# definition:

```
string GetInterfaces(UInt16 wStyle);
```

Return:
the list of interfaces supported by all controls (the format depends on value of wStyle)

Arguments
wStyle, the style of interfaces to build. Can be one of these values:

WRAPTYPE_CSV	0

WRAPTYPE_C	1
WRAPTYPE_CS	2
WRAPTYPE_XML	3

5 Annexes

5.1 Control status when editing and debugging

When control is in edit mode and in debug mode, it should assume that read only mode is active depending to the control status.
Set debug state can be made using **SetDebug** command and read only state can be changed calling **SetReadOnly** command.

	Editing	Debugging	
ST	W	RO	
LD	W	RO	
FBD	W	RO	
SFC and FFSFC (level 1)	W	RO	
Dictionary	W	RO	
SpyList	W	W	
		Active	Not active
Graphics	W	RO	W
Graphic property grid	W	RO	W

5.2 Exchange text between SFC level1 and SFC level2

One important thing to understand is that level1 document contains list of all SFC items and level2 source code also. The level2 content has to be edited by another control than SFC. This other control gets information from SFC control and saves information in the SFC control. Only SFC control can load/save both level1 and level2 contents on disk.

Here is the description of what level2 contains:
A SFC item contains a list of qualifiers:

- STEP contains: the action Block, P0, N, P1 and notes
- TRANSITION contains: the transition code and notes
- COMMENT contains: only notes

Action block and notes can contain text only.
P0, N and P1 can be programmed in LD, ST or FBD.
Transition code can be programmed in ST or LD.

Steps to follow when editing level2 of a SFC item:

1. When user double click on a level1 object(STEP, TRANSITION or COMMENT)

SFC control will receive one of these notifications:

W5EDITN_DBLCLK
W5EDITN_SFCOPENDEFAULT
W5EDITN_SFCOPENNOTES
W5EDITN_SFCOPENP1
W5EDITN_SFCOPENP0
W5EDITN_SFCOPENN

2. First you have to **lock** the level1 item using LockStep, LockTrans or LockComment.

To determinates which kind of object is selected, call
IsSelStep, IsSelTrans, IsSelComment. These functions return id of the selected item and 0 if item type is not the good one.

3. Open qualifier specified by notification and selected item:
   Ex: if you receive W5EDITN_SFCOPENP1 and item is step, you have to open the P1 content of the step.

4. To determinate which language the qualifier uses, call one of:
   GetStepLanguageN
   GetStepLanguageP1
   GetStepLanguageP0
   GetTransLanguage

   These functions return one of:
   - W5EDITSFCLV2_NONE
   - W5EDITSFCLV2_STIL
   - W5EDITSFCLV2_LD
   - W5EDITSFCLV2_FBD

Depending to the language returned, create the corresponding control for edit the level2 content.

5. To get the qualifier content, use one of (string is returned)
   GetStepCode_Def
   GetStepCode_P1
   GetStepCode_N
   GetStepCode_P0
   GetStepNote
   GetTransCode
   GetTransNote
   GetCommentNote

6. Once you have the string, fill level2 control with the text using SetText(s)
7. When user have done its modifications in level2 control, you have to save qualifier content in SFC control:
   Obtain level2 content using s = Ctrl.GetText()
   Save the level2 in SFC using one of:
   - SETSTEPCODE_DEF
   - SETSTEPCODE_P1
   - SETSTEPCODE_N
   - SETSTEPCODE_P0
   - SETSTEPNOTE
   - SETTRANSCODE
   - SETTRANSNOTE

SETMACRONOTE
SETCOMMENTNOTE

8. When user closes the level2 control, you have to **unlock** open item in SFC using LockStep, LockTrans, LockMacro or LockComment.

Changing qualifier language when a qualifier is open:
You can change language in qualifiers P0, P1 and N only if there are empty.

5.3 Supported drag'n drop formats

Here is the list of mainly supported drag n drop formats.
These formats are used to exchange data:

- between straton widgets/form/controls
- can be used to exchange data between your application and straton widgets (you have to register these formats in your application before use it)

RegisterClipboardFormat	String format	Description
"CF_K5DBEDIT"	id:varname	id: is the database id of the variable (not mandatory) and varname is the name of variable is used from dictionary to all languages to drag n drop variable
"CF_K5FBD"	FBD source gra file contents	is used to drag n drop FBD items from one FBD program to another or to itself is used in editor activeX to drag n drop from library of activeX objects to the editor (content of a gra file
"CF_K5LD"	LD source	is used to drag n drop LD items from one LD program to another or to itself
"CF_K5FFLD"	FFLD source	is used to drag n drop Free Form Ladder items form one FFLD program to another or to itself
"CF_K5SFC"	SFC level1 source	is used to drag n drop SFC items from one SFC program to another or to itself
"CF_DBNVARSPY"	id1:varname1\r\nid2:varname2….	is used to drag n drop multiple variables
"CF_K5FUNCBLOCK"	blockname	blockname: is the name of block to be dragged is used to drag n drop block from FB library to all other languages (except SFC)
"CF_K5FUNCBLOCKEX"	blockname,nbinput,nboutput,bInstanciable	is used to drop an imported block (xk5 file)

5.4 Using structures in C# wrappers

A particular way of working when using the C# wrapper is to pass structures in parameter.

Take the example of SetTooltipInfos.

1. First of all, you will have to declare the structure itself (to retrieve structure field, please refer to W5EditApi.h file):

```
[StructLayout(LayoutKind.Sequential)]
public struct str_W5TooltipInfos
{
   public Boolean bName;
   public Boolean bType;
   public Boolean bDim;
   public Boolean bAttrib;
   public Boolean bProps;
   public Boolean bSyb;
   public Boolean bInitValue;
   public Boolean bTag;
   public Boolean bDesc;
}
```

2. Second is to declare the managed object and pass it to the command:

```
W5.Structures.str_W5TooltipInfos info = new W5.Structures.str_W5TooltipInfos();
info. bName = true;
info.bType = true;
…//set all the structure fields
IntPtr ptrInfo = IntPtr.Zero;
try
{
   ptrInfo = Marshal.AllocHGlobal(Marshal.SizeOf(info));
   Marshal.StructureToPtr(info, ptrInfo, false);
   bOK = ctrl.SetTooltipInfos(true, ptrInfo);
}
catch
{
   bOK = false;
}
finally
{
   Marshal.FreeHGlobal(ptrInfo);
}
```

6 FAQ

6.1 How can I save and restore scroll positions when I open a program?

This has to be done using the "*GetScrollPos*" and "*SetScrollPos*" Windows standard functions.
When user is about to close a program, call "*GetScrollPos*" for vertical and horizontal scroll bars and serialize values (in text file for example).
When program opens and content has been loaded (using "*Load*" command) AND the control size has been set, the scroll position can be restored from your serialization using the "*SetScrollPos*" standard function.
NB: when restoring it is very important that the editing control has its correct size.

7 Command References

E

F

G

H

I

K

L

M

N

P

R

S

T