UI API Function Library

Here list the UI API functions for the following objects:

Bitmap Object Functions

1 BitmapDeleteBitmap

Purpose This function is used to delete/release the memory of the bitmap object in a specified form. This function should be called after BitmapEraseBitmap function.

Prototype Err BitmapDeleteBitmap(ObjectID bitmap_id)

Scope Application/.Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment This function only frees the memory, but not erase the bitmap object on the screen. Although the resource of the object is erased, the screen still displays the Bitmap as before. BitmapEraseBitmap should be called first to erase the object from screen and avoid any invalid action on this object.

2 BitmapDrawBitmap

Purpose This function draws the specified bitmap on the display.

Prototype Err BitmapDrawBitmap(ObjectID bitmap_id)

Scope Application / Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment The attribute bitmap_drawn is set to TRUE in order to show that the bitmap object is already drawn on the display.

3 BitmapEraseBitmap

Purpose This function is used to erase the bitmap object that is on the display.

Prototype Err BitmapEraseBitmap(ObjectID bitmap_id)

Scope Application / Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment The memory, which is used to hold the bitmap object, will not be released. The attribute bitmap_drawn is cleared

4 BitmapGetAttribute

Purpose This function is used to get the attributes of the specified bitmap object.

Prototype Err BitmapGetAttribute(ObjectID bitmap_id, BOOLEAN *att_drawn, BOOLEAN *att_enable, BOOLEAN *att_active, BOOLEAN *att_visible)

Scope Application / Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters

att_drawn	Pointer to boolean variable to show whether the bitmap is already drawn on the display or not
att_enable	Pointer to boolean variable to show whether the bitmap is already enabled on the display or not
att_active	Pointer to boolean variable to show whether the bitmap is already active on the display or not
att_visible	Pointer to boolean variable to show whether the bitmap is already visible on the display or not

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

5 BitmapGetBitmapBounds

Purpose This function is called in order to get the bounds of a bitmap object

Prototype Err BitmapGetBitmapBounds(ObjectID bitmap_id, ObjectBounds *bitmap_bounds)

Scope Application/.Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters	bitmap_bounds	The pointer to the new bounds of the bitmap object
Result	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found
Comment	None	

6 BitmapGetBitmapTemplate

Purpose	This function is called in order to get the two bitmap templates for a bitmap object	
Prototype	Err BitmapGetBitmapTemplate(ObjectID bitmap_id, BitmapTemplate bitmap_template_0, BitmapTemplate bitmap_template_1)	
Scope	Application / Internal	
Input Parameters	bitmap_id	The ID value of the specified line object .
Output Parameters	bitmap_template_0	Pointer to the BitmapTemplate structure for bitmap1 of a bitmap object
	bitmap_template_1	Pointer to the BitmapTemplate structure for bitmap2 of a bitmap object
Result	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found
Comment	NULL can be passed-in if application does not want to get that bitmap_template data	

7 BitmapInitBitmap

Purpose This function is used to load the data structure and initialize it from a resource file into the RAM. *BitmapInitBitmap* function is usually followed by the *BitmapDrawBitmap* in order to draw the bitmap object on the display.

Prototype Err BitmapInitBitmap(ObjectID bitmap_id)

Scope Application / Internal

Input Parameters

bitmap_id	The ID value of the specified line object .

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment This function will not draw the bitmap object on the display

8 BitmapSetAttribute

Purpose This function is set the attributes of a bitmap object.

Prototype Err BitmapSetAttribute(ObjectID bitmap_id, BOOLEAN att_drawn, BOOLEAN att_enable, BOOLEAN att_active, BOOLEAN att_visible)

Scope Application/Internal

Input Parameters

bitmap_id	The ID value of the specified line object .
att_drawn	Boolean value to set up the state of the bitmap_drawn attribute
att_enable	Boolean value to set up the state of the bitmap_enable attribute
att_active	Boolean value to set up the state of the bitmap_active attribute

	att_visible	Boolean value to set up the state of the bitmap_visible attribute

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

9 BitmapSetBitmapBounds

Purpose This function is called in order to set the bounds of a bitmap object

Prototype Err BitmapSetBitmapBounds(ObjectID bitmap_id, ObjectBounds bitmap_bounds)

Scope Application/.Internal

Input Parameters

bitmap_id	The ID value of the specified line object .
bitmap_bounds	The new bounds of the bitmap object

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

10 BitmapSetBitmapTemplate

Purpose This function is called in order to set the two bitmap templates for a bitmap object

Prototype Err BitmapSetBitmapTemplate(ObjectID bitmap_id, BitmapTemplate bitmap_template_0, BitmapTemplate bitmap_template_1)

Scope Application / Internal

Input Parameters

bitmap_id	The ID value of the specified line object .
bitmap_template_0	The BitmapTemplate structure for bitmap1 of a bitmap object
bitmap_template_1	The BitmapTemplate structure for bitmap2 of a bitmap object

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

Clipboard Object Functions

1 ClipboardEraseItem

Purpose This function is used to clear all the content in the clipboard. After the call, the data_type of clipboard will be set to CLIP_EMPTY and the pointer to the data will be freed

Prototype Err ClipboardEraseItem( )

Scope Application/Internal

Input Parameters None

Output Parameters None

Result

TRUE	Success. (always TRUE)

Comment None

2 ClipboardGetItem

Purpose This function is used to get the item that is already copied to the clipboard

Prototype void *ClipboardGetItem(BYTE *data_type, WORD *data_size)

Scope Application / Internal

Input Parameters None

Output Parameters

data_type	Pointer to a data_type variable to show the type of data that is stored in the clipboard
data_size	Pointer to a data_size variable to show the size of the data that is stored in the clipboard

Result The return is a void pointer that points to the item that is stored in the clipboard. If there is no data in the clipboard, then the returned void pointer should be equal to NULL.

Comment The clipboard will make a copy of the data. It means that after the data has been used, the data pointer should be freed by the application.

3 ClipboardPutItem

Purpose This function is used to add item (text, bitmap or binary data) to the clipboard. This function should be called when text, bitmap or binary is highlighted and copied.

Prototype Err ClipboardPutItem(BYTE data_type, void *data_ptr, WORD data_size)

Scope Application/Internal

Input Parameters

data_type	the type of item that is stored in the clipboard 0 = CLIP_EMPTY, 1 = CLIP_TEXT_DATA 2 = CLIP_BITMAP_DATA 3 = CLIP_BINARY_DATA
data_ptr	pointer to the data that is put into the clipboard
data_size	the size of the data in terms of number of BYTE

Output Parameters None

Result

TRUE	Success. (always TRUE)

Comment The clipboard will make its own copy of the passed-in pointer if the

data_type is CLIP_TEXT_DATA. If not, the data_ptr cannot be destroyed after the data is put into the clipboard. It means that if something is required to be put into the clipboard and the data_type is not CLIP_TEXT_DATA, a copy of the data must be prepared for the clipboard.

Control Object Functions

1 ControlDeleteControl

Purpose This function is used to delete the specified control object from memory in order to free and release the occupied memory.

Prototype Err ControlDeleteControl (ObjectID control_id)

Scope Application

Input parameters

control_id	The ID value of the specified control object.

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments Delete the memory block which contains the specified control object. Although the resource of the object is deleted, the screen still displays the control as before. *ControlEraseControl* should be called first to erase the object from screen and avoid any invalid action on this object.

2 ControlDrawArc

Purpose This function is used to draw 4 arc for the round corner of the specified control object in a specified form.

Prototype
```
void ControlDrawArc ( ObjectBounds *bounds,
                      BOOLEAN draw_fill,
                      SHORT draw_radius,
                      BYTE bg_color )
```

Scope Internal

Input parameters

bounds	Pointer to the bounds of the control object.
draw_fill	Boolean value to indicate fill in the arc or not.
draw_radius	Radius of the arc.
bg_color	Background color of the arc.

Output parameters None

Return None

Comments None

3 ControlDrawControl

Purpose This function is used to draw the specified control object in a specified form. Before calling this function, the *ControlInitControl* function should be called first in order to initialise the control object from the resource file.

Prototype
```
Err ControlDrawControl (ObjectID control_id)
```

Scope Application / Internal

Input parameters

control_id	The ID value of the specified control object.

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments After successful drawing of the control object, the control_drawn attribute is set.

4 ControlDrawEmptyButton

Purpose This function is used to draw 4 lines of the specified control's button according to the boundary.

Prototype
```
void ControlDrawEmptyButton ( ObjectBounds *bounds,
                              SHORT draw_radius,
                              BYTE bg_color )
```

Scope Internal

Input parameters

bounds	Pointer to the bounds of the control object.
draw_radius	Radius of the arc.
bg_color	Background color of the arc

Output parameters None

Return None

Comments Actually this function will only draw 2 vertical and 2 horizontal lines on the display, to compare an empty button *ControlDrawArc* will then be called.

5 ControlDrawTapButton

Purpose This function is used to draw 4 lines and 4 round corners of the specified control's button with filled color according to the boundary.

Prototype
```
void ControlDrawTapButton(Control *addr,
                          ObjectBounds *bounds,
                          SHORT draw_radius,
                          BYTE bg_color)
```

Scope Internal

Input parameters

bounds	Pointer to the bounds of the control object.
draw_radius	Radius of the arc.
bg_color	Background color of the arc.

Output parameters None

Return None

Comments Actually this function will only draw 2 vertical and 2 horizontal lines on the display, to compare an empty button *ControlDrawArc* will then be called.

6 ControlDrawString

Purpose This function is used to draw 4 lines of the specified control's button according to the boundary.

Prototype

```
void ControlDrawString ( Control *addr,
                         ObjectBounds *bounds,
                         BYTE draw_color,
                         BYTE *draw_text,
                         BYTE bg_color )
```

Scope Internal

Input parameters

addr	Pointer to the control structure.
bounds	Pointer to the bounds of the control object.
draw_color	Color of the text.
draw_text	Pointer to the character
bg_color	Background color of the arc.

Output parameters None

Return None

Comments None

7 ControlEraseControl

Purpose This function is used to erase a control object in a form.

Prototype Err ControlEraseControl (ObjectID control_id)

Scope Application / Internal

Input parameters

control_id	The ID value of the specified control object.

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments None

8 ControlGetAttributes

Purpose The function is called in order to retrieve all 5 attributes for a control object. The attributes are control_enable, control_drawn, control_save_behind and control_active.

Prototype

```
Err ControlGetAttr ( ObjectID control_id,
                     BOOLEAN *  enable_attr,
                     BOOLEAN * drawn_attr,
                     BOOLEAN * savebehind_attr,
                     BOOLEAN * active_attr,
                     BOOLEAN * visible_attr )
```

Scope Application / Internal

Input parameters None

Output parameters

control_id	The ID value of the specified control object.
enable_attr	Boolean value that to indicate the state of the control_enable attribute.
drawn_attr	Boolean value that to indicate the state of the control_drawn attribute.
savebehind_attr	Boolean value that to indicate the state of the control_save_behind attribute.
active_attr	Boolean value that indicate the state of the control_active attribute.
visible_attr	Boolean value that set up the state of the control_visible attribute.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments None

9 ControlGetCheckedCheckbox

Purpose This function is called in order to get the object ID of the checkbox (with the desired group id) that is in the ON state.

Prototype

```
Err ControlGetCheckedCheckbox(ObjectID form_id,
                              USHORT group_id,
                              ObjectID *control_id)
```

Scope Application / Internal

Input parameters

form_id	The ID value of the specified FORM object.

Output parameters

group_id	The group id of the wanted checkbox
*control_id	Pointer to a object id value of the checkbox (with the desired group id) that is in the ON state

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_NO_OBJECT_FOUND	There is no object with the

		same group id or the object with the same group id is not a checkbox or the state is OFF

Comments None

10 ControlGetLabel

Purpose Return a pointer to the control's text.

Prototype Err ControlGetLabel (ObjectID control_id, BYTE **control_label)

Scope Application / Internal

Input parameters

control_id	The ID value of the specified control object.

Output parameters

control_label	Pointer reference to the address that points to the control's text.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments None

11 ControlGetPushedPushButton

Purpose This function is called in order to get the object ID of the push button (with the desired group id) that is in the ON state.

Prototype Err ControlGetPushedPushButton(ObjectID form_id,
USHORT group_id,
ObjectID *control_id)

Scope Application / Internal

Input parameters

form_id	The ID value of the specified FORM object.

Output parameters

group_id	The group id of the wanted pushbutton
*control_id	Pointer to a object id value of the pushbutton (with the desired group id) that is in the ON state

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_NO_OBJECT_FOUND	There is no object with the same group id or the object with the same group id is not a push button or the state is OFF

Comments None

12 ControlHighlightOneItem

Purpose This function is called to highlight one item or un-highlight one item of the popup popup trigger.

Prototype

```
Err ControlHighlightOneItem (Control *addr,
                             SHORT item_num,
                             BOOLEAN item_onoff)
```

Scope Internal

Input parameters

addr	Pointer to control object
item_num	the item number of the item that wants to be highlight or un-highlight
item_onoff	TRUE if highlight FALSE if un-highlight

Output parameters None

Return

TRUE	Success
FALSE	Not handled

Comments None

13 ControlHitControl

Purpose This function is simulate tapping a control's object and adds a EVT_CONTROL_SELECT to the event queue.

Prototype Err ControlHitControl (ObjectID control_id)

Scope Application

Input parameters

control_id	The ID value of the specified control object.

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments This function is useful for testing.

14 ControlInitControl

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *ControlInitControl* is usually followed by the *ControlDrawControl* in order to draw the control object on the display.

Prototype Err ControlIInitControl (ObjectID control_id)

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_CANT_CREATE_LOOKUP_TABLE	
ERR_UI_TYPE_MISMATCH	
ERR_UI_CANT_CREATE_POINTER	
ERR_RES_OBJ_MISS	

Comment This function will not draw the control object on the display.

15 ControlPopupClickedRegion

Purpose Return the clicked region of the popuped trigger.

Prototype
```
USHORT ControlPopupClickedRegion ( Control *addr,
                                   SHORT x_input,
                                   SHORT y_input )
```

Scope Application

Input parameters

addr	Pointer to the control structure.
x_input	X-coordinate of the pen.
y_input	Y-coordinate of the pen.

Output parameters None

Return

REGION_ITEMS	Popup item region.
REGION_UP_ARROW	Upper arrow region
REGION_DN_ARROW	Lower arrow region

Comments If the number of items in the popup list are more than the screen can display, then an upper or a lower arrow will appear in the upper / lower right corner of the particular popup list.

16 ControlPopupDeleteAllItems

Purpose This function is called for deleting all the items in a popup trigger object.

Prototype Err ControlPopupDeleteAllItems(ObjectID control_id)

Scope Application/Internal

Input Parameters

control_id	The ID value of the control object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The item number is invalid

Comment All the items are deleted. The related parameters are also updated correspondingly.

17 ControlPopupDeleteItem

Purpose This function is used to delete an item in the popup list and all the items will then be arranged.

Prototype Err ControlPopupDeleteItem (ObjectID control_id
USHORT item_number)

Scope Application

Input Parameters

control_id	The ID value of the specified control object .
item_number	The item number that you want to delete.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_ITEM_NUMBER_EXCEED	Invalid item number.
ERR_UI_OBJECT_NOT_MATCH	

Comment None

18 ControlPopupFindItemNum

Purpose To search out the item number in a popup trigger object by providing the text of the item.

Prototype **Err ControlPopupFindItemNum(ObjectID control_id, BYTE *text,
USHORT *item_num);**

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object .
text	Pointer to the text of the wanted item

Output Parameters

item_num	Pointer to the value of item number

Result	TRUE	item with the text is found
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
	ERR_UI_OBJECT_NOT_MATCH	The ID value is belongs to a control object, but the object is not a popup trigger object
	FALSE	no item found

Comment None

19 ControlPopupGetCurrentNumOfDisplayedItems

Purpose This function is used to obtain the number of items in a specified popuped trigger.

Prototype

Err ControlPopupGetCurrentNumOfDisplayedItems (ObjectID control_id,
USHORT *items_displayed)

Scope Application / Internal

Input Parameters	control_id	The ID value of the specified control object .
Output Parameters	items_displayed	Return is an unsigned integer that represents the number of the items in the specified popuped object.

Result	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
	ERR_UI_OBJECT_NOT_MATCH	

Comment None

20 ControlPopupGetPopupItems

Purpose To get the text of a specific popup item.

Prototype **Err ControlPopupGetPopupItems (ObjectID control_id,**
USHORT item_number,
BYTE **item_text)

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object .
item_number	The item number that you want to get the text.

Output Parameters

items_text	Pointer pointed to the char array of the item text.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_ITEM_NUMBER_EXCEED	Invalid item number.
ERR_UI_OBJECT_NOT_MATCH	

Comment None

21 ControlPopupGetSelectedItem

Purpose This function is used to obtain the item number of the selected item in a popup trigger object.

Prototype Err ControlPopupGetSelectedItem (ObjectID control_id,
USHORT *selected_item)

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object .

Output Parameters

selected_item	Pointer to the item number of the selected item

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_OBJECT_NOT_MATCH	

Comment If there is no selected item, then the returned value is NO_SELECTION.

22 ControlPopupGetTopItemNumber

Purpose To get the top item number of a specific popup trigger.

Prototype **Err ControlPopupGetTopItemNumber (ObjectID control_id,**
USHORT *top_num)

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object .

Output Parameters

top_num	Return the top item number of a specific popup trigger.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_OBJECT_NOT_MATCH	

Comment None

23 ControlPopupGetTotalItems

Purpose This function is used to obtain the total numbers of items in a specified popuped trigger.

Prototype Err ControlPopupGetTotalItems (ObjectID control_id,
USHORT *total_num_items)

Scope Application / Internal

Input Parameters

control_id	The ID value of the specified control object .

Output Parameters

total_num_items	Return is an unsigned integer that represents the total numbers of the items in the specified popuped trigger.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_OBJECT_NOT_MATCH	

Comment None

24 ControlPopupInsertItem

Purpose To insert a new item in the popup list and the whole list will then be re-arranged.

Prototype

```
Err ControlPopupInsertItems ( ObjectID control_id,
                              USHORT item_number,
                              BYTE *item_text )
```

Scope Application

Input Parameters

control_id	The ID value of the specified control object .
item_number	The number that you want to add item to.
item_text	Pointer to the char array of item text.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_OBJECT_NOT_MATCH	Invalid item number

Comment None

25 ControlPopupKeyboardHandleControlPopup

Purpose This function is called to handle the KEY_UP, KEY_DOWN, KEY_PAGEUP and KEY_PAGEDOWN when the popup trigger is popup

Prototype Err ControlPopupKeyboardHandleControlPopup(EvtType *Event)

Scope Application

Input Parameters	Event	Pointer to event
Output Parameters	None	
Result	TRUE FALSE	Success Not handled
Comment	None	

26 ControlPopupPopupTrigger

Purpose	This function is called to popup a popup trigger by application	
Prototype	Err ControlPopupPopupTrigger(ObjectID control_id)	
Scope	Application	
Input Parameters	control_id	ID of control object
Output Parameters	None	
Result	TRUE FALSE ERR_UI_RES_NOT_FOUND	Success Not handled Resource not found
Comment	None	

27 ControlPopupSetSelectedItem

Purpose	This function is used to set the item number of the selected item in a popup trigger object.	
Prototype	Err ControlPopupGetSelectedItem (ObjectID control_id, SHORT *selected_item)	
Scope	Application / Internal	
Input Parameters	control_id selected_item	The ID value of the specified control object . The item number of the selected item

Output Parameters

Result	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
	ERR_UI_OBJECT_NOT_MATCH	

Comment If no selection of item is required, then the selected_item can be set to NO_SELECTION.

28 ControlPopupSetTotalItems

Purpose This function is used to set up the total number of items in the popup list.

Prototype Err ControlPopupSetTotalItems (Object control_id,
USHORT total_num_items)

Scope Application

Input Parameters

control_id	The ID value of the specified popup list.
total_num_items	An unsigned integer that represents the total numbers of the items in the specified popuped trigger.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_OBJECT_NOT_MATCH	

Comment None

29 ControlPopupTriggerClosePopupTrigger

Purpose This function is called to close popup popup trigger

Prototype Err ControlPopupTriggerClosePopupTrigger()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return

TRUE	Success
FALSE	Not handled

Comment None

30 ControlRestoreBitBehind

Purpose This function is used to restore the bitmap of the image that is covered by the control object.

Prototype Err ControlRestoreBitBehind (ObjectID control_id)

Scope Application.

Input parameters

control_id	The ID value of the specified control object

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_NO_BITMAP_FOR_RESTORE	

Comments Calling this function when restore the pixels covered by other control object.

31 ControlSaveBehindBits

Purpose This function is used to store the image that is being covered by the specified menu. This function should be called before the specified object is drawn.

Prototype Err ControlSaveBehindBit (ObjectID control_id)

Scope Application.

Input parameters control_id The ID value of the specified control object

Output parameters NULL

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments Call this function only when an UI object is covered by another control object. Release the attribute and restore the cover area is required, once the area is not being covered.

32 ControlSavePopupBounds

Purpose This function is used to save the pixels behind the bounds of the specified popuped trigger.

Prototype Err ControlSavePopupBounds (ObjectID control_id)

Scope Internal

Input parameters control_id The ID value of the specified control object

Output parameters Null

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments Call this function only when an UI object is covered by another control object.

33 ControlSearchSelectedItem

Purpose This function is used to search the selected item number from the popup list.

Prototype **SHORT ControlSearchSelectedItem (Control *control_ptr,**
SHORT input_xcoord,
SHORT input_ycoord)

Scope Internal

Input Parameters

control_ptr	Pointer to the control structure.
input_xcoord	X-coordinate of the pen.
niput_ycoord	Y-coordinate of the pen.

Output Parameters None

Result Return the item number of the selection.

Comment None

34 ControlSetAttributes

Purpose This function is called in order to set all 5 attributes of the control object. The attributes are control_enable, control_drawn, control_save_behind, control_visible and control_active.

Prototype

```
Err ControlSetAttributes (ObjectID control_id,
                          BOOLEAN att_enable,
                          BOOLEAN att_drawn,
                          BOOLEAN att_save_behind,
                          BOOLEAN att_active,
                          BOOLEAN att_visible )
```

Scope Application

Input parameters

control_id	The ID value of the specified control object
att_enable	Boolean value to set up the state of the control_enable attribute.
att_drawn	Boolean value to set up the state of the control_drawn attribute.
att_save_behind	Boolean value to set up the state of the control_save_behind attribute.
att_active	Boolean value to set up the state of the control_active attribute.
att_visible	Boolean value to set up the state of the control_visible attribute.

Output parameters Null

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object

		that is stored in the RAM cannot be found.

Comments When a check box is not selected, the control_value is equal to 0. Otherwise the return value should be equal to 1. Invalid value will return from other control object.

35 ControlSetLabel

Purpose Set the text of the specified control object.

Prototype Err ControlSetLabel (ObjectID control_id, BYTE *control_label)

Scope Application

Input parameters

control_id	The ID value of the specified control object.
control_label	Pointer to the text of the specified object.

Output parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments None

36 ControlSetPopupScroll

Purpose To set the top item number of a specific popup trigger.

Prototype **void ControlSetPopupScroll (Control *addr,**
BYTE up_down)

Scope Internal

Input Parameters

addr	Pointer to the control structure.
up_down	Value to scroll the popup list.

Output Parameters None

Result None

Comment Calculate the new top item number of the popup trigger

37 ControlSetPopupTriggerUpDownArrow

Purpose This function is used to draw the Up/Down arrow on the popup list.

Prototype **void ControlSetPopupTriggerUpDownArrow (Control *addr,**
BYTE up_down)

Scope Internal

Input Parameters

addr	Pointer to the control structure.
up_down	Value to scroll the popup list.

Output Parameters None

Result None

Comment Once the number of items in the popup list are more then the screen can be displayed. An arrow will appear in the popuped trigger.

38 ControlSetPopupTriggerUpDownArrowOnly

Purpose This function is used to draw the Up/Down arrow on the popup list.

Prototype **void ControlSetPopupTriggerUpDownArrowOnly (Control *addr)**

Scope Internal

Input Parameters

addr	Pointer to the control structure.

Output Parameters None

Result None

Comment None

39 ControlUpdatePopupTrigger

Purpose To set the top item number of a specific popup trigger. The item in the list will then be rearranged, call ControlDrawControl to refresh the screen.

Prototype **Err ControlUpdatePopupTrigger (ObjectID control_id,**
USHORT new_top_num
)

Scope Internal

Input Parameters

control_id	The ID value of the specified control object.
new_top_num	New top item number of the popup list.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

Field Object Functions

1 FieldAddKeyInChar

Purpose This function is called in order to add a character to the field object at the position of the insertion point. After a character is added, the insertion point will be moved accordingly too.

Prototype Err FieldAddKeyInChar(ObjectID field_id, BYTE key)

Scope Application/Internal

Input Parameters field_id — The ID value of the specified field object
key — ASCII code of the key that will be added

Output Parameters None

Result TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

2 FieldCharPosToLineNum

Purpose This function is called in order to find the line number of the line that contains the character specified by the character position

Prototype Err FieldCharPosToLineNum (Field *addr, WORD char_pos, WORD *line_num)

Scope Application/Internal

Input Parameters addr — Pointer to data structure of field object
char_pos — The character position of the character in the string

Output Parameters line_num — Pointer to the line number of the line that contains the character specified by the character position

Result TRUE — No Error
ERR_UI_NOT_VALID_CHAR_POS — The required character position is out of range

Comment None

3 FieldCopy

Purpose Copy the current highlighted selection in an active field object to the text clipboard

Prototype Err FieldCopy (ObjectID field_id)

Scope Application/Internal

Input Parameters fieldID The ID value of the field object

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The field object must be active currently and highlight is on

4 FieldCut

Purpose Copy the current selection to the text clipboard, delete the selection from the field

Prototype Err FieldCut (ObjectID field_id)

Scope Internal

Input Parameters fieldID The ID value of the field object

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The function only works when the field object is the active one and the attribute field_highlight is must be set. The content in the clipboard is replace by the current highlighted text. The function does not redraw the field object.

5 FieldCutBackspaceHighlightText

Purpose To delete highlighted text in a field object when backspace is keyed-in.

Prototype void FieldCutBackspaceHighlightText(Field *addr)

Scope Internal

Input Parameters addr Pointer to field object

Output Parameters None

Return None

Comment Undo buffer is used to save the cut highlighted text

6 FieldCutSelectedText

Purpose This function is a routine function to cut the highlighted text from a field object. This function is called by *FieldCut* and *FieldInsertText.*

Prototype void FieldCutSelectedText (Field *addr)

Scope Internal

Input Parameters addr Pointer to field object

Output Parameters None

Return None

Comment This function is the main core function of CUT. This function cannot be used alone. This function should be called by other function.

7 FieldDelete

Purpose This function is a routine function and is used to delete a range of characters from a field object

Prototype Err FieldDelete (ObjectID field_id, WORD start_char,
WORD cut_length)

Scope Internal

Input Parameters

field_id	The ID value of the field object.
start_char	Starting character position of the character.
cut_length	The number of characters to be deleted

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After deletion, the field object should be redrawn again.

8 FieldDeleteField

Purpose The memory of the corresponding field is released. It is used when a field object is required to be deleted

Prototype Err FieldDeleteField(ObjectID field_id)

Scope Application

Input Parameters

field_id	The ID value of the field object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment Only memory will be released/deleted. Display will stay the same as before. The display should be erased first by calling *FieldEraseField* before the memory is erased by calling *FieldDeleteField*.

9 FieldDeleteString

Purpose This function is called to delete a section of text in the field object

Prototype Err FieldDeleteString(ObjectID field_id, WORD start_char, WORD cut_length)

Scope Application

Input Parameters

field_id	The ID value of the field object
start_char	The starting character position of section of text
cut_length	The number of characters that required to be delete

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The field object will not be redrawn

10 FieldDirty

Purpose It is used to check whether user has modified the field object or not. It means to return the attribute field_dirty.

Prototype Err FieldDirty(ObjectID field_id, BOOLEAN *dirty)

Scope Application/Internal

Input Parameters field_id The ID value of the field object.

Output Parameters dirty Pointer to variable to show field_dirty attribute

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment None

11 FieldDrawField

Purpose Draw the text of the field and set the field_drawn attribute. It will draw the frame, the text, the insertion point and highlight section of a field object.

Prototype Err FieldDrawField (ObjectID field_id)

Scope Application/Internal

Input Parameters field_id The ID value of the field object.

Output Parameters None

Return TRUE No Error

ERR_UI_RES_NOT_FOUND
The required object not found

Comment If field_highlight is set, then selected text is highlighted.
If field_insert_pt_visible is set, then insertion point is on

12 FieldDrawField2

Purpose Draw the text of the field and set the field_drawn attribute. It will draw the frame, the text, the insertion point and highlight section of a field object.

Prototype Err FieldDrawField2 (ObjectID field_id)

Scope Internal

Input Parameters field_id The ID value of the field object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment This function is different from FieldDrawField since FieldDrawField2 does not erase the screen before drawing

13 FieldEraseField

Purpose This function is used to erase the field object from the display.

Prototype Err FieldEraseField(ObjectID field_id)

Scope Application/Internal

Input Parameters field_id The ID value of the field object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment This function doesn't modify the contents of the field. If the field_insert_pt_visible is set, the inserting point is turned off. Clear the object from screen and the data of the object is still in memory. The attribute field_drawn is cleared

14 FieldGetAttributes

Purpose This function is used to get the attributes of a specified field object

Prototype Err FieldGetAttribute(ObjectID field_id, FieldAttr *field_attr)

Scope Application/Internal

Input Parameters field_id — The ID value of the field object.

Output Parameters field_attr — Pointer to attribute structure of a field object

Return TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

15 FieldGetCurrentHighlightedSelection

Purpose This function is used to get the parameters of the current highlighted text of a specified field object

Prototype Err FieldGetCurrentHighlightedSelection(ObjectID field_id,
WORD *star_char_pos,
WORD *end_char_pos)

Scope Application/Internal

Input Parameters field_id — The ID value of the field object.

Output Parameters	start_char_pos	Pointer to variable of character position of starting character of a highlight section
	end_char_pos	Pointer to variable of character position of end character of a highlight section

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment If there is no highlighted text (field_highlight is cleared), then both *start_char_pos and *end_char_pos are –1.

16 FieldGetDisplayRowNum

Purpose With a given input y-coordinate, the display row number of a field object can be worked out by this function.

Prototype

```
Err FieldGetDisplayRowNum (ObjectID field_id,
                           SHORT input_y,
                           WORD *row_num)
```

Scope Application/Internal

Input Parameters

	field_id	The ID value of the field object.
	input_y	Input y-coordinate
Output Parameters	row_num	Pointer to variable to show the row number of the line on the display

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment row_num is the row number of the lines on the display. The row_num of current top line is 0. If the input y-coordinate is not within the bounds of a field object, then –1 is returned instead.

17 FieldGetFieldBounds

Purpose Return the current bounds of a field.

Prototype Err FieldGetFieldBounds(ObjectID field_id,
ObjectBounds *bounds)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

bounds	Pointer to the bounds structure.

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

18 FieldGetFirstVisibleChar

Purpose This function is used to get the character position of the character that is on the top left corner of the display of the field object.

Prototype Err FieldGetFirstVisibleChar (ObjectID field_id,
WORD *char_pos)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

char_pos	Pointer to variable that stores the character position of the character that is at the left top corner of the display of field object

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

19 FieldGetFont

Purpose This function is called to get the font type of the field object.

Prototype Err FieldGetFont(ObjectID field_id, BYTE *font_id)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

font_id	Pointer to variable that stores the font type of the field object

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The output of the function is field_font_id in the data structure of a field object

20 FieldGetInsertPointPosition

Purpose This function is called in order to get the character position of the insertion point.

Prototype Err FieldGetInsertPointPosition(ObjectID field_id, WORD *char_pos)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

char_pos	Pointer to variable that stores the character position of the insertion point

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The output of this function is field_insert_pt_char_pos of a field object. This function only works when there is a insertion point flashing on the display. Otherwise the char_pos will be –1.

21 FieldGetLastVisibleChar

Purpose This function is used to get the character position of the character that is on the bottom right corner of the display of the field object.

Prototype Err FieldGetLastVisibleChar (ObjectID field_id, WORD *char_pos)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

char_pos	Pointer to variable that stores the character position of the character that is at the bottom right corner of the display of the field object

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

22 FieldGetMaxNumChars

Purpose This function is called in order to give the maximum number of characters that the field object can store.

Prototype Err FieldGetMaxNumChars(ObjectID field_id,
WORD *max_num_of_chars)

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters

max_num_of_chars	Pointer to the variable that stores the maximum number of characters that is allowed to hold by the field object

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

23 FieldGetMaxNumLinesDisplay

Purpose This function is called in order to give the maximum number of lines that can be displayed on the screen.

Prototype Err FieldGetMaxNumLinesDisplay(ObjectID field_id,
WORD *max_num_lines)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	max_num_lines	Pointer to the variable that stores the maximum number of lines that can be displayed on the screen.
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

24 FieldGetNumBlankLines

Purpose This function is called in order to get the number of blank lines that is on the display of the field object right now.

Prototype Err FieldGetNumBlankLines (ObjectID field_id,
UWORD *num_blank_line)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	num_blank_line	Pointer to the number of blank lines
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

25 FieldGetNumOfChars

Purpose This function gives the current number of characters in a field object

Prototype Err FieldGetNumOfChars(ObjectID field_id, WORD *num_chars)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	num_chars	Pointer to the current number of characters in a field object
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

26 FieldGetNumOfLinesDisplayed

Purpose This function is called in order to output the number of lines that are displaying on the display

Prototype Err FieldGetNumOfLinesDisplayed(ObjectID field_id, WORD *num_lines_displayed)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	num_lines_displayed	Pointer to variable that stores the number of lines of text on the display for a field object
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

27 FieldGetScrollbarAttribute

Purpose This function is called to get the field_has_scrollbar attribute.

Prototype Err FieldGetScrollbarAttribute(ObjectID field_id,
BOOLEAN *has_scrollbar);

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	has_scrollbar	Pointer to a boolean variable to show whether the field object has scrollbar or not

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment If the attribute is set, it means that the scrollbar for the field object is drawn on the display.

28 FieldGetTextPointer

Purpose This function is called in order to get the text pointer for the stored string in field object.

Prototype Err FieldGetTextPointer(ObjectID field_id, BYTE **text_ptr)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	text_ptr	Pointer reference to the string of a specified field object

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

29 FieldGetTopLineNum

Purpose This function is called to get the line number of the top displaying line of text

Prototype Err FieldGetTopLineNum(ObjectID field_id,
WORD *top_line_num)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	top_line_num	Pointer to variable that stores the line number of the top displaying line of text
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

30 FieldGetTotalNumOfLines

Purpose This function is called in order to give the total number of lines of a specified field object

Prototype Err FieldGetTotalNumOfLines(ObjectID field_id,
WORD *total_num_lines)

Scope Application/Internal

Input Parameters	field_id	The ID value of the field object.
Output Parameters	total_num_lines	Pointer to variable that stores the total number of lines in a specified field object
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

31 FieldInitField

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *FieldInitField* is usually followed by the *FieldDrawField* in order to draw the field object on the display. Object must be initialized once before using it without problem.

Prototype Err FieldInitField(ObjectID field_id)

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment None

32 FieldInsert

Purpose This function is called to paste text string from clipboard to a specified field object. The position for pasting is the position of the insertion point.

Prototype Err FieldInsert (ObjectID field_id, WORD *paste_size)

Scope Internal

Input Parameters

field_id	The ID value of the field object.

Output Parameters	paste_size	Pointer to the size of the text in the clipboard
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment The insertion point will be moved accordingly.

33 FieldInsertFunction

Purpose This function is the core function for pasting a string into the content of the field object. This function can only be called by FieldInsertText function.

Prototype Void FieldInsertFunction(Field *addr,
BYTE *paste_text,
WORD paste_size)

Scope Internal

Input Parameters	addr	Pointer to field object
	paste_text	Pointer to the text that will be pasted
	paste_size	The size of the text that will be pasted
Output Parameters	None	

Return None

Comment The insertion point will not be adjusted accordingly.

34 FieldInsertPtMove

Purpose This function is called in order to calculate the position of the insertion point one step to up or down, or left or right.

Prototype Err FieldInsertPtMove (ObjectID field_id)

Scope Internal

Input Parameters field_id The ID value of the specified field

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The direction of movement is determined by the parameter field_insert_pt_movement. The value can be NO_MOVEMENT, MOVE_UP, MOVE_DOWN, MOVE_LEFT or MOVE_RIGHT. This function only calculates out the correct position of insertion point after movement. The insertion point must be reset in order to display it in the correct position.

35 FieldInsertString

Purpose This function is called to insert a string into a field object where insert point is set and there is no highlight

Prototype Err FieldInsertString(ObjectID field_id, BYTE* paste_string)

Scope Internal

Input Parameters field_id The ID value of the specified field
paste_string The paste string

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment Don’t redraw

36 FieldInsertText

Purpose This function is called to insert a text to the specified field object. It is different from *FieldInsert* because it is not inserting a string from the clipboard.

Prototype Err FieldInsertText (ObjectID field_id,
BYTE *insert_string,

WORD field_len)

Scope Internal

Input Parameters

field_id	The ID value of the specified field
insert_string	Pointer to a string
field_len	the length of the string

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After the insertion of the string, the position of the insertion point will not be changed.

37 FieldPasteString

Purpose This function is called to paste string into the field object

Prototype Err FieldPasteString(ObjectID field_id, BYTE *paste_string)

Scope Application

Input Parameters

field_id	The ID value of the specified field
paste_string	the paste-in string

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

38 FieldRestoreBackspaceChar

Purpose This function is called to get the character that is deleted by Backspacing to the original field object

Prototype Err FieldRestoreBackspaceChar(ObjectID field_id,
UBYTE *character)

Scope Internal

Input Parameters

	field_id	The ID value of the field object that is being backspaced
Output Parameters	character	The pointer to ASCII of the deleted character

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_OBJECT_NOT_MATCH	The object is not a field object

Comment None

39 FieldScrollField

Purpose This function is called to scroll the content, which is being displayed on the screen, of the specified field object.

Prototype Err FieldScrollField(ObjectID field_id, WORD line_to_scroll)

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field
line_to_scroll	the number of lines to be scrolled

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The line_to_scroll can be positive or negative. Positive number means the scrolling direction is downward. Positive number means the scrolling direction is upward.

40 FieldSendChangeNotification

Purpose This function is called in order to send the EVT_FIELD_CHANGED to event queue.

Prototype void FieldSendChangeNotification()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment If the string of specified field object is changed, EVT_FIELD_CHANGED should be sent to notify the application.

41 FieldSendHeightChangeNotification

Purpose This function is used to send EVT_FIELD_HEIGHT to event queue

Prototype Err FieldSendHeightChangeNotification(ObjectID field_id)

Scope Application/Internal

Input Parameters

field_id	The ID value of field object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

42 FieldSetAttribute

Purpose This function is called in order to set the attributes of a specified field object

Prototype Err FieldSetAttribute(ObjectID field_id, FieldAttr input_field_attr)

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field
input_field_attr	The FieldAttr structure

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment Application should get the attributes by using *FieldGetAttribute*, then call *FieldSetAttribute* to set the attributes.

43 FieldSetBounds

Purpose This function is called to set the bounds of a specified field object

Prototype Err FieldSetBounds(ObjectID field_id, ObjectBounds bounds)

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field
bounds	Rectangular structure of the bounds of field object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

44 FieldSetDirty

Purpose This function is for setting the dirty attribute of the specified field object

Prototype Err FieldSetDirty(ObjectID field_id, BOOLEAN field_dirty)

Scope Application/Internal

Input Parameters

field_id	The ID value of field object
field_dirty	The variable that stores the dirty attribute

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

45 FieldSetFont

Purpose This function is called to set the type of font for a specified field object.

Prototype Err FieldSetFont(ObjectID field_id, BYTE font)

Scope Application/Internal

Input Parameters

field_id	The ID value of field object
font	The new type of font to be set

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After the new font type is set, the field object must be drawn again in order to display the string on the screen with new font type.

46 FieldSetHighlightSelection

Purpose This function is used to set the parameters of section of text that will be highlighted

Prototype

```
Err FieldSetHighlightSelection(ObjectID field_id,
                               WORD star_char_pos,
                               WORD end_char_pos)
```

Scope Application/Internal

Input Parameters

field_id	The ID value of the field object.
start_char_pos	Variable of character position of starting character of a highlight section
end_char_pos	Variable of character position of end character of a highlight section

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None.

47 FieldSetInsertPointOff

Purpose This function is called in order to turn off the insertion point of the specified field

Prototype Err FieldSetInsertPointOff(ObjectID field_id)

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The attribute field_insert_pt_visible is cleared

48 FieldSetInsertPointOn

Purpose This function is used to turn on the insertion point of the specified field object

Prototype Err FieldSetInsertPointOn(ObjectID field_id)

Scope Application/Internal

Input Parameters field_id The ID value of the field object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The attribute field_insert_pt_visible is set to TRUE

49 FieldSetInsertPointPositionByCharPos

Purpose This function is called for setting the insertion point position by inputting the new character position of the insertion point.

Prototype Err FieldSetInsertPointPositionByCharPos(ObjectID field_id,
WORD char_pos)

Scope Application

Input Parameters field_id The ID value of field object
char_pos The new character position of the insertion point

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment It will update the field_insert_pt_x and field_insert_py_y parameters correspondingly.

50 FieldSetInsertPointPositionByXY

Purpose This function is called for setting the insertion point position by inputting the x and y coordinates of the insertion point.

Prototype Err FieldSetInsertPointPositionByXY(ObjectID field_id,
SHORT insert_xcoord,
SHORT insert_ycoord)

Scope Application

Input Parameters

field_id	The ID value of field object
insert_xcoord	The new x coordinate of insertion point
insert_ycoord	The new y coordinate of insertion point

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment It will update the field_insert_pt_char_pos parameter correspondingly.

51 FieldSetMaxNumChars

Purpose This function is called to set the maximum number of characters in the field object.

Prototype Err FieldSetMaxNumChars(ObjectID field_id,
WORD max_num_of_chars)

Scope Application/Internal

Input Parameters

fieldID	The ID value of the field object.
max_num_of_charas	The new maximum number of characters in the field.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

52 FieldSetScrollbarAttribute

Purpose This function is called to set the attribute field_has_scrollbar of the specified field object.

Prototype Err FieldSetScrollbarAttribute(ObjectID field_id,
BOOLEAN has_scrollbar)

Scope Application/Internal

Input Parameters

fieldID	The ID value of the field object.
has_scrollbar	The new boolean variable of field_has_scrollbar attribute

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

53 FieldSetText

Purpose This function is called to set the string of the field and place the insertion point after the last visible character.

Prototype Err FieldSetText(ObjectID field_id, BYTE *string)

Scope Application/Internal

Input Parameters

fieldID	The ID value of the field object.
field_text	Pointer to the text of the field.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_FIELD_OVER_MAX_NUM_CHARS	The length of new string is longer than the maximum allowable number of characters

Comment None

54 FieldSetTopLineNum

Purpose This function is called to set which line to be the top line of the displaying field object.

Prototype Err FieldSetTopLineNum(ObjectID field_id,
WORD top_line_num)

Scope Application/Internal

Input Parameters

fieldID	The ID value of the field object.
top_line_num	The new top line number

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The new top_line_num must be a value between 0 and the total number of lines in the specified field object.

55 FieldStoreBackspaceChar

Purpose This function is called to store the character that is deleted by Backspacing

Prototype Err FieldStoreBackspaceChar(ObjectID field_id, UBYTE character)

Scope Internal

Input Parameters

field_id	The ID value of the field object that is being backspaced
character	The ASCII of the deleted character

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_OBJECT_NOT_MATCH	The object is not a field object

Comment None

56 FieldUndo

Purpose This function is called to undo the previous CUT or PASTE action.

Prototype Err FieldUndo (ObjectID field_id)

Scope Internal

Input Parameters

fieldID	The ID value of the field object.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment In order to undo CUT or PASTE, there should be no other action between the CUT or PASTE and the undo action. The field object must be active and the insertion point must be on.

Form Object Functions

1 FormAddOneObject

Purpose This function is called to add one object into a form object when the application is running

Prototype Err FormAddOneObject(ObjectID form_id, ObjectID object_id, BYTE object_type)

Scope Application

Input Parameters

form_id	The ID value of a form object
object_id	The ID value of a UI object
object_type	The object type of the add-in object

Output Parameters None

Result

TRUE	Handled
ERR_UI_RES_NOT_FOUND	The form object cannot be found

Comment None

2 FormCheckObjectExists

Purpose This function is called in order to find out whether a particular object is already initialized into the memory from resource file.

Prototype BOOLEAN FormCheckObjectExists(ObjectID object_id)

Scope Application/Internal

Input Parameters

object_id	The ID value of a UI object

Output Parameters None

Result

TRUE	The object is loaded
FALSE	The object is not loaded

Comment None

3 FormCheckStyle

Purpose There are a number of styles of a form. Each style has a different looking. This function is used to get to know the style of a specific form.

Prototype Err FormCheckStyle(ObjectID form_id, BYTE *form_style)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters

form_style	Pointer to variable form_style The styles are BACKGROUND, NORMAL_FORM, BITMAP_DIALOG and NON_BITMAP_DIALOG.

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

4 FormDeleteAllFormObjects

Purpose This function is called in order to delete all the objects in a form. The form object will be deleted too

Prototype Err FormDeleteAllFormObjects(ObjectID form_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

5 FormDeleteForm

Purpose The memories of the corresponding form object are released. It is used when a form is required to delete. The entries in both Link List Table and Lookup Table are deleted too

Prototype Err FormDeleteForm(ObjectID form_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment Only memory will be released/deleted. Display will stay the same as before. The display of the form should be erased before the memory is erased. In addition, the objects in the form should all be erased and deleted before the form is erased and deleted.

6 FormDispatchEvent

Purpose Dispatch an event to the application's handler for the form. This would be the last function in the EventLoop procedure of the application

Prototype Boolean FormDispatchEvent (EvtType *event)

Scope Application

Input Parameters event Pointer to an event

Output Parameters None

Return Boolean value to show whether the FormDispatchEvent handled the event or not.

Comment The event will be dispatched to the current active form.

7 FormDrawDialog

Purpose This function is called to draw the form object on the display according to the form style. This function only draw the form if the style is either BITMAP_DIALOG or NON_BITMAP_DIALOG.

Prototype void FormDrawDialog(Form *addr)

Scope Internal

Input Parameters addr Pointer to a form object

Output Parameters None

Return None

Comment This function is called automatically within *FormDrawForm*.

8 FormDrawDialogBackground

Purpose This function is called to draw the background of a dialog form

Prototype void FormDrawDialogBackground(ObjectBounds *bounds,

BYTE bg_color)

Scope Internal

Input Parameters

bounds	Pointer to object bounds
bg_color	The required background color

Output Parameters None

Return None

Comment None

9 FormDrawForm

Purpose The form and all the objects of the form are drawn on the display by using this function. If the objects are not loaded to RAM, then corresponding initialization functions are called to load the data automatically.

Prototype Err FormDrawForm(ObjectID form_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment Since the form may cover the other form, therefore the screen image behind the form must be stored before the form is drawn by accessing *FormSaveBehindBits* function. The form_drawn attribute is set after the form is drawn on the display. In order to draw the form on the display, the attribute form_visible must be set before calling *FormDrawForm*

10 FormDrawNormalForm

Purpose This function is called to draw the form object on the display according to the form style. This function only draw the form if the style is NORMAL_FORM.

Prototype void FormDrawNormalForm(Form *addr)

Scope Internal

Input Parameters addr Pointer to form object

Output Parameters None

Return None

Comment This function can only be called within *FormDrawForm*

11 FormEraseForm

Purpose The form on the display is erased. If there is a screen that is covered up by this form, then *FormRestoreSaveBehindBits* fucntion requires to be called in order to restore the image that is covered.

Prototype Err FormEraseForm(ObjectID form_id)

Scope Application/Internal

Input Parameters form_id ID of the form object

Output Parameters None

Return TRUE No error
ERR_UI_RES_NOT_FOUND
the specified UI Object not found

Comment The form on the display will be erased, but the memory will stay the same. Some forms may cover up the screen that is behind it, therefore, after the erase of the current form, FormRestoreSaveBehindBits may need to be accessed in order to restore the covered screen image.
The attribute form_drawn is cleared.

12 FormGetActiveFormID

Purpose This function is used to get the ID of the current active form

Prototype Err FormGetActiveFormID(ObjectID *form_id)

Scope Application/Internal

Input Parameters None

Output Parameters

form_id	Pointer to the ID value of the form

Return

TRUE	No error
ERR_UI_OBJECT_ID_NOT_FOUND	the specified UI Object not found

Comment None

13 FormGetActiveObject

Purpose This function is used to find out the ID value of the UI object that has the focus

Prototype Err FormGetActiveObject(ObjectID form_id, ObjectID *active_id)

Scope Application

Input Parameters

form_id	The ID value of the form object
active_id	Pointer to ID value of the active object in the specified form

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

14 FormGetControlValue

Purpose This function is used to get the value (checked or not, pushed or not) on/off of a control object (push buttons or checkbox)

Prototype Err FormGetControlValue(ObjectID control_id,
BOOLEAN *ctl_value)

Scope Application

Input Parameters

control_id	the ID value of the control object
ctl_value	Pointer to the control value

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found
ERR_UI_OBJECT_NOT_MATCH	the type of the specified control object is not push button or checkbox

Comment This function can only be used for push button and check box control. It is because only those 2 objects have this value attribute.

15 FormGetMenuID

Purpose This function is called in order to find out the Id value of the menu object for a specified form

Prototype Err FormGetMenuID(ObjectID form_id,ObjectID *menu_id)

Scope Application/Internal

Input Parameters

form_id	the ID value of the form

Output Parameters

menu_id	Pointer to the ID value of the menu of the specified form

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	One of the objects in the form is not in the lookup table
ERR_UI_OBJECT_NOT_FOUND	No menu for the specified form

Comment None

16 FormGetNumberOfObjects

Purpose This function is used to find out the number of UI objects in a specified form

Prototype Err FormGetNumberOfObjects(ObjectID form_id,
USHORT *num_objects)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters

num_objects	Pointer to a number of objects parameter that shows the total number of objects in the specified form

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

17 FormGetObjectBounds

Purpose This function is used to get the bounds of the specified UI object in a specified form

Prototype Err FormGetObjectBounds(ObjectID object_id,
ObjectBounds *bounds)

Scope Application/Internal

Input Parameters

object_id	the ID of the UI object

Output Parameters

bounds	Pointer to the a ObjectBounds structure

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

18 FormGetObjectPointer

Purpose This function is used to find the pointer to the data structure of an object in a specified form

Prototype Err FormGetObjectPointer(ObjectID object_id, BYTE *object_type, void **address)

Scope Application/Internal

Input Parameters

object_id	the ID value of the required UI Object

Output Parameters

object_type	Pointer to the object type of the object ID
address	Pointer reference to data structure of the object

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

19 FormInitAllFormObjects

Purpose This function is called to initialise the particular form and related objects in the form. All objects will not be drawn on the display.

Prototype Err FormInitAllFormObjects(ObjectID form_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the specified form

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment After initialization, the form object is put into the Link List Table and Lookup Table for searching. All the objects are put in Lookup Table

too.

20 FormInitForm

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *FormInitForm* is usually followed by the *FormDrawForm* in order to draw the form on the display. Object must be initialized once before using it without problem.

Prototype Err FormInitForm(ObjectID form_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the specified form

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment After initialization, the form object is put into the Link List Table and Lookup Table for searching.

21 FormObjectReleaseFocus

Purpose This function is used to release the focus of the active UI object in the current active form. The appearance of the object may be changed

Prototype Err FormObjectReleaseFocus()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return TRUE No error
ERR_UI_RES_NOT_FOUND
the specified UI Object not found

Comment After the focus of the object is released, the appearance of object may be changed. For example, if the focus of a field object is released, then the insertion point will be turned off.

22 FormObjectRestoreFocus

Purpose This function is called when the focus of the current active object in the current active form is released and it is required to restore it back. Therefore, *FormObjectReleaseFocus* is usually followed by *FormObjectRestoreFocus*.

Prototype Err FormObjectRestoreFocus()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return TRUE No error
ERR_UI_RES_NOT_FOUND
the specified UI Object not found

Comment After the focus of the object is restored, the appearance of object may be changed. For example, if the focus of a field object is restored, then the insertion point will be turned on again.

23 FormObjectSetFocus

Purpose This function is called in order to set an UI object as the current active object of the current active form

Prototype Err FormObjectSetFocus(ObjectID object_id)

Scope Application/Internal

Input Parameters object_id — The ID value of the object that is required to set as current active object of the current active form

Output Parameters None

Return
TRUE — No error
ERR_UI_RES_NOT_FOUND — the specified UI Object not found

Comment None

24 FormPointInObject

Purpose To check whether the coordinates that passed to the function is within the boundary of the object or not.

Prototype
```
Err FormPointInObject(ObjectID object_id,
                      SHORT x_coord,
                      SHORT y_coord)
```

Scope Application/Internal

Input Parameters
object_id — The ID value of the specified object
x_coord — the window-relative x-coordinate
y_coord — the window-relative y-coordinate

Output Parameters None

Return
TRUE — Success
ERR_UI_INV_INPUT_COORD — The x-coordinate and y-coordinate is not within the bounds of the specified object
ERR_UI_RES_NOT_FOUND — The object not found

Comment None

25 FormPopupForm

Purpose This function is used to popup or open a specific form. It will send EVT_FORM_LOAD and EVT_FORM_OPEN.

Prototype Err FormPopupForm(ObjectID form_id)

Scope Application/Internal

Input Parameters form_id The ID value of the form

Output Parameters None

Return TRUE Success.

Comment The data of the specified form object must be already loaded into the RAM from resource file.

26 FormRestoreBitsBehind

Purpose This function is used to draw back the covered image that is already saved by calling *FormSaveBehindBits* onto the screen.

Prototype Err FormRestoreBitBehind(ObjectID form_id)

Scope Application / Internal

Input Parameters form_id The ID value of the specified form

Output Parameters None

Return None

Comment The attribute form_save_behind is cleared

27 FormSaveBehindBits

Purpose This function is used to set the form_save_behind attribute of the data structure of the form and to store the image that is being covered by the specified form. This function should be called before the specified form is drawn.

Prototype Err FormSaveBehindBits(ObjectID form_id)

Scope Application / Internal

Input Parameters form_id The ID value of the specified form object

Output Parameters None

Return TRUE Success
ERR_UI_RES_NOT_FOUND
The specified object not found

Comment The attribute form_save_behind is set

28 FormSetActiveForm

Purpose This function is used to set the active form. By switching the active form, the objects in that form can operate properly.

Prototype Err FormSetActiveForm(ObjectID form_id)

Scope Application/Internal

Input Parameters form_id The ID value of the specified form

Output Parameters None

Return TRUE Success
ERR_UI_RES_NOT_FOUND
The required object is not in lookup table
ERR_UI_OBJECT_NOT_FOUND
The object is not in the Link List Table

Comment None

29 FormSetControlGroupSelection

Purpose This function is used to check or push a checkbox or push button object. The other related objects with the same group id will be cleared.

Prototype Err FormSetControlGroupSelection(ObjectID form_id,
ObjectID object_id)

Scope Application/Internal

Input Parameters

form_id	The ID value of the specified form object
control_id	The ID value of the specified control object

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI object not found in lookup table

Comment By using this function to select a particular control (push button or checkbox) object, the other objects having the same group number are cleared. The display will not be updated correspondingly too

30 FormSetDialogTitle

Purpose This function is called to set the dialog title of all form except form with BACKGROUND style

Prototype Err FormSetDialogTitle (ObjectID form_id, BYTE *title_text)

Scope Application/Internal

Input Parameters

form_id	The ID value of the form
title_text	Pointer to the title of the dialog form

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI object not found in lookup table

Comment None

31 FormSetEventHandler

Purpose This function is used to set the event handler callback routine for the specified form. This function can be used to set the corresponding function used to handle the events for each particular form.

Prototype Err FormSetEventHandler(ObjectID form_id,
BOOLEAN function(EvtType *))

Scope Application/Internal

Input Parameters

formID	The ID value of the specified form object
src_func	Pointer to the FormDispatcEvent function pointer
func	Pointer to the callback function

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI object not found in lookup table

Comment This function should be called by *ApplicationHandleEvent.*

32 FormSetFormActiveObject

Purpose This function is called in order to set a particular object in the form as an active object.

Prototype Err FormSetFormActiveObject(ObjectID form_id, ObjectID object_id)

Scope Application / Internal

Input Parameters

form_id	The ID value of the specified form object
object_id	The ID value of an object in the form

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI form not found in lookup table
ERR_UI_OBJECT_NOT_MATCH	UI object is not in the form

Comment None

33 FormSetObjectBounds

Purpose This function is used to set the bounds of the specified UI object

Prototype Err FormSetObjectBounds(ObjectID object_id, ObjectBounds bounds)

Scope Application / Internal

Input Parameters

object_id	The ID value of the specified UI object
bounds	Data structure of the bounds

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI object not found in lookup table

Comment The object with the new bounds will not be re-drawn on the screen automatically. The application programmer should access the FormDrawForm or function to update the UI object

34 FormSetObjectPosition

Purpose This function is called to set the position of object

Prototype Err FormSetObjectPosition(ObjectID object_id,
SHORT x_coord, SHORT y_coord)

Scope Application / Internal

Input Parameters

object_id	The ID value of the specified UI object
x_coord	The x-coordinate of the top-left corner of the object
y_coord	The y-coordinate of the top-right corner of the object

Output Parameters None

Return

TRUE	Success
ERR_UI_RES_NOT_FOUND	UI object not found in lookup table

Comment None

Keyboard Object Functions

1 KeyboardCheckKeyboardStatus

Purpose This function is called in order to check whether the keyboard is on the screen or not

Prototype BOOLEAN KeyboardCheckKeyboardStatus()

Scope Application/Internal

Input Parameters None

Output Parameters None

Result

TRUE	if keyboard is on screen
FALSE	if keyboard is not on screen

Comment None

2 KeyboardDrawInvertKey

Purpose This function is called to invert the color of a key button on the software keyboard.

Prototype void KeyboardDrawInvertKey(UBYTE key)

Scope Internal

Input Parameters

key	ASCII code of the key on the displaying software keyboard

Output Parameters None

Result None

Comment None

3 KeyboardDrawKeyboard

Purpose This function is called to draw the software keyboard on screen for the first time and re-initialize the keyboard structure. It is because when the keyboard is called to popup, it always displays the same bitmap with non-pressed CAP, SHIFT and INT'L.

Prototype void KeyboardDrawKeyboard()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment All attributes of Keyboard are cleared.

4 KeyboardDrawKeyboardBitmap

Purpose This function is called to draw one of the four keyboard bitmap onto the display

Prototype void KeyboardDrawKeyboardBitmap(BYTE bitmap_index)

Scope Internal

Input Parameters

bitmap_index	the bitmap number of the keyboard bitmap, starting from 1 to 6

Output Parameters None

Result None

Comment None

5 KeyboardGetClickedKey

Purpose This function is called to find out the ASCII code of the key that the pen is on.

Prototype UBYTE KeyboardGetClickedKey(SHORT xcoord, SHORT ycoord)

Scope Internal

Input Parameters

xcoord	the x-coordinate of the pen
ycoord	the y-coordinate of the pen

Output Parameters None

Result If the a key is being clicked then the ASCII code of that key is returned. Otherwise, NULL will be returned.

Comment None

6 KeyboardInitKeyboard

Purpose This function is called to load the related data to the keyboard structure in order to initialize the keyboard structure.

Prototype void KeyboardInitKeyboard()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

7 KeyboardRemoveKeyboard

Purpose This function is called to remove the popup keyboard on the screen. It will erase the region and erase all related memory

Prototype void KeyboardRemoveKeyboard()

Scope Application/Internal

Input Parameters None

Output Parameters None

Result None

Comment If the keyboard is not popup, then there will be no effect

8 KeyboardRestoreBitBehind

Purpose This function is called to re-draw the covered image behind the keyboard onto the display.

Prototype void KeyboardRestoreBitBehind()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment The attribute keyboard_save_behind is cleared

9 KeyboardSaveBehindBits

Purpose This function is called to save the covered image behind the keyboard.

Prototype void KeyboardSaveBehindBits()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment The attribute keyboard_save_behind is set

10 KeyboardSendEvent

Purpose This function is used to send an EVT_KEY event to field object or textbox in order to display it on the display.

Prototype void KeyboardSaveBehindBits(UBYTE key, BOOLEAN cap,

BOOLEAN shift,
BOOLEAN international)

Scope Internal

Input Parameters

key	the ASCII code of the key to be displayed
cap	Boolean valuable to show whether CAPLOCK is pressed or not
shift	Boolean valuable to show whether SHIFT is pressed or not
international	Boolean valuable to show whether INTERNATIONAL is pressed or not

Output Parameters None

Result None

Comment None

11 KeyboardSetKeyboardInvisible

Purpose This function is called to set keyboard invisible, so that it would not be popup.

Prototype void KeyboardSetKeyboardInvisible()

Scope Application/Internal

Input Parameters None

Output Parameters None

Result None

Comment If the keyboard is already popup, it would not be erased

12 KeyboardSetKeyboardStatus

Purpose This function is called to set the flag keyboard_popup to show whether the keyboard is popup or not

Prototype void KeyboardSetKeyboardStatus(BOOLEAN status)

Scope Application/Internal

Input Parameters

status	TRUE if keyboard is popup FALSE if keyboard is not popup

Output Parameters None

Result None

Comment If the keyboard is already popup, it would not be erased

13 KeyboardSetKeyboardVisible

Purpose This function is called to set the keyboard to visible, so that the keyboard can be popup

Prototype void KeyboardSetKeyboardVisible()

Scope Application/Internal

Input Parameters None

Output Parameters None

Result None

Comment None

Line Object Functions

1 LineDeleteLine

Purpose This function is used to delete the specified line object from memory in order to free and release the occupied memory.

Prototype Err LineDeleteLine (ObjectID line_id)

Scope Application

Input Parameters

line_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function won't change/update the display. LineEraseLine Should be called before calling LineDeleteLine.

2 LineDrawLine

Purpose This function is used to draw the specified line object in a specified form. Before calling this function, the LineInitLine function should be called first in order to initialise the line object from the resource file.

Prototype Err LineDrawLine (ObjectID line_id)

Scope Application

Input Parameters

line_id	The ID value of the specified line.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function doesn't erase the region behind the line before drawing.

3 LineEraseLine

Purpose This function is used to erase a line object in a form.

Prototype Err LineEraseLine (ObjectID line_id)

Scope Application

Input Parameters

line_id	The ID value of the specified line object.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment The line_drawn attribute is cleared.

4 LineGetAttribute

Purpose To check whether the specified line object is drawn on the display or not.

Prototype

```
Err LineGetAttribute (ObjectID line_id,
                      BOOLEAN *line_drawn,
                      BOOLEAN *line_visible)
```

Scope Application

Input Parameters

line_id	The ID value of the specified line object.

Output Parameters

line_drawn	Pointer to the Boolean value that show the specified line is drawn on the display or not.
line_visible	Pointer to the Boolean value that show the specified line is active or not.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

5 LineGetLineCharacteristics

Purpose To check the characteristics of the specified line object.

Prototype

```
Err LineGetLineCharacteristics (ObjectID line_id,
                                BYTE *line_color,
```

BOOLEAN *line_style,
BOOLEAN *line_thick)

Scope Application

Input Parameters

line_id	The ID value of the specified line object.

Output Parameters

line_color	Get the color of the specified line. 0 = COLOR_WHITE, 1 = COLOR_GREY1, 2 = COLOR_GREY2, 3 = COLOR_BLACK.
line_style	Get the style of the specified line. 0 = Non_Dotted_Line, 1 = Dotted_Line.
line_thick	Get the thickness of the specified line. 1 = Single Line, 2 = Double Line …etc

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

6 LineGetPosition

Purpose To get the start point and the end point of the specified line object.

Prototype Err LineGetPosition (ObjectID line_id,
SHORT *start_x,
SHORT *start_y,
SHORT *end_x
SHORT *end_y)

Scope Application

Input Parameters

line_id	The ID value of the specified line object.

Output Parameters

start_x	Get the starting x-coordinate.
start_y	Get the starting y-coordinate.
end_x	Get the ending x-coordinate.
end_y	Get the ending y-coordinate.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

7 LineInitLine

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. LineInitLine function is usually followed by the LineDrawLine in order to draw the line object on the display.

Prototype Err LineInitLine (ObjectID line_id)

Scope Application

Input Parameters

line_id	The ID value of the specified line object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_CANT_CREATE_LOOKUP_TABLE	
ERR_UI_TYPE_MISMATCH	
ERR_UI_CANT_CREATE_POINTER	
ERR_RES_OBJ_MISS	

Comment This function will not draw the line object on the display

8 LineSetAttribute

Purpose Set to indicate the specified line object is drawn on the display.

Prototype Err LineGetAttribute (ObjectID line_id,
BOOLEAN att_drawn,
BOOLEAN att_visible)

Scope Application

Input Parameters

line_id	The ID value of the specified line object.
att_drawn	The value to set up the drawn attribute of the specified line object.
att_visible	The value to set up the visible attribute of the specified line object.

Output Parameters None

Result

TRUE	Success.

ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

9 LineSetLineCharacteristics

Purpose To check whether the specified line object is drawn on the display or not.

Prototype
```
Err LineGetLineCharacteristics (ObjectID line_id,
                                BYTE line_color,
                                BOOLEAN line_style,
                                BOOLEAN line_thick)
```

Scope Application

Input Parameters

line_id	The ID value of the specified line object.
line_color	The value to set up the color of the specified line. White = COLOR_WHITE, 1 = COLOR_GREY1, 2 = COLOR_GREY2, Black = COLOR_BLACK.
line_style	The value to set up the style of the specified line. Non dotted line = Non_Dotted_Line, Dotted line = Dotted_Line.
line_thick	The value to set up the thickness of the specified line. Single line = Single, Double lines = Double…etc

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

10 LineSetPosition

Purpose To set the starting point and the ending point of the specified line object.

Prototype
```
Err LineGetPosition (ObjectID line_id,
                     SHORT start_x,
                     SHORT start_y,
                     SHORT end_x
                     SHORT end_y)
```

Scope Application

Input Parameters

line_id	The ID value of the specified line object.
start_x	The value to set up the starting x-coordinate of the specified line object.
start_y	The value to set up the starting y-coordinate of the specified line object.
end_x	The value to set up the ending x-coordinate of the specified line object.
end_y	The value to set up the ending y-coordinate of the specified line object.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

List Object Functions

1 ListClickedRegion

Purpose This function returns which region is currently entered by the pen

Prototype
```
USHORT ListClickedRegion(List *addr,
                         SHORT x_input,
                         SHORT y_input)
```

Scope Internal

Input Parameters

addr	Pointer to list object
x_input	The input x-coordinate of the pen
y_input	The input y-coordinate of the pen

Output Parameters None

Return

REGION_UP_ARROW	if the pen is within the region of up arrow
REGION_DN_ARROW	if the pen is within the region of down arrow
REGION_ITEMS	if the pen is not within the regions of up arrow and down arrow

Comment It is assumed that the pen is within the bounds of the list object.

2 ListDeleteAllItems

Purpose This function is called for deleting all the items in a list object.

Prototype Err ListDeleteAllItems(ObjectID list_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The item number is invalid

Comment All the items are deleted. The related parameters are also updated correspondingly.

3 ListDeleteItem

Purpose This function is called for deleting an item (selection) in the list object.

Prototype Err ListDeleteItem(ObjectID list_id, USHORT item_number)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
item_number	The item number of the item that is required to delete

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The item number is invalid

Comment After the deletion of an item, the total number of item will be updated. But the display will not be updated. Therefore, *ListDrawList* is required to call in order to update the list object accordingly. The list of items will be re-arranged after deletion.

4 ListDeleteList

Purpose The memory of the corresponding list object is released. It is used when a list object is required to be deleted.

Prototype Err ListDeleteList(ObjectID list_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment Only memory will be released/deleted. Display will stay the same as before. The display of the list object should be erased by calling *ListEraseList* before the object is deleted.

5 ListDrawBackgroundBox

Purpose This function is used to draw specified background box of the list.

Prototype
```
Void ListDrawBackgroundBox ( List *          addr,
                             ObjectBounds    bounds,
                             USHORT          list_border )
```

Scope Internal

Input Parameters

addr	The address of the List structure.
bounds	Bounds of the List object.
list_border	Border of the specified List.

Output Parameters None

Result None

Comment None

6 ListDrawList

Purpose This function is used to draw specified list object on the display. The list object should be already initialized by the function *ListInitList*

Prototype Err ListDrawList (ObjectID list_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The list_drawn attribute is set to show the list object drawn on the display. The style of list object that is drawn on the display is determined by list_style parameter.

7 ListDrawUpDownArrow

Purpose This is an internal function and this function is called by *ListDrawList* to draw the up and down arrow for scrolling within the list object.

Prototype void ListDrawUpDownArrow(List *addr,
ObjectBounds *list_bounds)

Scope Internal

Input Parameters

addr	Pointer to list object
list_bounds	Pointer to rectangular structure of the bounds of the specified list object

Output Parameters None

Result None

Comment This function will only be called if the attribute list_set_scroll is set. After the drawing of arrow, the attribute list_arrow_up or/and list_arrow_down will be set to TRUE if the corresponding arrow is drawn

8 ListEraseList

Purpose This function is used to erase the list object on the display. The function won't delete the memory of the list object.

Prototype Err ListEraseList(ObjectID list_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The list_drawn is cleared to show that the list object is not drawn on the display

9 ListGetAttribute

Purpose This function is called to get the attributes of the specified list object The attributes that can be retrieved are list_enable, list_drawn, list_active, list_visible and list_set_scroll.

Prototype
```
Err ListGetAttribute(ObjectID list_id,
                     BOOLEAN *att_enable,
                     BOOLEAN *att_drawn,
                     BOOLEAN *att_active,
                     BOOLEAN *att_visible,
                     BOOLEAN *att_set_scroll)
```

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters

att_enable	Pointer to variable to store list_enable attribute
att_drawn	Pointer to variable to store list_drawn attribute
att_active	Pointer to variable to store list_active attribute
att_visible	Pointer to variable to store list_visible attribute
att_set_scroll	Pointer to variable to store list_set_scroll attribute

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

10 ListGetHighlightedItem

Purpose This function is called get the item number of the highlighted item

Prototype Err ListGetHighlightedItem(ObjectID list_id, SHORT *item_num)

Scope Application/Internal

Input Parameters list_id The ID value of the list object

Output Parameters item_num Pointer to the value of the highlighted item number. The value –1 means that there is no highlighted item

Result TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment None

11 ListGetListBounds

Purpose This function is called to retrieve the bounds of the specified list object.

Prototype Err listGetListBounds(ObjectID list_id, ObjectBounds *bounds)

Scope Application/Internal

Input Parameters list_id The ID value of the list object

Output Parameters bounds Pointer to rectangular structure of bounds of the list object

Result TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment None

12 ListGetListItem

Purpose This function is for application to get an item in the list object.

Prototype Err ListGetListItem(ObjectID list_id,
USHORT item_number,
BYTE **item_text)

Scope Application/internal

Input Parameters list_id The ID value of the list object

	item_number	The item number of the item that is required
Output Parameters	item_text	Pointer reference to the string of the item
Result	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found
	ERR_UI_ITEM_NUMBER_EXCEED	The item number is not valid

Comment None

13 ListGetMaxNumItemsDisplay

Purpose This function is called to get the maximum number of items that can currently be displayed in the list object on the screen.

Prototype Err ListGetMaxNumItemsDisplay (ObjectID list_id,
USHORT *max_num_items)

Scope Application/Internal

Input Parameters	list_id	The ID value of the list object
Output Parameters	max_num_items	Pointer to variable of the maximum number of items that can be handled in the list object
Result	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

14 ListGetNumOfItemsDisplay

Purpose This function is called to get the number of items that can currently be displayed in the list object on the screen.

Prototype Err ListGetNumItemsDisplay (ObjectID list_id,
USHORT *num_items)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters

num_items	Pointer to variable of the number of items that are currently displaying in the list object

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

15 ListGetNoOfItems

Purpose This function is called to get the total number of items that are currently in a list object

Prototype Err ListGetNoOfItems(ObjectID list_id, USHORT *num_items)

Scope Application /Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters

num_items	Pointer to variable of total number of items

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

16 ListGetSelectedItem

Purpose This function is called in order to get the current selected item of a specified list object.

Prototype Err ListGetSelectedItem(ObjectID list_id,
SHORT *item_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters

item_num	Pointer to a variable that stores the item number of the selected item

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment If there is a selection, then the item number of the selected item is returned. Otherwise, a NO_SELECTION is returned instead.

17 ListGetTopItemNum

Purpose This function is called to get the item number of the top item of the list object on the display

Prototype Err ListGetTopItemNum(ObjectID list_id, USHORT *item_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters

item_num	Pointer to a variable that stores the item number of top item on the display

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

18 ListGetTotalItems

Purpose This function is called to get the total number of items in the list object

Prototype Err ListGetTotalItems(ObjectID list_id, USHORT *num_items)

Scope Application/Internal

Input Parameters list_id — The ID value of the list object

Output Parameters num_items — Pointer to a variable that stores the total number of items in the list object

Result
TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

19 ListHighlightOneItem

Purpose This function is called to highlight one item or un-highlight one item of the list object

Prototype
```
Err ListHighlightOneItem (List *addr,
                          SHORT item_num,
                          BOOLEAN item_onoff)
```

Scope Internal

Input parameters
addr — Pointer to list object
item_num — the item number of the item that wants to be highlight or un-highlight
item_onoff — TRUE if highlight; FALSE if un-highlight

Output parameters None

Return
TRUE — Success
FALSE — Not handled

Comments None

20 ListInitList

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *ListInitList* is usually followed by the *ListDrawList* in order to draw the list object on the display. Object must be initialized once before using it without problem.

Prototype
```
Err ListInitList(ObjectID list_id)
```

Scope Application/Internal

Input Parameters

field_id	The ID value of the specified field

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment None

21 ListInsertItem

Purpose This function is called to insert an item into the list of a specified list object.

Prototype

```
Err ListInsertItem(ObjectID list_id,
                   USHORT item_number,
                   BYTE *item_text)
```

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
item_number	The item number that is required to add the new item to
item_text	Pointer to string of the new item

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

22 ListRecalculateMaxNumItemsDisplay

Purpose This function is called to recalculate the maximum number of items that can be displayed in the list object

Prototype Err ListRecalculateMaxNumItemsDisplay(ObjectID list_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

23 ListSetAttribute

Purpose This function is called to set the attributes of the specified list object The attributes that can be set are list_enable, list_drawn, list_active, list_visible and list_set_scroll.

Prototype

```
Err ListGetAttribute(ObjectID list_id,
                     BOOLEAN att_enable,
                     BOOLEAN att_drawn,
                     BOOLEAN att_active,
                     BOOLEAN att_visible,
                     BOOLEAN att_set_scroll)
```

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
att_enable	new list_enable attribute
att_drawn	new list_drawn attribute
att_active	new list_active attribute
att_visible	new list_visible attribute
att_set_scroll	new list_set_scroll attribute

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

24 ListSetFont

Purpose This function is called to set the font of the list object

Prototype Err ListSetFont(ObjectID list_id, BYTE font_id)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
font_id	SMALL_FONT, MEDIUM_FONT, LARGE_FONT

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The maximum number of items that can be displayed is also updated

25 ListSetHighlightedItem

Purpose This function is called to set the item to be highlighted

Prototype Err ListSetHighlightedItem(ObjectID list_id, SHORT item_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
item_num	The item number of the highlighted item

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

26 ListSetListBounds

Purpose This function is called to set the bounds of the list object

Prototype Err ListSetListBounds(ObjectID list_id, ObjectBounds bounds)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
bounds	The new bounds of list object

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

27 ListSetNumOfItemsDisplay

Purpose This function is called to set the number of items for displaying on the screen.

Prototype Err ListSetNumItemsDisplay (ObjectID list_id,
USHORT num_items)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
num_items	The new number of items on the display

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment If the input number of items exceed the limit for displaying in the bounds of the list object, then the drawing function will adjust the

number of items on display in order to find out the maximum number of items for display.

28 ListSetScrollList

Purpose This function is called to scroll the list of items up or down. The list will be scrolled to next range of items.

Prototype Err ListSetScrollList(ObjectID list_id, BYTE up_down)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
up_down	the direction of scrolling. It can be either MOVE_UP or MOVE_DOWN. The list of items will be scrolled to next range of items

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

29 ListSetSelectedItem

Purpose This function is called to set the selected item of a list object.

Prototype Err ListSetSelectedItem(ObjectID list_id, SHORT item_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
item_number	item that is required to select

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The input item number is invalid

Comment item_num can be set to NO_SELECTION if no item is required to select.

30 ListSetTopItemNum

Purpose This function is called to set the top item number of the list object.

Prototype Err ListSetTopItemNum(ObjectID list_id, USHORT item_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
item_number	item number of the new top item

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The input item number is invalid

Comment This display will not be updated accordingly.

31 ListSetTotalItems

Purpose This function is called to set the total number of items in the list object.

Prototype Err ListSetTotalItems (ObjectID list_id, USHORT total_num_items)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
total_num_items	The new total number of items

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment Be careful to call this function, the parameter ‘total number of items’ should be set to the actual value of total number of items.

Otherwise, there will be unpredicted error.

32 ListUpdateList

Purpose This function is called in order to change the top item number and update the display at the same time.

Prototype Err ListUpdateList(ObjectID list_id, USHORT new_top_num)

Scope Application/Internal

Input Parameters

list_id	The ID value of the list object
new_top_num	The new top item number

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The input item number is invalid

Comment The display will be updated too

Menu Object Functions

1 MenuCloseMenu

Purpose This function is called to close the popup Menu

Prototype Err MenuCloseMenu()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return

TRUE	No Error
FALSE	No menu is popup
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_OBJECT_NOT_MATCH	The object type not match

Comment This function can only work when a menu is popup on the screen. Otherwise, there will be no effect

2 MenuDeleteAllItems

Purpose This function is called for deleting all the items in a menu object.

Prototype Err MenutDeleteAllItems(ObjectID menu_id)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the menu object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_ITEM_NUMBER_EXCEED	The item number is invalid

Comment All the items are deleted. The related parameters are also updated correspondingly.

3 MenuDeleteItem

Purpose This function is used to delete an item in the menu list and all the items will then be arranged.

Prototype Err MenuDeleteItem (ObjectID menu_id
USHORT item_number)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object .

	item_number	The item number that you want to delete.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_ITEM_NUMBER_EXCEED	Invalid item number.

Comment None

4 MenuDeleteMenu

Purpose This function is used to delete the specified menu object from memory in order to free and release the occupied memory.

Prototype Err MenuDeleteMenu (ObjectID menu_id)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function won't change/update the display. MenuEraseMenu Should be called before calling MenuDeleteMenu in order to avoid pen action on the object that is already deleted from memory

5 MenuDrawMenu

Purpose This function is used to draw the specified menu object in a specified form. Before calling this function, the *MenuInitMenu* function should be called first in order to initialise the menu object from the resource file.

Prototype Err MenuDrawMenu (ObjectID menu_id)

Scope Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function doesn't erase the region behind the menu before drawing. In order to save the image behind the popup menu, *MenuSaveBehindBits* should be called before *MenuDrawMenu*
After the drawing procedure, the attribute menu_drawn is set.

6 MenuGetAttrVisible

Purpose This function is called to get the the attribute menu_visible

Prototype

```
Err MenuGetAttrVisible ( ObjectID menu_id,
                         BOOLEAN *menu_visible )
```

Scope Application /Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters

menu_visible	Pointer to the return attribute

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This attribute shows whether the menu can be shown on the form. The attribute can be set by calling *MenuSetAttrVisible*

7 MenuGetMenuItem

Purpose To get the text of a specific popup item of the menu

Prototype Err MenuGetMenuItem (ObjectID menu_id,
USHORT item_number,
BYTE **item_text)

Scope Application /Internal

Input Parameters

menu_id	The ID value of the specified menu object .
item_number	The item number that you want to get the text.

Output Parameters

items_text	Pointer reference to the char array of the item text.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_ITEM_NUMBER_EXCEED	Invalid item number.

Comment None

8 MenuGetNumOfMenuItems

Purpose This function is used to obtain the number of menu items in a specified menu.

Prototype Err MenuGetNumOfMenuItems (ObjectID menu_id,
USHORT *num_items)

Scope Application / Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters

num_items	The number of the menu items in the specified menu object.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

9 MenuGetTotalItems

Purpose This function is used to obtain the total numbers of menu items in a specified menu.

Prototype Err MenuGetTotalItems (ObjectID menu_id,
USHORT *total_num_items)

Scope Application / Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters

total_num_items	The total numbers of the menu items in the specified menu object.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

10 MenuInitMenu

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *MenuInitMenu* function is usually followed by the *MenuDrawMenu* in order to draw the line object on the display.

Prototype Err MenuInitMenu (ObjectID menu_id)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_CANT_CREATE_LOOKUP_TABLE	
ERR_UI_TYPE_MISMATCH	
ERR_UI_CANT_CREATE_POINTER	
ERR_RES_OBJ_MISS	

Comment This function will not draw the menu object on the display

11 MenuInsertItem

Purpose To insert a new item in the menu list and the whole list will then be re-arranged.

Prototype
```
Err MenuInsertItems ( ObjectID menu_id,
                      USHORT item_number,
                      BYTE *item_text)
```

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object .
item_number	The number that you want to add item to.
item_text	Pointer to the char array of item text.

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_NUM_MENU_ITEMS_EXCEED	Invalid item number

Comment None

12 MenuKeyboardHandleMenu

Purpose To handle the direction keys to highlight item in menu object when the menu object is highlighted

Prototype
```
Err MenuKeyboardHandleMenu(EvtType *Event)
```

Scope System

Input Parameters

Event	Pointer to event

Output Parameters None

Result

TRUE	Success.
FALSE	Not handled.

Comment This function only works if the input parameter Event is with the event type of EVT_KEY and the pressed keys are KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT, KEY_PAGEUP and KEY_PAGEDOWN.

13 MenuRestoreBehindBits

Purpose This function is used to restore the bitmap of the image that is covered by the menu object when it is popup

Prototype Err MenuRestoreBehindBits (ObjectID menu_id)

Scope Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function should be called before *MenuEraseMenu* in order to change the display back to normal

14 MenuSaveBehindBits

Purpose This function is used to store the image that is being covered by the specified menu. This function should be called before the specified object is drawn.

Prototype Err MenuSaveBehindBits (ObjectID menu_id)

Scope Internal

Input Parameters

menu_id	The ID value of the specified menu object .

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot

be found.

Comment The function should be called before *MenuDrawMenu*

15 MenuSearchSelectedItem

Purpose This function is used to find out which item is being clicked.

Prototype
```
SHORT MenuSearchSelectedItem ( Menu *addr,
                               SHORT input_xcoord,
                               SHORT input_ycoord)
```

Scope Internal

Input Parameters

addr	Pointer to the menu object
input_xcoord	X-coordinate of the pen.
input_ycoord	Y-coordinate of the pen.

Output Parameters None

Return If the pen is on one of the items, then the item number is returned. If there is no selected item, then NO_SELECTION is returned

Comment None

16 MenuSetAttrVisible

Purpose This function is called to set the attribute menu_visible

Prototype
```
Err MenuSetAttrVisible ( ObjectID menu_id,
                         BOOLEAN menu_visible )
```

Scope Application /Internal

Input Parameters

menu_id	The ID value of the specified menu object .
menu_visible	the attribute value to be set

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object

		that is stored in the RAM cannot be found.

Comment None

17 MenuSetPopupBounds

Purpose This function is used to set the bounds of the popuped menu.

Prototype Err MenuSetPopupBounds (ObjectID menu_id)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

18 MenuSetTotalItems

Purpose This function is used to set up the total number of items in the menu.

Prototype Err MenuSetTotalItems (Object Menu_id,
USHORT total_num_items)

Scope Application/Internal

Input Parameters

menu_id	The ID value of the specified menu object .
total_num_items	An unsigned integer that represents the total numbers of the menu items in the specified menu object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object

		that is stored in the RAM cannot be found.

Comment None

Scheduler Line Object Functions

1 SchlineDeleteSchline

Purpose The memory of the corresponding scheduler is released. It is used when a scheduler object is required to be deleted

Prototype Err SchlineDeleteSchline (ObjectID schline_id)

Scope Application

Input Parameters

schline_id	The ID value of the specified schline object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment Only memory will be released/deleted. Display will stay the same as before. The display should be erased first by calling *SchlineEraseSchline* before the memory is erased by calling *SchlineDeleteSchline*.

2 SchlineDrawSchline

Purpose This function is used to draw the specified scheduler line object in a specified form. Before calling this function, the SchlineInitSchline function should be called first in order to initialize the scheduler line object from the resource file.

Prototype Err SchlineDrawSchline (ObjectID schlineID)

Scope Application

Input Parameters

schline_id	The ID value of the specified schline object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment After successful drawing of the scheduler object, the schline_drawn attribute is set.

3 SchlineDrawSmallSector

Purpose This function is used to draw the specified section in the scheduler line object. Before calling this function, the SchlineInitSchline function should be called first in order to initialize the scheduler line object from the resource file.

Prototype

```
Err SchlineDrawSmallSector (Schline   *schline_addr,
                            SHORT     line_num,
                            SHORT     section_num)
```

Scope Application

Input Parameters

schline_addr	Pointer to the scheduler Line object
line_num	Specified row in the Scheduler Line object (eg 1 ~ 7)
section_num	Specified column in the Scheduler Line object.

Output Parameters None

Result

TRUE	Success.
ERR_UI_INV_INPUT_VALUE	Invalid number.

4 SchlineEraseSchline

Purpose Erase the Scheduler Line object from screen.

Prototype Err SchlineEraseSchline (ObjectID schlineID)

Scope Application

Input Parameters

schline_id	The ID value of the specified schline object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment To specified a scheduler line that doesn't appear on screen and doesn't respond to the pen. Different from SchlineDeleteSchline, this function will keep the object's structure in resource.

5 SchlineGetAttributes

Purpose Retrieve all the relative attributes of the scheduler line object. They are include schline_drawn, schline_active, schline_visible, schline_enter and schline_enable.

Prototype

```
Err SchlineGetSchline (  ObjectID schline_id,
                         BOOLEAN *schline_drawn,
                         BOOLEAN *schline_visible,
                         BOOLEAN *schline_active,
                         BOOLEAN *schline_enable,
                         BOOLEAN *schline_enter )
```

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler Line object .

Output Parameters	schline_drawn	Boolean value to indicate the object is drawn on screen or not.
	schline_visible	Boolean value to indicate the object is visible or not.
	schline_active	Boolean value to indicate the object is active to the pen action or not.
	schline_enable	Boolean value to indicate the state of schline_enable attribute.
	schline_enter	Boolean value to indicate the state of the schline_enter attribute.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

6 SchlineGetClickedRegion

Purpose Retrieve the clicked region of the specified scheduler line.

Prototype

```
BYTE SchlineGetSchline  (  Schline *addr,
                           SHORT input_x,
                           SHORT input_y )
```

Scope Internal

Input Parameters

addr	Pointer to the schline object
input_x	X-coordinate of the pen.
input_y	Y-coordinate of the pen.

Output Parameters None

Result Return the region of the scheduler line being clicked.

SCROLL_NOT_HITTED	Not be clicked,
Line 1 ~ Line 7	A particular number of the line.

Comment None

7 SchlineGetDateBitmaps

Purpose This function is used to get the date's bitmaps of Scheduler Line.

Prototype Err SchlineSetDateBitmaps (ObjectID schline_id,
BYTE date_num,
BYTE *date_bitmaps)

Scope Application

Input Parameters

	schline_id	The ID value of the specified schline object .
	date_num	Date of the selection. (eg 0 = Sunday, 1= Monday)
Output Parameters	date_bitmaps	Pointer reference to the address that points to the date bitmaps.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_ITEM_NUMBER_EXCEED	Invalid number of the selection.

Comment None

8 SchlineGetDateText

Purpose This function is used to get the date's text of Scheduler Line.

Prototype Err SchlineGetDateText (ObjectID schline_id,
BYTE date_num,
BYTE **date_text)

Scope Application

Input Parameters

	schline_id	The ID value of the specified scheduler Line object .
	date_num	Date of the selection. (eg 0 = Sunday, 1= Monday…)
Output Parameters	date_text	Pointer reference to the address that points to the date's text.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object

		that is stored in the RAM cannot be found.
	ERR_UI_ITEM_NUMBER_EXCEED	Invalid number of the selection.

Comment None

9 SchlineInitSchline

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. SchlineDrawSchline function is usually followed by the SchlineInitSchline in order to draw the schedule Line object on the display.

Prototype Err SchlineInitSchline (ObjectID schline_id)

Scope Application

Input Parameters	schline_id	The ID value of the specified schedule Line object .

Output Parameters None

Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function will not draw the scheduler Line object on the display.

10 SchlineSetAttributes

Purpose This function is used to set the Scheduler Line object’s current position, its range, and the size of a page.

Prototype Err SchlineSetSchline (ObjectID schline_id,
BOOLEAN schline_drawn,
BOOLEAN schline_visible,
BOOLEAN schline_active,
BOOLEAN schline_enable,
BOOLEAN schline_enter)

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler Line object .
schline_drawn	Boolean value to set up the state of the schline_drawn attribute.
schline_visible	Boolean value to set up the state of the schline_visible attribute.
schline_active	Boolean value to set up the state of the schline_active attribute.
schline_enable	Boolean value to set up the state of the schline_enable attribute.
schline_enter	Boolean value to set up the state of the schline_enter attribute.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

11 SchlineSetHighlightText

Purpose This function is used to highlight the Line label of the specified row in the Scheduler Line object.

Prototype

```
Err SchlineSetDateText (    ObjectID schline_id,
                            BYTE line_num)
```

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler line object .
line_num	Line number from 1 to 7.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_INV_INPUT_VALUE	Invalid number.

Comment None

12 SchlineSetHourSettings

Purpose This function is used to define the hour mode in the Scheduler Line (e.g 12 / 24 hrs)

Prototype Err SchlineSetDateText (ObjectID schline_id,
BOOLEAN hour_mode)

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler line object .
hour_mode	SCHLINE_MODE_12 or SCHLINE_MODE_24.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

13 SchlineSetLineBitmap

Purpose This function is used to set the specified bitmap in the Scheduler Line object.

Prototype Err SchlineSetLineBitmap (ObjectID schline_id,
SHORT line_num,
SHORT section_num,
BYTE bitmap_num)

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler line object .
line_num	Specified row in the Scheduler Line object (eg 1 ~ 7)
section_num	Specified column in the Scheduler Line object.

	bitmap_num	Specified bitmap in the Scheduler Line object (e.g SCHLINE_TODO_12_TONE ~ to represent a tone alarm (12/hrs) in To Do List)
Output Parameters	None	
Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
	ERR_UI_INV_INPUT_VALUE	Invalid number.
Comment	Maximum number of sction_num depend on the mode of the Scheduler Line object. (e.g 1 ~ 16 (24 hrs mode) / 1 ~ 20 (12 hrs mode))	

14 SchlineSetLineLabel

Purpose This function is used to set the label in the Scheduler Line object.

Prototype
```
Err SchlineSetLineBitmap ( ObjectID schline_id,
                           SHORT line_num,
                           BYTE *schline_label)
```

Scope Application

Input Parameters	schline_id	The ID value of the specified scheduler line object .
	line_num	Specified row in the Scheduler Line object (eg 1 ~ 7)
	schline_label	Pointer to the text.
Output Parameters	None	
Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
	ERR_UI_INV_INPUT_VALUE	Invalid number.
Comment	None	

15 SchlineSetLineState

Purpose This function is used to set a particular section in the Scheduler Line object.

Prototype Err SchlineSetLineState (ObjectID schline_id,
SHORT line_num,
SHORT section_num)

Scope Application

Input Parameters

schline_id	The ID value of the specified scheduler line object .
line_num	Specified row in the Scheduler Line object (eg 1 ~ 7)
section_num	Specified column in the Scheduler Line object.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.
ERR_UI_INV_INPUT_VALUE	Invalid number.

Comment None

Scrollbar Object Functions

1 ScrollbarCalculateCurrentValue

Purpose Calculate the current value of the specified scrollbar object.

Prototype Err ScrollbarCalculateCurrentValue (ObjectID scrollbar_id,
SHORT input_x,
SHORT input_y,
WORD *scrollbar_value)

Scope Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
input_x	X-coordinate of the pen.

	input_y	Y-coordinate of the pen.
Output Parameters	scrollbar_value	Returns the current value of the specified scrollbar object.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

2 ScrollbarCalculateCurrentValue1

Purpose Calculate the current value of the specified scrollbar object.

Prototype

```
Err ScrollbarCalculateCurrentValue1 ( ObjectID scrollbar_id,
                                      SHORT input_x,
                                      SHORT input_y,
                                      WORD *scrollbar_value )
```

Scope Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
input_x	X-coordinate of the pen.
input_y	Y-coordinate of the pen.

Output Parameters	scrollbar_value	Returns the current value of the specified scrollbar object.

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

3 ScrollbarDeleteScrollbar

Purpose This function is used to delete the specified scrollbar object from memory in order to free and release the occupied memory.

Prototype Err ScrollbarDeleteScrollbar (ObjectID scrollbar_id)

Scope Application

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function won't change/update the display. *ScrollbarEraseScrollbar* should be called before calling *ScrollbarDeleteSccrollbar* in order to avoid pen action ion the scrollbar object again.

4 ScrollbarDrawScrollbar

Purpose This function is used to draw the specified scrollbar object in a specified form. Before calling this function, the *ScrollbarInitScrollbar* function should be called first in order to initialise the scrollbar object from the resource file.

Prototype Err ScrollbarDrawScrollbar (ObjectID scrollbarID)

Scope Application

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment The attribute scrollbar_drawn is set to TRUE after the successful drawing.

5 ScrollBarEraseScrollBar

Purpose Erase the Scroll Bar from screen.

Prototype Err ScrollBarEraseScrollBar (ObjectID scrollbarID)

Scope Application

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment The scrollbar_drawn attribute is cleared.

6 ScrollbarGetCarStartCoord

Purpose To calculate the starting coordinate of the scrollcar of the scrollbar

Prototype SHORT ScrollbarGetCarStartCoord(Scrollbar *addr,
WORD scroll_value)

Scope Internal

Input Parameters

addr	Pointer to a scrollbar object
scroll_value	the expected scroll_value

Output Parameters None

Result X-coordinate or y-coordinate (depending on the orientation of the scrollbar)

Comment This function is to calculate the coordinate of the starting point of the scrollcar with expected scrollbar value. If the scrollbar is placed vertically, the returned coordinate will be the y-coordinate of the scrollcar. Otherwise, the returned value is the x-coordinate of the

scrollcar.

7 ScrollbarGetClickedRegion

Purpose Retrieve the clicked region of the specified scrollbar.

Prototype
```
BYTE ScrollbarGetScrollbar  ( Scrollbar *addr,
                              SHORT input_x,
                              SHORT input_y )
```

Scope Internal

Input Parameters

addr	Pointer to the scrollbar object
input_x	X-coordinate of the pen.
input_y	Y-coordinate of the pen.

Output Parameters None

Result Return the region of the scrollbar being clicked.

SCROLL_UP_ARROW	to indicate on the upper arrow of the scrollbar.
SCROLLBAR_UP_REGION	to indicate below the upper arrow and above the scroll car.
SCROLLCAR_REGION	to indicate on the scroll car area.
SCROLLBAR_DOWN_REGION	to indicate below the scroll car and above the lower arrow.
SCROLL_DOWN_ARROW	to indicate on the lower arrow of the scrollbar.

Comment It is assumed that the current position of the pen is within the bounds of the scrollbar object.

8 ScrollbarGetScrollbar

Purpose Retrieve the Scroll Bar’s current position, its range, and the size of a page.

Prototype
```
Err ScrollbarGetScrollbar  ( ObjectID scrollbar_id,
                             WORD *value,
                             WORD *max_value,
                             WORD *min_value,
```

WORD *pagesize,
WORD *total_num_lines)

Scope Application

Input Parameters

	scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters

value	Returns the current value (position) of the scrollbar.
max_value	Returns the maximum value of the scrollbar.
min_value	Returns the minimum value of the scrollbar.
pagesize	Returns the size of a page (used when page scrolling).
total_num_lines	Returns the total number of lines of the scrollbar.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment It is advised to call *ScrollbarGetScrollbar* before calling *ScrollbarSetScrollbar* in order to get the previous values of the scrollbar object

9 ScrollbarGetScrollbarText

Purpose This function is called to get the text of the scrollbar object

Prototype Err ScrollbarGetScrollbarText (ObjectID scrollbar_id,
BYTE **scrollbar_text)

Scope Application

Input Parameters

	scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters

scrollbar_text	Pointer reference to the text of the scrollbar object

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

10 ScrollbarGetScrollbarVisible

Purpose This function is called in order to get the attribute scrollbar_visible of a scrollbar object.

Prototype Err ScrollbarGetScrollbarVisible(ObjectID scrollbar_id,
BOOLEAN *visible);

Scope Application

Input Parameters	scrollbar_id	The ID value of the specified scrollbar object .
Output Parameters	visible	Pointer to the boolean attirbute scrollbar_visible

Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

11 ScrollbarHardButtonSetScrollbar

Purpose This function is called to generate a EVT_SCROLLBAR_SELECT event while the scroll car is moved up or down one step

Prototype ScrollbarHardButtonSetScrollbar(ObjectID scrollbar_id,
BYTE UP_DOWN);

Scope Application

Input Parameters	scrollbar_id	The ID value of the specified scrollbar object .
	UP_DOWN	To indicate the movement of the scroll car. SCROLLBAR_UP_ARROW means the scroll car is moved towards left or right one step. In contrast, SCROLLBAR_DOWN_ARROW

		means the scroll car is moved towards right or bottom one step.

Output Parameters None

Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

12 ScrollbarInitScrollbar

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *ScrollbarDrawScrollbar* function is usually followed by the *ScrollbarInitScrollbar* in order to draw the scroll bar object on the display.

Prototype Err ScrollbarInitScrollbar (ObjectID scrollbar_id)

Scope Application

Input Parameters	scrollbar_id	The ID value of the specified scrollbar object .

Output Parameters None

Return	TRUE	Success.
	ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment This function will not draw the scroll bar object on the display

13 ScrollbarSetScrollbar

Purpose This function is used to set the Scrollbar’s current position, its range, and the size of a page.

Prototype Err ScrollBarSetScrollBar (ObjectID scrollbar_id,

WORD scrollbar_value,
WORD scrollbar_min_value,
WORD scrollbar_max_value,
WPRD scrollbar_pagesize,
WORD scrollbar_total_num_lines)

Scope Application/Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
scrollbar_value	Current value of the scroll bar.
scrollbar_min_value	Minimum value of the scroll bar.
scrollbar_max_value	Maximum value of the scroll bar.
scrollbar_pagesize	Size of a page (used when page scrolling).
scrollbar_total_num_lines	Total number of lines of the scrollbar.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment The display of the scrollbar will not be updated automatically and accordingly.

14 ScrollbarSetScrollbarDrawPagesize

Purpose This function is called in order to set the draw page size of a scrollbar object.

Prototype Err ScrollbarSetScrollbarDrawPage(ObjectID scrollbar_id,
SHORT draw_pagesize);

Scope Application/Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
draw_pagesize	The value of the draw page size.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object

	that is stored in the RAM cannot be found.

Comment This function MUST BE called to set the pagesize of the scrollbar in order to display the scrollbar properly.

15 ScrollbarSetScrollbarText

Purpose This function is called to set the text of the scrollbar

Prototype Err ScrollbarSetScrollbarText (ObjectID scrollbar_id,
BYTE *scrollbar_text)

Scope Application/Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
scrollbar_text	The text pointer

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment None

16 ScrollbarSetScrollbarType

Purpose Set the type of the Scroll Bar. By default, there are three types for the scrollbar object.

Prototype Err ScrollbarSetScrollbarType (ObjectID scrollbar_id,
BYTE scrollbar_type)

Scope Application

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
scrollbar_type	Define the type of the scroll bar: SCROLLBAR_STYLE_0 = Windows type

		Scrollbar. SCROLLBAR_STYLE_1 = Volume type scrollbar with highlight region. SCROLLBAR_STYLE_2 = Volume type scrollbar without highlight region.

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment *ScrollbarDrawScrollbar* should be called in order to change the display of the scrollbar to the new type

17 ScrollbarSetScrollbarVisible

Purpose This function is called in order to set the attribute scrollbar_visible of a scrollbar object.

Prototype Err ScrollbarGetScrollbarVisible(ObjectID scrollbar_id,
BOOLEAN visible);

Scope Application/Internal

Input Parameters

scrollbar_id	The ID value of the specified scrollbar object .
visible	The new value of the attribute scrollbar_visible

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comment If the attribute is set to TRUE, it means that the scrollbar can be drawn on the display. Otherwise, the scrollbar is invisible.

String Object Functions

1 StringDeleteString

Purpose This function is used to free/release the memory that is used by the specified string object. This function should follow the use of *StringEraseString*.

Prototype Err StringDeleteString(ObjectID string_id)

Scope Application/Internal

Input Parameters

string_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment The display will stay unchanged. The display should be erased from the display before removing from the memory in order to avoid any pen action after the removing the string object from memory

2 StringDrawString

Purpose This function is used to draw the specified string object in specified form. This function should only be used after the calling of *StringInitString* function.

Prototype Err StringDrawString(ObjectID string_id)

Scope Application/Internal

Input Parameters

string_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be

	found

Comment The string_drawn attribute is set as TRUE to show that the string object is drawn on the display already

3 StringEraseString

Purpose This function is used to erase/hide the string object in a form. The memory of the string object will not be changed.

Prototype Err StringEraseString(ObjectID string_id)

Scope Application/Internal

Input Parameters

string_id	The ID value of the specified line object .

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment The string_drawn attribute is set to FALSE in order to indicate that the string object is not on the display

4 StringGetAttribute

Purpose This function is called to get the attributes of the specified string object.

Prototype Err StringGetAttribute(ObjectID string_id, BOOLEAN *att_drawn, BOOLEAN *att_visible)

Scope Application/Internal

Input Parameters

string_id	The ID value of the specified line object .

Output Parameters

att_drawn	Pointer to boolean variable to show whether the string is already drawn on the display or not
att_visible	Pointer to boolean variable to show whether the string is already visible on the display or not

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

5 StringGetText

Purpose This function is for application to read back the text of a specified string object.

Prototype Err StringGetText(ObjectID string_id, BYTE **text)

Scope Application/Internal

Input Parameters

stringID	the ID value of the specified string

Output Parameters

text	Pointer reference to the text of the string object

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

6 StringInitString

Purpose This function is used to initialize the data structure of a string object from the resource file. After this function, the string object can be drawn.

Prototype Err StringInitString(ObjectID string_id)

Scope Application/Internal

Input Parameters

stringID	the ID value of the specified string

Output Parameters None

Result ERR_UI_TYPE_MISMATCH

	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment This function will not draw the string object on the display.

6 StringSetAttribute

Purpose This function is to set the attributes of a string object.

Prototype Err StringSetAttribute(ObjectID string_id, BOOLEAN att_drawn, BOOLEAN att_visible)

Scope Application/Internal

Input Parameters

string_id	The ID value of the specified line object .
att_drawn	string_drawn attribute
att_visible	string_visible attribute

Output Parameters None

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment None

7 StringSetText

Purpose This function is used to set the text in the string object that will be displayed on the screen.

Prototype Err StringSetText(ObjectID string_id, BYTE *text)

Scope Application/Internal

Input Parameters string_id The ID value of the specified line object .
text Pointer to the memory for holding the text

Output Parameters None

Result

TRUE	Success.
ERR_UI_RES_NOT_FOUND	the data structure of the object that is stored in RAM cannot be found

Comment *StringDrawString* should be called afterwards in order to update the changes

Table Object Functions

1 TableCheckCellHasBitmap

Purpose This function is used to find out the specified cell has a bitmap or not.

Prototype

```
Err TableCheckCellHasBitmap (ObjectID table_id,
                             USHORT row_num,
                             USHORT col_num,
                             BOOLEAN *cell_bitmap )
```

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.
row_num	The row number of the specific table.
col_num	The column number of the specific table.

Output Parameters

cell_bitmap	Returns a boolean value to indicate the specific table contains a bitmap or not. TRUE = Has bitmap, FALSE = No bitmap.

Result

TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment Actually, this function is particular used by the Scheduler application.

2 TableCheckCellHighlight

Purpose This function is used to check whether the specific cell is being selected or not.

Prototype
```
Err TableCheckCellHighlight (ObjectID table_id,
                             USHORT row_num,
                             USHORT col_num,
                             BOOLEAN *cell_highlight )
```

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.
row_num	The row number of the specific table.
col_num	The column number of the specific table.

Output Parameters

cell_highlight	Returns a boolean value to indicate the specific cell is being selected or not. TRUE = Selected, FALSE = Not selected.

Result
TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

3 TableCheckHighlightEnable

Purpose This function is used to check whether the specific cell can be highlighted after selection or not.

Prototype
```
Err TableCheckCellHighlight (ObjectID table_id,
                             BOOLEAN highlight_enable )
```

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.
highlight_enable	TRUE = Selected, FALSE = Not selected.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment If the highlight_enable is set to TRUE, that means after selection of a particular cell, the cell will be highlighted. If highlight_enable is set to FALSE, then no highlight will be done after selection of the cell.

4 TableDeleteTable

Purpose This function is used to delete the specified table object from memory in order to free and release the occupied memory.

Prototype Err TableDeleteTable (ObjectID table_id)

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment This function won't change/update the display. *TableEraseTable* should be called before calling *TableDeleteTable* in order to avoid any other pen action on the table object.

5 TableDrawDisplay

Purpose This function is used to draw a value, number or text in the table object.

Prototype

```
Err TableDrawDisplay (ObjectID table_id,
                      ObjectBounds cell,
                      USHORT count,
                      BYTE cell_color )
```

Scope Internal

Input Parameters

table_id	The ID value of the specified table object.
cell	The bounds of the specified cell in the table.
count	The specified cell number.

	cell_color	Indicate to invert the cell color and cell background or not.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

6 TableDrawTable

Purpose This function is used to draw the specified table and all the objects contain in it. Before calling this function, the *TableInitTable* function should be called first in order to initialise the table and all relate objects from the resource file.

Prototype Err TableDrawTable (ObjectID table_id)

Scope Application/Internal

Input Parameters	table_id	The ID value of the specified table object.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment This function will erase the region behind the table before drawing. In addition, the table_drawn attribute is set to TRUE

7 TableDrawTableBounds

Purpose This function is used to draw the frame of the specified table object.

Prototype Err TableDrawDisplay (ObjectID table_id)

Scope Internal

Input Parameters	table_id	The ID value of the specified table object.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment Draw the frame or not depend on the table_style.

8 TableDrawUIObject

Purpose This function is used to draw the relative objects in the table.

Prototype
```
Err TableDrawUIObject (ObjectID table_id,
                       Object table_item_id,
                       Object table,
                       Object cell )
```

Scope Internal

Input Parameters

table_id	The ID value of the specified table object.
table_item_id	The ID value of the relative UI object.
table	The bounds of the specified table.
cell	The bounds of the specified cell in the table.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

9 TableEnableTable

Purpose This function is used to enable the table in order to let user to select items in the table

Prototype
```
Err TableEnableTable (ObjectID table_id,
                      BOOLEAN enable_attr )
```

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.
enable_attr	The status of the table_enable attribute. TRUE = editable, FALSE = Non editable.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment After calling this function, the table object will accept pen action on it

10 TableEraseNotUIRegion

Purpose This fucntion is called to erase the regions in a cell that are not covered by the UI object in a particular cells

Prototype Err TableEraseNotUIRegion(ObjectID table_item_id,
ObjectBounds table_cell)

Scope Internal

Input Parameters

table_item_id	The Id value of the item in a table
table_cell	The object bounds of the table cell

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

11 TableEraseTable

Purpose This function is used to erase the table object on the display

Prototype Err TableEraseTable (ObjectID table_id)

Scope Application/Internal

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment The table_drawn attribute is cleared.

12 TableGetAttributes

Purpose The function is called in order to retrieve all 5 attributes for a table object. The attributes are table_drawn, table_scrollbar, table_enable, table_active, table_enter, table_visible, table_highlight_enable.

Prototype
```
Err TableGetAttributes ( ObjectID table_id,
                         BOOLEAN *enable_attr,
                         BOOLEAN *drawn_attr,
                         BOOLEAN *active_attr,
                         BOOLEAN *visible_attr
                         BOOLEAN *scrollbar_attr )
```

Scope Application/Internal

Input parameters

table_id	The ID value of the specified table object.

Output parameters

enable_attr	Returns a Boolean value that set up the state of the table_enable attribute.
drawn_attr	Returns a Boolean value that set up the state of the table_drawn attribute.
active_attr	Returns a Boolean value that set up the state of the table_active attribute.
visible_attr	Returns a Boolean value that set up the state of the table_visible attribute.
scrollbar_attr	Returns a Boolean value that set up the state of the table_scrollbar attribute.

Return

TRUE	Success.
ERR_UI_RES_NOT_FOUND	The data structure of the object that is stored in the RAM cannot be found.

Comments None

13 TableGetCellBounds

Purpose This function is used to get the bounds of a specified cell in the table.

Prototype
```
Err TableGetCellBounds (ObjectID table_id,
                        USHORT *row_number,
                        USHORT *column_number,
                        ObjectBounds *cell_bounds )
```

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters

row_number	Returns the row number of the specific table.
column_number	Returns the column number of the specific table.
cell_bounds	Returns the bounds of the particular cell in the table.

Result
TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

14 TableGetClickedCell

Purpose This function is used by input the X & Y coordinate of the pen, and return the row, column and cell number of a particular cell in the table.

Prototype
```
Err ControlPopupClickedRegion ( ObjectID table_id,
                                SHORT x_input,
                                SHORT y_input,
                                USHORT *row_number,
                                USHORT *col_number,
                                USHORT *cell_number )
```

Scope Application

Input parameters

table_id	The ID value of the specified table object.
x_input	X-coordinate of the pen.
y_input	Y-coordinate of the pen.

Output parameters

row_number	Returns the row number of the specific table.
column_number	Returns the column number of the specific table.
cell_number	Returns the cell number that the pen pointed to.

Return

TRUE	Success search for a cell.
FALSE	Search for a cell not success.
ERR_UI_RES_NOT_FOUND	
ERR_UI_INV_INPUT_COORD	

Comments If the number of items in the popup list are more than the screen can display, then an upper or a lower arrow will appear in the upper / lower right corner of the particular popup list.

15 TableGetColumnWidth

Purpose This function is used to get the width of a particular column in a table

Prototype

```
Err TableGetColumnWidth (ObjectID table_id,
                         USHORT column_number,
                         SHORT *column_width )
```

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.
column_number	The column number of the specific table.

Output Parameters

column_width	Returns the width of the specific cell in the table.

Result

TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_COL_NUM

Comment None

16 TableGetItemText

Purpose This function is used to get text string of the specified item in a table

Prototype
```
Err TableGetItemText (ObjectID table_id,
                      USHORT row_number,
                      USHORT column_number
                      BYE **item_text )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object.
row_number	The row number of the specific table.
column_number	The column number of the specific table.

Output Parameters

item_text	Return is a pointer reference to the text of the item.

Result
TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

17 TableGetItemType

Purpose This function is used to get the type of the specified item in a table.

Prototype
```
Err TableGetItemType (ObjectID table_id,
                      USHORT row,
                      USHORT column,
                      BYTE *table_data_type )
```

Scope Application / OS

Input Parameters

table_id	The ID value of the specified table object.
row	The row number of the specific table.
column	The column number of the specific table.

Output Parameters

table_data_type	Returns the type of the specified item in the table.	
	TABLE_TEXT	to indicate a text is placed in the specified cell.
	TABLE_VALUE	to indicate a value is placed in the specified cell.

	TABLE_UI_OBJECT	to indicate an UI object is placed in the specified cell.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

18 TableGetItemValue

Purpose This function is used to get the value of the specified item in a table

Prototype
```
Err TableGetItemValue (ObjectID table_id,
                       USHORT row,
                       USHORT column,
                       WORD *table_value )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object.
row	The row number of the specific table.
column	The column number of the specific table.

Output Parameters

table_value	Returns a value or a number.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

19 TableGetNumOfColumns

Purpose This function is used to get the total number of columns in a table.

Prototype
```
Err TableGetNumOfColumns (ObjectID table_id,
                          USHORT *num_columns )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters num_columns — Returns an unsigned integer to show the total number of columns in the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

20 TableGetNumOfRows

Purpose This function is used to get the total number of rows in a table.

Prototype Err TableGetNumOfRows (ObjectID table_id,
USHORT *num_rows)

Scope Application

Input Parameters table_id — The ID value of the specified table object.

Output Parameters num_rows — Returns an unsigned integer to show that the total number of rows in the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

21 TableGetNumOfRowsDisplayed

Purpose This function is used to find out number of rows is being displayed on the table.

Prototype Errt TableGetNumOfRowsDisplayed (ObjectID table_id,
USHORT *num_row)

Scope Application / Internal

Input Parameters table_id — The ID value of the specified table object.

Output Parameters num_row — Returns the number of rows is being displayed in the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

22 TableGetRowColOfSelection

Purpose This function is used to get the row number and column number of the specified table.

Prototype
```
Err TableGetRowColOfSelection (ObjectID table_id,
                               USHORT *row_number,
                               USHORT *col_number )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters

row_number	Returns the number of row in the specified table.
col_number	Returns the number of column in the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

23 TableGetRowHeight

Purpose This function is used to get the height of a particular row in the table

Prototype
```
Errt TableGetRowHeight (ObjectID table_id,
                        USHORT row_num,
                        SHORT *row_height )
```

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.
row_num	A specified row number in the table.

Output Parameters

row_height	Returns the height of a particular row in the table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

24 TableGetTableBounds

Purpose This function is used to get the bounds of a table.

Prototype Err TableGetTableBounds (ObjectID table_id,
ObjectBounds *table_bounds)

Scope Application / Internal

Input Parameters table_id The ID value of the specified table object.

Output Parameters table_bounds Returns the bounds of the table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

25 TableGetTopRowNum

Purpose This function is used to find out the top row number of the specified table.

Prototype Errt TableGetNumOfRowsDisplayed (ObjectID table_id,
USHORT *top_row)

Scope Application / Internal

Input Parameters table_id The ID value of the specified table object.

Output Parameters top_row Returns the top row number of the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

26 TableInitTable

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. TableInitTable is usually followed by the TableDrawTable in order to draw the form on the display.

Prototype Err TableInitTable (ObjectID table_id)

Scope Application

Input Parameters table_id The ID value of the specified table object.

Output Parameters None

Return

Comment This function will not draw the table object on the display

27 TablelSetAttributes

Purpose This function is called in order to set 5 attributes of the table object. The attributes are table_enable, table_drawn, table_active, table_visible and table_scrollbar.

Prototype Err TableSetAttributes (ObjectID table_id,
BOOLEAN att_drawn,
BOOLEAN att_enable,
BOOLEAN att_active,
BOOLEAN att_visible
BOOLEAN att_scrollbar)

Scope Application

Input parameters

table_id	The ID value of the specified table object
att_drawn	Boolean value to set up the state of the table_drawn attribute.
att_enable	Boolean value to set up the state of the table_enable attribute.
att_active	Boolean value to set up the state of the table_active attribute.
att_visible	Boolean value to set up the state of the table_visible attribute.
att_scrollbar	Boolean value to set up the state of the table_scrollbar attribute.

Output parameters None

Return TRUE
ERR_UI_RES_NOT_FOUND

Comments None

28 TableSetBounds

Purpose This function is used to set up the X-coordinate, Y-coordinate, width and height of the table.

Prototype Err TableSetBounds (ObjectID table_id,
ObjectBounds table_bounds)

Scope Application

Input Parameters

table_id	The ID value of the specified table object
table_bounds	Pointer to the bounds structure of the specified table.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

29 TableSetCellHasBitmap

Purpose This function is used to set or clear whether there is a bitmap in a specified cell.

Prototype
```
Err TableSetCellHasBitmap (ObjectID table_id,
                           USHORT row_num,
                           USHORT col_num,
                           BOOLEAN bitmap_on )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object
row_num	The row number of the specific table.
col_num	The column number of the specific table.
bitmap_on	Boolean value to indicate whether there is a bitmap in the specified cell. Set = TRUE, Clear = FALSE.

Output Parameters None

Result
TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment Actually, this function is particular used by the Scheduler application.

30 TableSetColumnWidth

Purpose This function is used to set the width of a particular column in the table.

Prototype
```
Err TableSetColumnWidth (ObjectID table_id,
                         USHORT col_number,
                         SHORT new_width )
```

Scope Application

Input Parameters

table_id	The ID value of the specified table object
col_number	The number of the specific column.
new_width	The width of the specified column.

Output Parameters None

Result
TRUE
ERR_UI_RES_NOT_FOUND

ERR_UI_INVALID_COL_NUM

Comment None

31 TableSetHighlightCell

Purpose This function is used to set or clear the table_cell_highlight attribute of a particular cell in the table.

Prototype Err TableSetHighlightCell (ObjectID table_id,
USHORT row_num,
USHORT col_num,
BOOLEAN highlight_on)

Scope Application

Input Parameters

table_id	The ID value of the specified table object
row_num	The row number of the specific table.
col_num	The column number of the specific table.
highlight_on	Boolean value to set up the state of the table_cell_highlight attribute. Set = TRUE, Clear = FALSE.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

32 TableSetHighlightEnable

Purpose This function is used to set or clear the table_highlight_enable attribute of a particular table.

Prototype Err TableSetHighlightEnable (ObjectID table_id,
BOOLEAN highlight_enable)

Scope Application

Input Parameters

table_id	The ID value of the specified table object
highlight_enable	Boolean value to set up the state of the table_highlignt_enable attribute. Set = TRUE, Clear = FALSE.

Output Parameters None

Result
TRUE
ERR_UI_RES_NOT_FOUND

Comment None

33 TableSetItemText

Purpose This function is used to set the text string of a specified cell in the table.

Prototype
```
Err TableSetItemText (ObjectID table_id,
                      USHORT row,
                      USHORT column,
                      BYTE *text_string)
```

Scope Application

Input Parameters

table_id	ID of the table object
row	The number of the specific table.
column	The column number of the specific table.
*text_string	Pointer pointed to the text of the item.

Output Parameters None

Result
TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

34 TableSetItemType

Purpose This function is used to set the data type of a specified cell in the table.

Prototype
```
Err TableSetItemType (ObjectID table_id,
                      USHORT row_number,
                      USHORT col_number,
```

BYTE new_type)

Scope Application

Input Parameters

table_id	ID of the table object
row_number	The row number of the specific table.
col_number	The column number of the specific table.
new_type	The data type of a specified item in the table. 0 = TABLE_TEXT, 1 = TABLE_VALUE, 2 = TABLE_UI_OBJECT.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

35 TableSetItemValue

Purpose This function is used to set the value /number of a specified item in the table.

Prototype Err TableSetItemValue (ObjectID table_id,
USHORT col_number,
USHORT row_number,
WORD new_value)

Scope Application

Input Parameters

table_id	ID of the table object
row_number	The row number of the specific table.
col_number	The column number of the specific table.
new_value	The value / number of a specified item in the table.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM
ERR_UI_INVALID_COL_NUM

Comment None

36 TableSetRowHeight

Purpose This function is used to set the height of a particular row in the table.

Prototype Err TableSetColumnWidth (ObjectID table_id,
USHORT row_num,
SHORT row_height)

Scope Application

Input Parameters

table_id	The ID value of the specified table object
row_num	The number of the specific row.
row_height	The height of the specified row.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_COL_NUM

Comment None

37 TableSetTopRowNum

Purpose This function is used to set the top row number of the specified table.

Prototype Errt TableSetNumOfRowsDisplayed (ObjectID table_id,
USHORT top_row)

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters

top_row	The top row number of the specified table.

Result TRUE
ERR_UI_RES_NOT_FOUND
ERR_UI_INVALID_ROW_NUM

Comment None

38 TableUpdateNumRowDisplay

Purpose This function is used to update the number of rows that are displayed on the screen

Prototype BOOLEAN TableUpdateNumRowDisplay(ObjectID table_id

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.

Output Parameters None

Result

TRUE
ERR_UI_RES_NOT_FOUND

Comment None

39 TableUpdateObjectScreenBounds

Purpose This function is used to get the bounds of a table.

Prototype BOOLEAN TableUpdateObjectScreenBounds(ObjectID table_id, ObjectID object_id)

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.
object_id	The ID value of the object that the bounds is required to update

Output Parameters None

Result

TRUE
ERR_UI_RES_NOT_FOUND

Comment None

40 TableUpdateTable

Purpose This function is used to rearrange the top row and redraw the specified table on the screen.

Prototype Errt TableUpdateTable (ObjectID table_id,
USHORT top_row_num)

Scope Application / Internal

Input Parameters

table_id	The ID value of the specified table object.
top_row_num	Top row number will then be updated to the specified table.

Output Parameters None

Result TRUE
ERR_UI_RES_NOT_FOUND

Comment None

Textbox Object Functions

1 TextboxAddKeyInChar

Purpose This function is called in order to add a character to the textbox object at the position of the insertion point. After a character is added, the insertion point will be moved accordingly too.

Prototype Err TextboxAddKeyInChar(ObjectID textbox_id, BYTE key)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the specified textbox object
key	ASCII code of the key that will be added

Output Parameters None

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

2 TextboxCopy

Purpose Copy the current highlighted selection in an active textbox object to the text clipboard

Prototype Err TextboxCopy (ObjectID textbox_id)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The textbox object must be active currently and highlight is on

3 TextboxCut

Purpose Copy the current selection to the text clipboard, delete the selection from the textbox

Prototype Err TextboxCut (ObjectID textbox_id)

Scope Internal

Input Parameters textbox_id The ID value of the textbox object

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The function only works when the textbox object is the active one and the attribute textbox_highlight is must be set. The content in the clipboard is replaced by the current highlighted text. The function

does not redraw the textbox object.

4 TextboxCutBackspaceHighlightText

Purpose To delete highlighted text in a textbox object when backspace is keyed-in.

Prototype void TextboxCutBackspaceHighlightText(Teextbox *addr)

Scope Internal

Input Parameters addr Pointer to textbox object

Output Parameters None

Return None

Comment Undo buffer is used to save the cut highlighted text

5 TextboxCutSelectedText

Purpose This function is a routine function to cut the highlighted text from a textbox object. This function is called by *TextboxCut* and *TextboxInsertText.*

Prototype void TextboxCutSelectedText (Textbox *addr)

Scope Internal

Input Parameters addr Pointer to textbox object

Output Parameters None

Return None

Comment This function is the main core function of CUT. This function cannot be used alone and it should be called by other function.

6 TextboxDelete

Purpose This function is a routine function and is used to delete a range of characters from a textbox object

Prototype Err TextboxDelete (ObjectID textbox_id, WORD start_char, WORD cut_length)

Scope Internal

Input Parameters

textbox_id	The ID value of the textbox object.
start_char	Starting character position of the character.
cut_length	The number of characters to be deleted

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After deletion, the textbox object should be redrawn again.

7 TextboxDeleteString

Purpose This function is called to delete a section of text in the textbox object

Prototype Err TextboxDeleteString(ObjectID textbox_id, WORD start_char, WORD cut_length)

Scope Application

Input Parameters

textbox_id	The ID value of the textbox object
start_char	The starting character position of section of text
cut_length	The number of characters that required to be delete

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The textbox object will not be redrawn

8 TextboxDeleteTextbox

Purpose The memory of the corresponding textbox is released. It is used when a textbox object is required to be deleted

Prototype Err TextboxDeleteTextbox(ObjectID textbox_id)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment Only memory will be released/deleted. Display will stay the same as before. As the memory is released, therefore, the textbox should also be erased in order to avoid the actions on the display by the user. The function should follow the calling of the function TextboxEraseTextbox.

9 TextboxDirty

Purpose It is used to check whether user has modified the textbox object or not. It means to return the attribute textbox_dirty.

Prototype Err TextboxDirty(ObjectID textbox_id, BOOLEAN *dirty)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object.

Output Parameters	dirty	Pointer to variable to show textbox_dirty attribute

Return TRUE No Error
ERR_UI_RES_NOT_FOUND The required object not found

Comment The attribute textbox_dirty is set by the core of UI internally when the content of the textbox is changed. The function *TextboxSetDirty* should be called to set the new value for the attribute.

10 TextboxDrawRoundTextbox

Purpose This function is used to draw specified round corner Textbox.

Prototype Void TextboxDrawRoundTextbox (ObjectBounds * bounds)

Scope Internal

Input Parameters	bounds	Pointer to the Textbox structure.
Output Parameters		

Result None

Comment None

11 TextboxDrawTextbox

Purpose Draw the text of the textbox and set the textbox_drawn attribute. It will draw the frame, the text, the insertion point and highlighted section of a textbox object.

Prototype Err TextboxDrawTextbox (ObjectID textbox_id)

Scope Application/Internal

Input Parameters	textbox_id	The ID value of the textbox object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment If textbox_highlight is set, then selected text is highlighted.
If textbox_insert_pt_visible is set, then insertion point is on.
This function should be called if the textbox is required to draw or redraw.

12 TextboxDrawTextbox2

Purpose Draw the text of the textbox and set the textbox_drawn attribute. It will draw the frame, the text, the insertion point and highlighted section of a textbox object.

Prototype Err TextboxDrawTextbox (ObjectID textbox_id)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment This function won't erase the textbox object before redrawing.

13 TextboxEraseTextbox

Purpose This function is used to erase the textbox object from the display.

Prototype Err TextboxEraseTextbox(ObjectID textbox_id)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND

		The required object not found

Comment This function doesn't modify the contents of the textbox. If the textbox_insert_pt_visible is set, the inserting point is turned off. Clear the object from screen and the data of the object is still in memory. The attribute textbox_drawn is cleared. This function should be called before the call of TextboxDeleteTextbox.

14 TextboxGetAttributes

Purpose This function is used to get the attributes of a specified textbox object

Prototype Err TextboxGetAttribute(ObjectID textbox_id,
TextboxAttr *textbox_attr)

Scope Application/Internal

Input Parameters	textbox_id	The ID value of the textbox object.
Output Parameters	textbox_attr	Pointer to attribute structure of a textbox object.

Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

15 TextboxGetCurrentHighlightedSelection

Purpose This function is used to get parameters of the highlighted selection of text specified textbox object

Prototype Err TextboxGetCurrentHighlightedSelection(ObjectID textbox_id,
WORD *star_char_pos,
WORD *end_char_pos)

Scope Application/Internal

Input Parameters	textbox_id	The ID value of the textbox object.

Output Parameters	start_char_pos	Pointer to variable of character position of starting character of a highlight section
	end_char_pos	Pointer to variable of character position of end character of a highlight section
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment If there is no highlighted text (textbox_highlight is cleared), then both *start_char_pos and *end_char_pos are –1.

16 TextboxGetFont

Purpose This function is called to get the font type of the textbox object.

Prototype Err TextboxGetFont(ObjectID textbox_id, BYTE *font_id)

Scope Application/Internal

Input Parameters	textbox_id	The ID value of the textbox object.
Output Parameters	font_id	Pointer to variable that stores the font type of the textbox object
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment The output of the function is textbox_font_id in the data structure of a textbox object.

17 TextboxGetInsertPointPosition

Purpose This function is called in order to get the character position of the insertion point.

Prototype Err TextboxGetInsertPointPosition(ObjectID textbox_id, WORD *char_pos)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the textbox object.

Output Parameters

char_pos	Pointer to variable that stores the character position of the insertion point

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment The output of this function is textbox_insert_pt_char_pos of a textbox object. This function only works when there is an insertion point flashing on the display. Otherwise the char_pos will be –1.

18 TextboxGetLeftCharPos

Purpose To return the character position of the leftmost displayed character in the textbox.

Prototype Err TextboxGetLeftCharPos(ObjectID textbox_id,
WORD *left_char_pos)

Scope Application

Input Parameters

textbox_id	The ID value of the textbox object.

Output Parameters

left_char_pos	Pointer to the character position of the leftmost character in the textbox.

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

19 TextboxGetMaxNumChars

Purpose This function is called in order to give the maximum number of characters that the textbox object can store.

Prototype Err TextboxGetMaxNumChars(ObjectID textbox_id,

WORD *max_num_of_chars)

Scope Application/Internal

Input Parameters textbox_id — The ID value of the textbox object.

Output Parameters — max_num_of_chars — Pointer to the variable that stores the maximum number of characters that is allowed to hold by the textbox object

Return TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment If the current number of characters in the textbox is equal to the maximum number of characters, then adding new characters to the textbox is not allowed.

20 TextboxGetNumOfChars

Purpose This function gives the current number of characters in a textbox object

Prototype Err TextboxGetNumOfChars(ObjectID textbox_id,
WORD *num_chars)

Scope Application/Internal

Input Parameters textbox_id — The ID value of the textbox object.

Output Parameters — num_chars — Pointer to the current number of characters in a textbox object

Return TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

21 TextboxGetNumOfCharsDisplayed

Purpose This function gives the current number of characters, which are being displayed, in a textbox object

Prototype Err TextboxGetNumOfCharsDisplayed(ObjectID textbox_id, WORD *num_chars)

Scope Application/Internal

Input Parameters textbox_id — The ID value of the textbox object.

Output Parameters num_chars — Pointer to the current number of characters, which are being displayed, in a textbox object

Return TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

22 TextboxGetRightCharPos

Purpose To return the character position of the rightmost displayed character in the textbox.

Prototype Err TextboxGetRightCharPos(ObjectID textbox_id, WORD *right_char_pos)

Scope Application

Input Parameters textbox_id — The ID value of the textbox object.

Output Parameters right_char_pos — Pointer to the character position of the rightmost character in the textbox.

Return TRUE — No Error
ERR_UI_RES_NOT_FOUND — The required object not found

Comment None

23 TextboxGetTextPointer

Purpose This function is called in order to get the text pointer of the stored string in textbox object.

Prototype Err TextboxGetTextPointer(ObjectID textbox_id, BYTE **text_ptr)

Scope Application/Internal

Input Parameters	textbox_id	The ID value of the textbox object.
Output Parameters	text_ptr	Pointer reference to the string of a specified textbox object
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment After the change of the content that is pointed by the returned pointer, the content of the textbox can be changed accordingly.

24 TextboxGetTextboxBounds

Purpose Return the current bounds of a textbox.

Prototype Err TextboxGetTextboxBounds(ObjectID textbox_id,
ObjectBounds *bounds)

Scope Application

Input Parameters	textbox_id	The ID value of the textbox object.
Output Parameters	bounds	Pointer to the bounds structure.
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

25 TextboxInitTextbox

Purpose This function is used to load the data structure and initialize it from a resource file in the RAM. *TextboxInitTextbox* is usually followed by the *TextboxDrawTextbox* in order to draw the textbox object on the display. Object must be initialized once before using it without problem.

Prototype Err TextboxInitTextbox(ObjectID textbox_id)

Scope Application

Input Parameters

textbox_id	The ID value of the specified textbox

Output Parameters None

Return

ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created
ERR_RES_OBJ_MISS	Object ID not found.

Comment If the object is already loaded into the memory before the initialization, then the content of the objects will be initialized once again.

26 TextboxInsert

Purpose This function is called to paste text string from clipboard to a specified textbox object. The position for pasting is the position of the insertion point.

Prototype Err TextboxInsert (ObjectID textbox_id, WORD *paste_size)

Scope Internal

Input Parameters

textbox_id	The ID value of the textbox object.

Output Parameters	paste_size	Pointer to the size of the text in the clipboard
Return	TRUE	No Error
	ERR_UI_RES_NOT_FOUND	The required object not found

Comment The insertion point will be moved accordingly.

27 TextboxInsertFunction

Purpose This function is the core function for pasting a string into the content of the textbox object. This function can only be called by TextboxInsertText function.

Prototype
```
Void TextboxInsertFunction(Textbox *addr,
                           BYTE *paste_text,
                           WORD paste_size)
```

Scope Internal

Input Parameters	addr	Pointer to textbox object
	paste_text	Pointer to the text that will be pasted
	paste_size	The size of the text that will be pasted
Output Parameters	None	

Return None

Comment The insertion point will not be adjusted accordingly.

28 TextboxInsertPtMove

Purpose This function is called in order to calculate the position of the insertion point one step right or left.

Prototype
```
Err TextboxInsertPtMove (ObjectID textbox_id)
```

Scope Internal

Input Parameters textbox_id The ID value of the specified textbox

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The direction of movement is determined by the parameter textbox_insert_pt_movement. The value can be NO_MOVEMENT, MOVE_RIGHT or MOVE_RIGHT. This function only calculates out the correct position of insertion point after movement. The insertion point must be reset in order to display it in the correct position.

29 TextboxInsertString

Purpose This function is called to insert a string into a textbox object where insert point is set and there is no highlight

Prototype Err TextboxInsertString(ObjectID textbox_id, BYTE* paste_string)

Scope Internal

Input Parameters textbox_id The ID value of the specified textbox
paste_string The paste string

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment Don’t redraw

30 TextboxInsertText

Purpose This function is called to insert a text to the specified textbox object. It is different from *TextboxInsert* because it is not inserting a string from the clipboard. This function can handle the passing of string when there is highlighted text or not.

Prototype
```
Err TextboxInsertText (ObjectID textbox_id,
                       BYTE *insert_string,
                       WORD textbox_len)
```

Scope Internal

Input Parameters

textbox_id	The ID value of the specified textbox
insert_string	Pointer to a string
textbox_len	the length of the string

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After the insertion of the string, the position of the insertion point will not be changed.

31 TextboxPasteString

Purpose This function is called to paste string into the textbox object

Prototype Err TextboxPasteString(ObjectID textbox_id, BYTE *paste_string)

Scope Application

Input Parameters

textbox_id	The ID value of the specified textbox
paste_string	the paste-in string

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

32 TextboxRestoreBackspaceChar

Purpose This function is called to get the character that is deleted by Backspacing to the original textbox object

Prototype Err TextboxRestoreBackspaceChar(ObjectID textbox_id,
UBYTE *character)

Scope Internal

Input Parameters textbox_id The ID value of the textbox object that is being backspaced

Output Parameters character The pointer to ASCII of the deleted character

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found
ERR_UI_OBJECT_NOT_MATCH
The object is not a textbox object

Comment None

33 TextboxSetAttribute

Purpose This function is called in order to set the attributes of a specified textbox object

Prototype Err TextboxSetAttribute(ObjectID textbox_id,
TextboxAttr input_textbox_attr)

Scope Application/Internal

Input Parameters textbox_id The ID value of the specified textbox
input_textbox_attr The TextboxAttr structure

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment Sometimes, application only wants to set a particular attribute of the textbox object. Therefore, application should get the attributes by using *TextboxGetAttribute*, then call *TextboxSetAttribute* to set the

attributes (only change the want that is required to change).

33 TextboxSetBounds

Purpose This function is called to set the bounds of a specified textbox object

Prototype Err TextboxSetBounds(ObjectID textbox_id, ObjectBounds bounds)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the specified textbox
bounds	Rectangular structure of the bounds of textbox object

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

34 TextboxSetDirty

Purpose This function is for setting the dirty attribute of the specified textbox object

Prototype Err TextboxSetDirty(ObjectID textbox_id,
BOOLEAN textbox_dirty)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of textbox object
textbox_dirty	The variable that stores the dirty attribute

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

35 TextboxSetFont

Purpose This function is called to set the type of font for a specified textbox object.

Prototype Err TextboxSetFont(ObjectID textbox_id, BYTE font)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of textbox object
font	The new type of font to be set

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment After the new font type is set, the textbox object must be drawn again in order to display the string on the screen with new font type.

36 TextboxSetHighlightSelection

Purpose This function is used to set the parameters of section of text that will be highlighed

Prototype Err TextboxSetHighlightSelection(ObjectID textbox_id,
WORD star_char_pos,
WORD end_char_pos)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the textbox object.
start_char_pos	Variable of character position of starting character of a highlight section
end_char_pos	Variable of character position of end character of a highlight section

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment None.

37 TextboxSetInsertPointOff

Purpose This function is called in order to turn off the insertion point of the specified textbox

Prototype Err TextboxSetInsertPointOff(ObjectID textbox_id)

Scope Application

Input Parameters textbox_id The ID value of the specified textbox

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND
The required object not found

Comment The attribute textbox_insert_pt_visible is cleared

38 TextboxSetInsertPointOn

Purpose This function is used to turn on the insertion point of the specified textbox object

Prototype Err TextboxSetInsertPointOn(ObjectID textbox_id)

Scope Application/Internal

Input Parameters textbox_id The ID value of the textbox object.

Output Parameters None

Return TRUE No Error
ERR_UI_RES_NOT_FOUND

	The required object not found

Comment The attribute textbox_insert_pt_visible is set to TRUE

39 TextboxSetInsertPointPositionByCharPos

Purpose This function is called for setting the insertion point position by inputting the new character position of the insertion point.

Prototype Err TextboxSetInsertPointPositionByCharPos(ObjectID textbox_id, WORD char_pos)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of textbox object
char_pos	The new character position of the insertion point

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment It will update the textbox_insert_pt_x and textbox_insert_py_y parameters correspondingly. The textbox must be redrawn in order to place the insertion point in the new position.

40 TextboxSetInsertPointPositionByXY

Purpose This function is called for setting the insertion point position by inputting the x and y coordinates of the insertion point.

Prototype Err TextboxSetInsertPointPositionByXY(ObjectID textbox_id, SHORT insert_xcoord, SHORT insert_ycoord)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of textbox object
insert_xcoord	The new x coordinate of insertion point
insert_ycoord	The new y coordinate of insertion

point

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment It will update the textbox_insert_pt_char_pos parameter correspondingly. If the input x and y-coordinates of the new interstion point is outside the bounds of the textbox object, then the nearest position within the textbox will be the new position of the insertion point.

41 TextboxSetLeftCharPos

Purpose To set the character position of the leftmost displayed character in the textbox in order to adjust the starting character on the display.

Prototype Err TextboxSetLeftCharPos(ObjectID textbox_id,
WORD left_char_pos)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the textbox object.
left_char_pos	Character position of the leftmost character in the textbox.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

42 TextboxSetMaxNumChars

Purpose This function is called to set the maximum number of characters in the textbox object.

Prototype Err TextboxSetMaxNumChars(ObjectID textbox_id,

WORD max_num_of_chars)

Scope Application/Internal

Input Parameters

textboxID	The ID value of the textbox object.
max_num_of_charas	The new maximum number of characters in the textbox.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

43 TextboxSetRightCharPos

Purpose To set the character position of the rightmost displayed character in the textbox.

Prototype Err TextboxSetRightCharPos(ObjectID textbox_id,
WORD right_char_pos)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the textbox object.
right_char_pos	Character position of the rightmost character in the textbox.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment None

44 TextboxSetText

Purpose This function is called to set the string of the textbox and place the insertion point after the last visible character.

Prototype Err TextboxSetText(ObjectID textbox_id, BYTE *string)

Scope Application/Internal

Input Parameters

textbox_id	The ID value of the textbox object.
textbox_text	Pointer to the text of the textbox.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_TEXTBOX_OVER_MAX_NUM_CHARS	The length of new string is longer than the maximum allowable number of characters

Comment None

45 TextboxStoreBackspaceChar

Purpose This function is called to store the character that is deleted by Backspacing

Prototype Err TextboxStoreBackspaceChar(ObjectID textbox_id, UBYTE character)

Scope Internal

Input Parameters

textbox_id	The ID value of the textbox object that is being backspaced
character	The ASCII of the deleted character

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found
ERR_UI_OBJECT_NOT_MATCH	The object is not a textbox object

Comment None

46 TextboxUndo

Purpose This function is called to undo the previous CUT or PASTE action.

Prototype Err TextboxUndo (ObjectID textbox_id)

Scope Application

Input Parameters

textbox_id	The ID value of the textbox object.

Output Parameters None

Return

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The required object not found

Comment In order to undo CUT or PASTE, there should be no other action between the CUT or PASTE and the undo action. The textbox object must be active and the insertion point must be on.

UI Global Functions

1 UIAddLinkListFormElement

Purpose It is a global function and it is called to add a Form object to a form Link List Table when it is first initialized. This function will also add the form object to the another lookup table for fast searching later. The Link List Table provides information about which form is the active one.

Prototype Err UIAddLinkListFormElement(ObjectID form_id, Form **form_ptr)

Scope Internal

Input Parameters

form_id	The ID value of the form object

Output Parameters

form_ptr	Pointer reference to form object

Result

TRUE	No Error
ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created

Comment This function will be called when the data structure of the data object is not loaded to RAM and it is embedded in the function *FormInitForm*.

2 UIAddressToLookupTable

Purpose It is a global function and it is called to add an UI object to a lookup table when it is first initialized. It will be more convenient later to get the pointer of a particular UI object by searching in the lookup table. If the UI object is not added to the lookup table, it is seen as non-existing.

Prototype

```
Err UIAddressToLookupTable(ObjectID object_id,
                           BYTE object_type,
                           void **object_ptr)
```

Scope Internal

Input Parameters

object_id	The ID value of the object
object_type	The type of the object

Output Parameters

object_ptr	Pointer reference to the object

Result

TRUE	No Error
ERR_UI_TYPE_MISMATCH	The type of the object in the Lookup table is different from the type of the current object ID
ERR_UI_CANT_CREATE_LOOKUP_TABLE	The object cannot be added to lookup table
ERR_UI_CANT_CREATE_POINTER	New pointer for the specified object cannot be created

Comment This function will be called when *FormDrawForm* is called

3 UIApplicationInit

Purpose It is a global function and it should be called before any other UI function is called

Prototype UIApplicationInit

Scope Application/Internal

Input Parameters None

Output Parameters None

Result None

Comment All UI global variables and pointers are initialised

4 UICheckObjectPopupStatus

Purpose This function is called by application to check whether menu or popup trigger object has been popup or not.

Prototype Err UICheckObjectPopupStatus(BOOLEAN *menu_popup_status,
BOOLEAN *popup_popup_status)

Scope Application

Input Parameters None

Output Parameters

menu_popup_status	pointer to a BOOLEAN value that shows whether menu is popup or not
popup_popup_status	pointer to a BOOLEAN value that shows whether popup trigger is popup or not

Result None

Comment None

5 UIColorConversion

Purpose This function is called to map the color settings of each object in resource file according to system settings

Prototype BYTE UIColorConversion(BYTE color_in)

Scope Internal

Input Parameters

color_in	a color that is chosen in resource file

Output Parameters None

Result A BYTE that states the mapped color is returned.

Comment None

6 UIDeleteAllAppObjects

Purpose This function is called in order to delete ALL the UI objects in the application

Prototype Err UIDeleteAllAppObjects()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return

TRUE	No error
ERR_UI_RES_NOT_FOUND	the specified UI Object not found

Comment None

7 UIDeleteLinkListFormElement

Purpose It is a global function and it is called to delete a form object entry from the Link List Table.

Prototype Err UIDeleteLinkListFormElement(ObjectID form_id)

Scope Internal

Input Parameters form_id The ID value of the specific form object

Output Parameters None

Result TRUE No Error
ERR_UI_NO_ACTIVE_FORM_FOUND
The specific form object cannot be found

Comment This function should be called within *FormDeleteForm*.

8 UIDeleteLookupTableElement

Purpose It is a global function and it is called to delete an UI object entry in the lookup table.

Prototype Err UIDeleteLookupTableElement(ObjectID form_id)

Scope Internal

Input Parameters form_id The ID value of the specific form object

Output Parameters None

Result TRUE No Error
ERR_UI_RES_NOT_FOUND
The UI object not found

Comment This function should be called within the functions that destroy the UI object.

9 UIGetUndoStatus

Purpose This function is called to get the status of UNDO in the system

Prototype BYTE UIGetUndoStatus()

Scope Application/Internal

Input Parameters None

Output Parameters None

Return UNDO_EMPTY
UNDO_CUT

UNDO_PASTE
UNDO_BACKSPACE

Comment None

10 UIInit

Purpose It is a global function and it should be included in the function main() to initialise the UI system before it can operate properly

Prototype UIInit()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment All UI global variables and pointers are initialised

11 UISearchForAddress

Purpose It is a global function and it is called to find out the pointer address to the memory of the data structure of a specific UI object.

Prototype
```
Err UISearchForAddress(ObjectID object_id,
                       BYTE *object_type,
                       void ** object_ptr)
```

Scope Internal

Input Parameters

object_id	The ID value of the specific object

Output Parameters

object_type	Pointer to variable to show the type of the object
object_ptr	Pointer reference of object

Result

TRUE	No Error
ERR_UI_RES_NOT_FOUND	The UI object not found

Comment This function can be called whenever the pointer to an object is required