UI Events

1) UI Events List

UI Object	*UI Object Events*
Application	➢ EVT_APP_STOP ➢ EVT_APP_LAUNCH
Form	➢ EVT_FORM_OPEN ➢ EVT_FORM_LOAD
Control	➢ EVT_CONTROL_ENTER ➢ EVT_CONTROL_EXIT ➢ EVT_CONTROL_POPUP_SELECT ➢ EVT_CONTROL_REPEAT ➢ EVT_CONTROL_SELECT
Field	➢ EVT_FIELD_CHANGED ➢ EVT_FIELD_ENTER ➢ EVT_FIELD_MODIFIED ➢ EVT_FIELD_SELECT ➢ EVT_FIELD_JOT_PASTE_STRING
List	➢ EVT_LIST_ARROW ➢ EVT_LIST_ENTER ➢ EVT_LIST_EXIT ➢ EVT_LIST_SELECT
Menu	➢ EVT_MENU_ENTER ➢ EVT_MENU_EXIT ➢ EVT_MENU_SELECT ➢ EVT_MENU_SELECT_ITEM
Scrollbar	➢ EVT_SCROLLBAR_ARROW_DELAY ➢ EVT_SCROLLBAR_ENTER ➢ EVT_SCROLLBAR_ENTER_REPEAT ➢ EVT_SCROLLBAR_EXIT ➢ EVT_SCROLLBAR_REPEAT ➢ EVT_SCROLLBAR_SELECT
Table	➢ EVT_TABLE_ENTER ➢ EVT_TABLE_SELECT ➢ EVT_TABLE_EXIT
Schedule Line	➢ EVT_SCHLINE_ENTER ➢ EVT_SCHLINE_EXIT ➢ EVT_SCHLINE_SELECT
Bitmap	➢ EVT_BITMAP_ENTER ➢ EVT_BITMAP_SELECT

	➢ EVT_BITMAP_EXIT
Textbox	➢ EVT_TEXTBOX_CHANGED ➢ EVT_TEXTBOX_ENTER ➢ EVT_TEXTBOX_JOT_PASTE_STRING ➢ EVT_TEXTBOX_MODIFIED ➢ EVT_TEXTBOX_SELECT
Keyboard	➢ EVT_KEY ➢ EVT_KEYBOARD_ENTER ➢ EVT_KEYBOARD_EXIT ➢ EVT_KEYBOARD_STATUS
Inlay	➢ EVT_INLAY_ENTER ➢ EVT_INLAY_EXIT ➢ EVT_INLAY_SELECT

2) Descriptions of Events

2a) Events of Application Start and Application Stop

EVT_APP_STOP

Sent by: Application / System

Handled by: Application

Description: This event is used to request an application to stop. When the active running application receives this event, the application event loop is stopped and the application should save all required information for next application restore.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_APP_STOP
eventID	=	NULL
para1	=	NULL
para2	=	NULL
evtPBP	=	NULL

EVT_APP_LAUNCH

Sent by: Application / System

Handled by: System

Description: This event is used to request the system to launch a particular application. This event should be sent after the EVT_APP_STOP is sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_APP_LAUNCH
eventID	=	the application ID
para1	=	launch command
para2	=	NULL
evtPBP	=	launch command pointer. It should be NULL if launch command pointer is not required.

2a) Events of Form

EVT_FORM_OPEN

Sent by: Form API functions *FormPopupForm*

Handled by: Application

Description: This event is the request to ask the application to initialize all the form objects and draw a form

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FORM_OPEN
eventID	=	the object ID of the form that is going to be

		opened
para1	=	NULL
para2	=	NULL
evtPBP	=	NULL

EVT_FORM_LOAD

Sent by: Form API functions *FormPopupForm*

Handled by: Application

Description: There are 2 purposes:
- To load the form object into memory if required.
- to request and switch the active form to the loaded form

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FORM_LOAD
eventID	=	the object ID of the form that is going to be loaded
para1	=	NULL
para2	=	NULL
evtPBP	=	NULL

2c) Events of Control

EVT_CONTROL_ENTER

Sent by: *ControlHandleEvent*

Handled by: *ControlHandleEvent*

Description: This event indicates and shows that a pen-down event is received and the pen touches the screen within the bounds of a control object. When the *ControlHandleEvent* receives this event, corresponding procedure is proceeded. For example, the control object being entered will be inverted in color.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_CONTROL_ENTER
eventID	=	the object ID of the control object that the pen entered its bounds
para1	=	NULL
para2	=	the style of the control object 0 = BUTTON 1 = PUSH_BUTTON 2 = REPEAT_BUTTON 3 = CHECKBOX 4 = POPUP_TRIGGER
evtPBP	=	pointer to the memory that stored the data of the control object

EVT_CONTROL_EXIT

Sent by: *ControlHandleEvent*

Handled by: *ControlHandleEvent*

Description: This event indicates that a pen-move event is received and the pen is moved outside the bounds of the object that is entered before. When the *ControlHandleEvent* receives this event, corresponding procedure is proceeded. For example, the control object that the pen is exit will be inverted back to original color.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_CONTROL_EXIT
eventID	=	the object ID of the control object that the pen just moved out
para1	=	NULL
para2	=	the style of the control object
evtPBP	=	pointer to the memory of the control object

EVT_CONTROL_POPUP_SELECT

Sent by: *ControlHandleEvent*

Handled by: Application

Description: This event indicates that pen-up event is received and the pen is lifted within the bounds of one of the selection of the popup window of the popup trigger control object that the pen entered. The event will be sent back to application layer for further processing.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_CONTROL_POPUP_SELECT
eventID	=	the object ID of the selected control object
para1	=	the item_number of selected item
para2	=	NULL
evtPBP	=	pointer to the memory of the control object

EVT_CONTROL_REPEAT

Sent by: *ControlHandleEvent*

Handled by: Application

Description: This event indicates that an action of a REPEAT_BUTTON control object is repeated once. The event will be sent back to application layer for further processing.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_CONTROL_REPEAT
eventID	=	the object ID of the repeating REPEAT_BUTTON control object
para1	=	NULL
para2	=	REPEAT_BUTTON
evtPBP	=	pointer to the memory of the control object

EVT_CONTROL_SELECT

Sent by: *ControlHandleEvent*

Handled by: Application *or ControlHandleEvent*

Description: This event indicates that pen-up event is received and the pen is lifted within the bounds of the control object that the pen entered. The event will be sent back to application layer for further processing or the ControlHandleEvent will absorb this event if the control style is POPUP_TRIGGER because the popup window will be shown for further selection of options.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_CONTROL_SELECT
eventID	=	the object ID of the selected control object
para1	=	NULL
para2	=	style of the control object
evtPBP	=	pointer to the memory of the control object

2d) Events of Field

EVT_FIELD_CHANGED

Sent by: *FieldHandleEvent*

Handled by: Application

Description: This event is sent to application in order to notify the application about the change of the content and the change of the displaying screen of the corresponding field object. Therefore, if the text in the field object is changed or the text in the field object is scrolled up or down, this event is sent. After the reception of the event, application can have enough information to set the parameters of corresponding scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FIELD_CHANGED
eventID	=	the object ID of the FIELD object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the field object

EVT_FIELD_ENTER

Sent by: *FieldHandleEvent*

Handled by: *FieldHandleEvent*

Description: This event indicates that a pen down event is received within the bounds of the field object. After the EVT_FIELD_ENTER event is received by the *FieldHandleEvent*, insert point is disabled and is ready for highlighting the text in the field object.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FIELD_ENTER
eventID	=	the object ID of the FIELD object
para1	=	x-coordinate of the current position of the pen when it enters the bounds of the field object
para2	=	y-coordinate of the current position of the pen when it enters the bounds of the field object
evtPBP	=	pointer to the memory of the field object

EVT_FIELD_JOT_PASTE_STRING

Sent by: *Jot Recognition System*

Handled by: *FormHandleEvent*

Description: This event is to paste jot symbol into a field object. The insert point and highlighted region of the field object are updated on the screen and the field object is redrawn.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FIELD_JOT_PASTE_STRING
eventID	=	the active application ID
para1	=	NULL
para2	=	NULL
evtPBP	=	the text to be pasted

EVT_FIELD_MODIFIED

Sent by: *FieldHandleEvent*

Handled by: Application

Description: This event is sent to application in order to notify the application about the change of the content of the corresponding field object. Therefore, if the text in the field object is changed, this event is sent. After the reception of the event, application can have enough information to set the parameters of corresponding scrollbar. This event can be used to check whether the content is modified or not.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FIELD_MODIFIED
eventID	=	the object ID of the FIELD object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the field object

EVT_FIELD_SELECT

Sent by: *FieldHandleEvent*

Handled by: Application

Description: This event indicates that a pen up event is received while a pen down event for that field object was already received. This event gives a chance the application to proceed a section of codes under the selected condition. The insert point is shown if there is no highlight of the text.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_FIELD_SELECT
eventID	=	the object ID of the FIELD object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the field object

2e) Events of List

EVT_LIST_ARROW

Sent by: *ListHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the scroll-up or scroll-down button on the list object. Application can have a chance to do something after the reception of the notification of the selection of the scroll-up or scroll-down button.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_LIST_ARROW
eventID	=	the object ID of the LIST object
para1	=	REGION_UP_ARROW if the scroll-up button is selected REGION_DN_ARROW if the scroll-down button is selected
para2	=	NULL
evtPBP	=	pointer to the memory of the list object

EVT_LIST_ENTER

Sent by: *ListHandleEvent*

Handled by: *ListHandleEvent*

Description: This event is sent when the *ListHandleEvent* receives a pen-down event within the bounds of a list object.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_LIST_ENTER
eventID	=	the object ID of the LIST object
para1	=	REGION_ITEMS if the entered region is the region of one of the selections in the list object REGION_UP_ARROW if the entered region is the region of the scroll-up arrow in the list object REGION_DN_ARROW if the entered region is

		the region of the scroll-down arrow in the list object
para2	=	NULL
evtPBP	=	pointer to the memory of the list object

EVT_LIST_EXIT

Sent by: *ListHandleEvent*

Handled by: *ListHandleEvent*

Description: *ListHandleEvent* keeps track of the action of the pen after the EVT_LIST_ENTER event is received. If the pen is moved out of the bounds of the field object. EVT_LIST_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_LIST_EXIT
eventID	=	the object ID of the LIST object
para1	=	REGION_ITEMS if the region being moved out is the region of one of the selections in the list object REGION_UP_ARROW if the region being moved out is the region of the scroll-up arrow in the list object REGION_DN_ARROW if the region being moved out is the region of the scroll-down arrow in the list object
para2	=	NULL
evtPBP	=	pointer to the memory of the list object

EVT_LIST_SELECT

Sent by: *ListHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the field object that is entered in the last pen down section.

Data passed by The following shows the required data that is passed with the event

event:

eventType	=	EVT_LIST_SELECT
eventID	=	the object ID of the LIST object
para1	=	the item number of the new selection of the list
para2	=	NULL
evtPBP	=	pointer to the memory of the list object

2f) Events of Menu

EVT_MENU_ENTER

Sent by: *MenuHandleEvent*

Handled by: *MenuHandleEvent*

Description: This event is sent when the *MenuHandleEvent* receives a pen-down event within the bounds of one of the selection in the menu popup window.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_MENU_ENTER
eventID	=	the object ID of the MENU object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the menu object

EVT_MENU_EXIT

Sent by: *MenuHandleEvent*

Handled by: *MenuHandleEvent*

Description: *MenuHandleEvent* keeps track of the action of the pen after the EVT_MENU_ENTER event is received. If the pen is moved out of the bounds of the selection in the menu popup window. EVT_MENU_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_MENU_ENTER
eventID	=	the object ID of the MENU object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the menu object

EVT_MENU_SELECT

Sent by: *InlayHandleEvent*

Handled by: *MenuHandleEvent*

Description: This event is sent when the menu button is clicked. After *MenuHandleEvent* receives this event, the menu popup window will be initiated.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_MENU_SELECT
eventID	=	the object ID of the MENU object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the menu object

EVT_MENU_SELECT_ITEM

Sent by: *MenuHandleEvent*

Handled by: Application

Description: This event is sent back to application to notify it about the selection of menu item. Then the application can have a chance to proceed the corresponding procedures.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_MENU_ENTER
eventID	=	the object ID of the MENU object
para1	=	the item number of the menu selection
para2	=	NULL
evtPBP	=	pointer to the memory of the menu object

2g) Events of Scrollbar

EVT_SCROLLBAR_ARROW_DELAY

Sent by: *ScrollbarHandleEvent*

Handled by: *ScrollbarHandleEvent*

Description: This event is used to introduce an intermediate state to generate a delay between the first EVT_SCROLLBAR_REPEAT and the second EVT_SCROLLBAR_REPEAT. This event is only sent when the first click is on the arrow region of the scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLBLAR_ARROW_DELAY
eventID	=	the object ID of the SCROLLBAR object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the scrollbar object

EVT_SCROLLBAR_ENTER

Sent by: *ScrollbarHandleEvent*

Handled by: *ScrollbarHandleEvent*

Description: This event is sent when the *ScrollbarHandleEvent* receives a pen-down event within the bounds of the scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLBLAR_ENTER
eventID	=	the object ID of the SCROLLBAR object
para1	=	current x-coordinate of the pen on the touch panel
para2	=	current y-coordinate of the pen on the touch panel
evtPBP	=	pointer to the memory of the scrollbar object

EVT_SCROLLBAR_ENTER_REPEAT

Sent by: *ScrollbarHandleEvent*

Handled by: *ScrollbarHandleEvent*

Description: This event is used to introduce an intermediate state to generate a delay between the first EVT_SCROLLBAR_REPEAT and the second EVT_SCROLLBAR_REPEAT. This event is only sent when the first click is on the upper region or the lower region of the scroll car.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLBLAR_ENTER_DELAY
eventID	=	the object ID of the SCROLLBAR object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the scrollbar object

EVT_SCROLLBAR_EXIT

Sent by: *ScrollbarHandleEvent*

Handled by: *ScrollbarHandleEvent*

Description: *ScrollbarHandleEvent* keeps track of the action of the pen after the EVT_SCROLLBAR_ENTER event is received. If the pen is moved out of the bounds of the scrollbar, EVT_SCROLLBAR_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLLBAR_ENTER
eventID	=	the object ID of the SCROLLBAR object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the scrollbar object

EVT_SCROLLBAR_REPEAT

Sent by: *ScrollbarHandleEvent*

Handled by: Application

Description: This event is sent when the pen is continually held within the bounds of a scrollbar. Application should watch for this event if dynamic scrolling is required. Dynamic scrolling means that the display is updated as the user click on the scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLLBAR_REPEAT
eventID	=	the object ID of the SCROLLBAR object
para1	=	previous value of the scrollbar
para2	=	new value of the scrollbar
evtPBP	=	pointer to the memory of the scrollbar object

EVT_SCROLLBAR_SELECT

Sent by: *ScrollbarHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the scrollbar object that is entered in the last pen down section. Application can change the display according to the current value of the scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCROLLBAR_SELECT
eventID	=	the object ID of the SCROLLBAR object
para1	=	current value of the scrollbar
para2	=	current value of the scrollbar
evtPBP	=	pointer to the memory of the SCROLLBAR object

2h) Events of Table

EVT_TABLE_ENTER

Sent by: *TableHandleEvent*

Handled by: *TableHandleEvent*

Description: This event is sent when the *TableHandleEvent* receives a pen-down event within the bounds of one of the cell of the table object.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TABLE_SELECT
eventID	=	the object ID of the TABLE object
para1	=	row number of the cell that absorbs the pen down event
para2	=	column number of the cell that absorbs the pen down event
evtPBP	=	pointer to the memory of the table object

EVT_TABLE_EXIT

Sent by: *TableHandleEvent*

Handled by: *TableHandleEvent*

Description: *TableHandleEvent* keeps track of the action of the pen after the EVT_TABLE_ENTER event is received. If the pen is moved out of the bounds of the entered cell, EVT_TABLE_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TABLE_EXIT
eventID	=	the object ID of the SCROLLBAR object
para1	=	row number of the entered cell
para2	=	column number of the entered cell
evtPBP	=	pointer to the memory of the table object

EVT_TABLE_SELECT

Sent by: *TableHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the cell that is entered in the last pen down section.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TABLE_ENTER
eventID	=	the object ID of the TABLE object
para1	=	row number of the selected cell
para2	=	column number of the selected cell
evtPBP	=	pointer to the memory of the table object

2i) Events of Schedule Line

EVT_SCHLINE_ENTER

Sent by: *SchlineHandleEvent*

Handled by: *SchlineHandleEvent*

Description: This event is sent when the *SchlineHandleEvent* receives a pen-down event within the bounds of one of the section in the object

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCHLINE_ENTER
eventID	=	the object ID of the SCHLINE object
para1	=	the section number of the entered region
para2	=	NULL
evtPBP	=	pointer to the memory of the schedule line object

EVT_SCHLINE_EXIT

Sent by: *SchlineHandleEvent*

Handled by: *SchlineHandleEvent*

Description: *SchlineHandleEvent* keeps track of the action of the pen after the EVT_SCHLINE_ENTER event is received. If the pen is moved out of the bounds of entered section, EVT_SCHLINE_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCHLINE_EXIT
eventID	=	the object ID of the SCHLINE object
para1	=	the section number of the entered region
para2	=	NULL
evtPBP	=	pointer to the memory of the schedule line object

EVT_SCHLINE_SELECT

Sent by: *SchlineHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the section that is entered in the last pen down section.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_SCHLINE_SELECT
eventID	=	the object ID of the SCHLINE object
para1	=	the section number of the selected region
para2	=	NULL
evtPBP	=	pointer to the memory of the schedule line object

2j) Events of Bitmap

EVT_BITMAP_ENTER

Sent by: *BitmapHandleEvent*

Handled by: *BitmapHandleEvent*

Description: This event is sent when the *BitmapHandleEvent* receives a pen-down event within the bounds of the bitmap object.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_BITMAP_SELECT
eventID	=	the object ID of the BITMAP object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the bitmap object

EVT_BITMAP_EXIT

Sent by: *BitmapHandleEvent*

Handled by: *BitmapHandleEvent*

Description: *BitmapHandleEvent* keeps track of the action of the pen after the EVT_BITMAP_ENTER event is received. If the pen is moved out of the bounds of entered section, EVT_BITMAP_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_BITMAP_SELECT
eventID	=	the object ID of the BITMAP object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the bitmap object

EVT_BITMAP_SELECT

Sent by: *BitmapHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the bitmap object that is entered in the last pen down section.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_BITMAP_SELECT
eventID	=	the object ID of the BITMAP object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the bitmap object

2k) Events of Textbox

EVT_TEXTBOX_CHANGED

Sent by: *TextboxHandleEvent*

Handled by: Application

Description: This event is sent to application in order to notify the application about the change of the content and the change of the displaying screen of the corresponding textbox object. Therefore, if the text in the textbox object is changed or the text in the textbox object is scrolled left or right, this event is sent. After the reception of the event, application can have enough information to set the parameters of corresponding scrollbar.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TEXTBOX_CHANGED
eventID	=	the object ID of the TEXTBOX object

para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the textbox object

EVT_TEXTBOX_ENTER

Sent by: *TextboxHandleEvent*

Handled by: *TextboxHandleEvent*

Description: This event indicates that a pen down event is received within the bounds of the textbox object. After the EVT_TEXTBOX_ENTER event is received by the *TextboxHandleEvent*, insert point will be disabled and is ready for highlighting.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TEXTBOX_ENTER
eventID	=	the object ID of the TEXTBOX object
para1	=	the x-coordinate of the current position of the pen when it enters the bounds of the textbox object
para2	=	the y-coordinate of the current position of the pen when it enters the bounds of the textbox object
evtPBP	=	pointer to the memory of the textbox object

EVT_TEXTBOX_JOT_PASTE_STRING

Sent by: *Jot Recognition System*

Handled by: *FormHandleEvent*

Description: This event is to paste jot symbol into a textbox object. The insert point and highlighted region of the textbox object are updated on the screen and the textbox object is redrawn.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TEXTBOX_JOT_PASTE_STRING
eventID	=	the active application ID
para1	=	NULL

para2	=	NULL
evtPBP	=	the text to be pasted

EVT_TEXTBOX_MODIFIED

Sent by: *TextboxHandleEvent*

Handled by: Application

Description: This event is sent to application in order to notify the application about the change of the content of the corresponding textbox object. Therefore, if the text in the textbox object is changed, this event is sent. After the reception of the event, application can have enough information to set the parameters of corresponding scrollbar. This event can be used to check whether the content is modified or not.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TEXTBOX_MODIFIED
eventID	=	the object ID of the TEXTBOX object
para1	=	NULL
para2	=	NULL
evtPBP	=	pointer to the memory of the textbox object

EVT_TEXTBOX_SELECT

Sent by: *TextboxHandleEvent*

Handled by: Application

Description: This event indicates that a pen up event is received while a pen down event for that textbox object was already received. This event gives a chance the application to proceed a section of codes under the selected condition.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_TEXTBOX_SELECT
eventID	=	the object ID of the TEXTBOX object
para1	=	NULL
para2	=	NULL

evtPBP	=	pointer to the memory of the textbox object

2l) Events of Keyboard

EVT_KEY

Sent by: *KeyboardHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the entered key on the software keyboard that is entered in the last pen down section.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_KEY
eventID	=	SOFT_KEY
para1	=	NULL if the ASCII code of the clicked key is not between 32 to 255. Otherwise, the correct ASCII code of the clicked key is put in para1
para2	=	the lowest 16 bits of para2 are used the lowest 8 bits are used if the ASCII code the clicked key is smaller than 32. Otherwise, it will be set to NULL for the next lowest 8 bits, bit 0 = CTRL bit 1 = ALT bit 2 = SHIFT bit 3 = CAP bit 4 = International if any one of those keys is set, the corresponding bit(s) will be set corresponding too
evtPBP	=	NULL

EVT_KEYBOARD_ENTER

Sent by: *KeyboardHandleEvent*

Handled by: *KeybardHandleEvent*

Description: This event is sent when the *KeyboardHandleEvent* receives a pen-down event within the bounds of one of the key on the software keyboard.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_KEYBOARD_ENTER
eventID	=	NULL
para1	=	mapped ASCII code of the entered key on the keyboard
para2	=	NULL
evtPBP	=	NULL

EVT_KEYBOARD_EXIT

Sent by: *KeyboardHandleEvent*

Handled by: *KeyboardHandleEvent*

Description: *KeyboardHandleEvent* keeps track of the action of the pen after the EVT_KEYBOARD_ENTER event is received. If the pen is moved out of the bounds of entered key button on the software keyboard, EVT_KEYBOARD_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_KEYBOARD_EXIT
eventID	=	NULL
para1	=	mapped ASCII code of the previous entered key on the keyboard
para2	=	NULL
evtPBP	=	NULL

EVT_KEYBOARD_STATUS

Sent by: *HardwareButtonHandleEvent*

Handled by: *Application*

Description: *HardwareButtonHandleEvent* sends this event whenever the software keyboard is pop up or closed. The application should do the redrawing of the form on the screen whenever the keyboard is popup or closed. Therefore, this is a notification event.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_KEYBOARD_STATUS	
eventID	=	NULL	
para1	=	KEYBOARD_ON	if the keyboard is now on
		KEYBOARD_OFF	if the keyboard is now off
para2	=	NULL	
evtPBP	=	NULL	

2m) Events of Inlay

EVT_INLAY_ENTER

Sent by: *InlayHandleEvent*

Handled by: *InlayHandleEvent*

Description: This event is sent when the *InlayHandleEvent* receives a pen-down event within the bounds of one of the inlay button.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_INLAY_ENTER
eventID	=	0
para1	=	INLAY_KEYBOARD or INLAY_OK or INLAY_EXIT or INLAY_MENU or INLAY_CALCUALTOR or INLAY_MAIN_MENU or INLAY_LEFT or INLAY_RIGHT
para2	=	0
evtPBP	=	NULL

EVT_INLAY_EXIT

Sent by: *InlayHandleEvent*

Handled by: *InlayHandleEvent*

Description: *InlayHandleEvent* keeps track of the action of the pen after the EVT_INLAY_ENTER event is received. If the pen is moved out of the bounds of entered section, EVT_INLAY_EXIT will be sent.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_INLAY_EXIT
eventID	=	0
para1	=	INLAY_KEYBOARD or INLAY_OK or INLAY_EXIT or INLAY_MENU or INLAY_CALCUALTOR or INLAY_MAIN_MENU or INLAY_LEFT or INLAY_RIGHT
para2	=	0
evtPBP	=	NULL

EVT_INLAY_SELECT

Sent by: *InlayHandleEvent*

Handled by: Application

Description: This event is sent out if the pen is lifted within the bounds of the section that is entered in the last pen down section. By checking this event, the application can know what inlay button is pressed.

Data passed by event: The following shows the required data that is passed with the event

eventType	=	EVT_INLAY_SELECT
eventID	=	0
para1	=	INLAY_KEYBOARD or INLAY_OK or

		INLAY_EXIT or INLAY_MENU or INLAY_CALCUALTOR or INLAY_MAIN_MENU or INLAY_LEFT or INLAY_RIGHT
para2	=	0
evtPBP	=	NULL