Documentation for OEMs:
CoDeSys V2.3
Target-Visualization, Interface Description

Document Version 1.0

CONTENT

1 Functionality of the Target-Visualization

The aim of the code generator is to provide a complete visualisation with as simple commands as possible. Of central importance in this connection are the individual elements of the visualisation for each of which ST-code is generated separately. These elements are thus displayed correctly without hardly any general pre-settings (e.g. standard font or the like). Which in turn is made possible above all by references and groups.

1.1 Current requirements for a Target Visualisation:

The task configuration has to contain a VISU_TASK. An additional VISU_INPUT _TASK allows to execute the code for all inputs in a separate task which can be configured with a lower cycle time. The visualisation with the name PLC_VISU is used as start visualisation. If no such visualisation is available the first visualisation in the alphabetically sorted tree of visualization objects will be used.

2 Functional specification of the external library SysLibTargetVisu.lib

The external library SysLibTargetVisu.lib consists of a few functions witch are different for the different target systems and thus have to be implemented individually for each runtime environment. In this concern it is also important to regard that some functions have to be adapted to the Windows style in order to achieve an identical display in the target system. These functions subsequently will be provided with a Note each.

2.1 FUNCTION BeginPaint : BOOL

```
VAR_INPUT
      Name: STRING;           (Visualization name)
      dwFlags:DWORD;          (Associated task)
      nBmpId:INT;             (Bitmap ID of the background bitmap)
      dwBgColor:DWORD;        (Background color)
END_VAR
```

Begin Paint opens a drawing cycle. In this function the DoubleBuffering should be implemented in order to prevent possible flickering.

The flag DwFlags indicates the currently active task. For performance reasons the functions for user inputs have been separated from the functions for pure display. INPUT_TASK processes all „IsClickedXXX“ functions.

```
#define BOTH_TASKS 1
#define INPUT_TASK  2
#define VISU_TASK   4
```

2.2 FUNCTION EndPaint : BOOL

```
VAR_INPUT
      Name: STRING;           (Visualization name)
      dwFlags:DWORD;          (Associated task)
END_VAR
```

EndPaint issues the altered bitmap.

The flag DwFlags indicates the currently active task. For performance reasons the functions for user inputs have been separated from the functions for pure display. INPUT_TASK processes all alle „IsClickedXXX“ functions.

```
#define BOTH_TASKS 1
#define INPUT_TASK  2
#define VISU_TASK   4
```

2.3 FUNCTION DrawRect : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      stText:STRING;          (text within the rectangle)
```

```
      dwDrawFlags:DWORD;      (text position)
      nType:INT;              (see below)
      bVisible:BOOL;          (visibility of the rectangle)
END_VAR
```

The function need four coordinates (x + y of the left upper corner and x + y of the right lower corner of a square) to draw a figure which can contain text. This text can be adjusted in relation to the four coordinates with the following text flags:

DF_LEFT ,

DF_RIGHT,

DF_HCENTER,

DF_TOP,

DF_BOTTOM and

DF_VCENTER

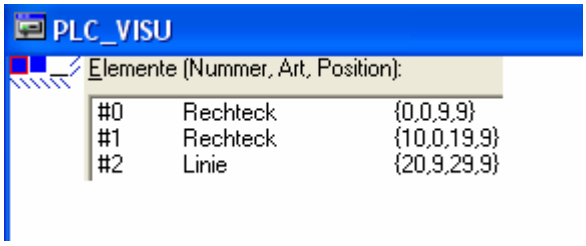
The shapes should be drawn according to the specifications made in SetFill and SetLine (is executed before each DrawRect). The font and font size is sent by SetFont.

These shapes can communicate with the user via EditControls (DrawEditC) or as a button (IsClickedRect).

The parameter nType could be one of the following

```
#define VSS_RECTANGLE     0
#define VSS_ROUNDRECT     1
#define VSS_CIRCLE        2
#define VSS_SIMPLELINE    3
#define VSS_SELECTION     4
#define VSS_TABLE         5
```

Note: In the following diagram it is shown, how the Windows Api function Rectangle uses the coordinates. For example a rectangle with a frame will be drawn smaller than defined by one pixel in X- and Y-direction.



If type VSS_SIMPLELINE is used, please regard, that normalized rectangles (upper left in this case really is upper left) will draw the line from lower left to upper right. Non-nomalized rectangles however will be drawn from upper left to lower right.

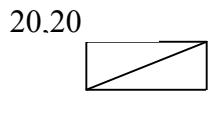


Fig.: 2.3.1

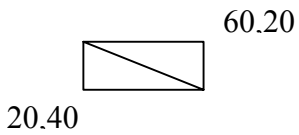


Fig.: 2.3.2

2.4 FUNCTION DrawPolygon : BOOL

```
VAR_INPUT
      nNumberOfPoints:INT;
      pnXPoints:POINTER TO INT;
      pnYPoints:POINTER TO INT;
      stText:STRING;
      dwDrawFlags:DWORD;
      nType:INT;
      bVisible:BOOL;
END_VAR
```

Draws a polygon with a maximum of 512 corners. This function is similar to the functions DrawRect.

The parameter nType could be one of the following

```
#define VPS_POLYGON       0
#define VPS_POLYLINE      1
#define VPS_POLYBEZIERS   2
```

2.5 FUNCTION DrawText : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      stText:STRING;          (Text)
      dwDrawFlags:DWORD;      (Text position)
      bVisible:BOOL;          (Visibility of the text)
END_VAR
```

Writes text into a square. The text flags are interpreted as described under DrawRect.

2.6 FUNCTION DrawBitmap : BOOL

```
VAR_INPUT
nId:INT;
nXCorner:INT;
nYCorner:INT;
mXCorner:INT;
mYCorner:INT;
nIso:INT;
dwColorTransparent:DWORD;
bVisible:BOOL;
END_VAR
```

Draws the bitmap with the index ID. A square is defined as frame. The parameter nIso defines whether the visualisation is distorted or shown in relation to the original size.

```
#define ANISOTROPISCH      0
#define ISOTROPISCH        1
#define  ORIGINALSIZE      2
```

2.7 FUNCTION DrawButton : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      stText:STRING;
      dwDrawFlags:DWORD;
      bIsClicked:BOOL;
      dwFlags:DWORD;
      bVisible:BOOL;
END_VAR
```

Draws a button which appears either normal or activated (dependent on bIsClicked).

2.8 FUNCTION DrawPie : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      nAngleStart:INT;
      nAngleEnd:INT;
      nAngle:DINT;
      dwPieFlags:DWORD;
      stText:STRING;
      dwDrawFlags:DWORD;
      bVisible:BOOL;
END_VAR
```

Draws a pie dependent on `nAngleStart, nAngleEnd and nAngle.`

The parameter `dwPieFlags` could be `DRAWCIRCLEONLY, where only the line of the pie would be painted.`

```
#define DRAWCIRCLEONLY 1
```

2.9 FUNCTION DrawEditC : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      stText:STRING;
      stExpression:STRING;
      nIsString:INT;
      nType:INT;
      stMin:STRING;
      stMax:STRING;
      stComment:STRING;
END_VAR
```

Opens a simple edit window (EditControl) which can only be closed by a click (which should normally show no effect) or by pressing the ESC or the return key. If the return key is used the correctness of the entered value should be checked using the variable IsString. This variable returns one of the following values:

```
#define TYPE_STRING       1
#define TYPE_REAL         2
#define TYPE_DINT         3
#define TYPE_INT          4
#define TYPE_WORD         5
#define TYPE_DWORD        6
#define TYPE_SINT         7
#define TYPE_USINT        8
#define TYPE_UINT         9
#define TYPE_UDINT        10
#define TYPE_LREAL        11
#define TYPE_BOOL         12
#define TYPE_BYTE         13
#define TYPE_TIME         14
#define TYPE_ARRAY        15
#define TYPE_ENUM         16
#define TYPE_USERDEF      17
#define TYPE_BITORBYTE    18
#define TYPE_POINTER      19
#define TYPE_DATE         20
#define TYPE_TOD          21
#define TYPE_DT           22
#define TYPE_VOID         23
#define TYPE_REF          24
```

If the value is not correct the EditControl should remain open.

The variable “Expression” is important for the Funktion IsClosedEditC. The variable should be saved so that the correct variable is not overwritten.

2.10 FUNCTION ReturnEnteredValue :STRING

```
VAR_INPUT
      stExpression:STRING;
END_VAR
```

Returns the value in EditControls after closing. Is only executed if the following function returns TRUE.

2.11 FUNCTION IsClosedEditC : BOOL

```
VAR_INPUT
      stExpression:STRING;
END_VAR
```

The function returns TRUE as soon as an EditControl which changes the variable stExpression is closed with the return key (i.e. with a correct value).

2.12 FUNCTION CreateBitmap : INT

```
VAR_INPUT
      SIZE:DINT;
      PBITMAP:POINTER TO DWORD;
END_VAR
```

SIZE: The size of the bitmap file.

PBITMAP: pointer to an array

CreateBitmap creates a copy and returns an integer which defines the bitmap (An array of pointers to bitmaps is created).

The bitmap is only created so that later on it can be visualised with DrawBitmap.

2.13 FUNCTION DeleteBitmap : BOOL

```
VAR_INPUT
      ID:INT;
END_VAR
```

Deletes a bitmap with the index ID.

2.14 FUNCTION SendBitmap : INT

```
VAR_INPUT
      stFile:STRING;
END_VAR
```

SendBitmap creates a handle to a bitmap and returns the id in the storage buffer. The parameter `stFile` describes the name of the bitmap file.

2.15 FUNCTION IsClickedPolygon : BOOL

```
VAR_INPUT
      nNumberOfPoints:INT;
      pnXPoints:POINTER TO INT;
      pnYPoints:POINTER TO INT;
      bUp:BOOL;               (MOUSE_UP or MOUSE_DOWN event)
      bToggle:BOOL;           (Toggle or Tap variable)
      nType:INT;
      bVisible:BOOL;
END_VAR
```

This function defines an area which is to react upon mouse click (either click down or click up) (Set by the variable nUp: TRUE reacts on click up FALSE reacts on click down). Whether the mouse movements are to be followed or not (after the click down in the target area) and whether click down is saved (Toggle = FALSE) or not (Toggle = TRUE) is determined by toggling.

nType specifies one of the following types:

```
#define VPS_POLYGON       0
#define VPS_POLYLINE      1
#define VPS_POLYBEZIERS   2
```

2.16 FUNCTION IsClickedRect: SINT

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      bUp:BOOL;               (MOUSE_UP or MOUSE_DOWN event)
      bToggle:BOOL;           (Toggle or Tap variable)
      nType:INT;
      bVisible:BOOL;
END_VAR
```

This function defines an area which is to react upon mouse click (either click down or click up) (Set by the variable nUp: TRUE reacts on click up FALSE reacts on click down). Whether the mouse movements are to be followed or not (after the click down in the target area) and whether click down is saved (Toggle = FALSE) or not (Toggle = TRUE) is determined by toggling.

nType specifies one of the following types:

```
#define VSS_RECTANGLE     0
#define VSS_ROUNDRECT     1
#define VSS_CIRCLE        2
#define VSS_SIMPLELINE    3
```

2.17 FUNCTION IsClickedEditC : BOOL

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      bUp:BOOL;
      bToggle:BOOL;
      bVisible:BOOL;
END_VAR
```

Is only used with nUp = True and Toggle = TRUE. Otherwise see above.

2.18 FUNCTION ISKeyPressed: BOOL

```
VAR_INPUT
      nKeyCode:INT;
      bUp:BOOL;
      nInfo:INT;
END_VAR
```

Responsible for an interaction between the visualisation and the keybord. The parameter nKeyCode describes the virtual key number, bUp if the key has been released and nInfo, if additional keys(Ctrl or Shift) are necessary.

2.19 FUNCTION Printf : STRING

```
VAR_INPUT
      stFormat: STRING;
      dwValue: DWORD;
      nType: INT;
END_VAR
```

Is to be used just like the C function sprintf, yet only one variable is needed in the formatted string.

2.20 FUNCTION RGBColor : DINT

```
VAR_INPUT
      byRed: BYTE;
      byGreen: BYTE;
      byBlue: BYTE;
END_VAR
```

Returns a colour of three components (red, green und blue).

2.21 FUNCTION SetFill : BOOL

```
VAR_INPUT
      dwFillFlags:DWORD;
      dwFillColor:DWORD;
END_VAR
```

Sets a filling colour which effects subsequent draw events.

Until now only FF_SOLID is used for the fillflags. FF_SOLID creates a homogenous filling colour as CoDeSys does not support a structured filling.

2.22 FUNCTION SetFont : BOOL

```
VAR_INPUT
      stFontName:STRING;
      nFontSize:INT;
      dwFontFlags:DWORD;
      dwFontColor:DWORD;
END_VAR
```

Sets a font which effects subsequent draw events. The FONTFLAGS contain the information whether the font is

#define FF_ITALIC	0x0001
#define FF_BOLD	0x0002
#define FF_UNDERLINE	0x0004
#define FF_STRIKEOUT	0x0008

Note: Per default the font size (nFontSize) is given in Windows style. With target setting „FontInPixelSize“ of type BOOL the actually set size will be given in the font selection dialog.

2.23 FUNCTION SetLine : BOOL

```
VAR_INPUT
      nBorderWidth:INT;
      dwBorderFlags:DWORD;
      dwBorderColor:DWORD;
END_VAR
```

Sets a line type which effects the frame of the subsequent draw event. BORDERWIDTH is the thickness of the line in pixels, dwBorderFlags is the line type and dwBorderColor is the color. The following types are supported:

#define PS_SOLID	0	
#define PS_DASH	1	/* ------- */
#define PS_DOT	2	/* ..... */
#define PS_DASHDOT	3	/* _._._._ */
#define PS_DASHDOTDOT	4	/* _.._.._ */
#define PS_NULL	5	

2.24 FUNCTION StopVisu : BOOL

```
VAR_INPUT
      stName: STRING;
END_VAR
```

StopVisu should delete all bitmaps and reset all changed settings (e.g. by SetLine, SetFill). Until now only called up before the initialisation.

2.25 FUNCTION PushTransformation : BOOL

```
VAR_INPUT
      mXmother:INT;
      mYmother:INT;
      nXChild:INT;
      nYChild:INT;
      mXChild:INT;
      mYChild:INT;
      bIso:BOOL;
      bClipFrame:BOOL;
END_VAR
```

See PopTransformation

2.26 FUNCTION PopTransformation : BOOL

```
VAR_INPUT
      mXmother:INT;
      mYmother:INT;
      nXChild:INT;
      nYChild:INT;
      mXChild:INT;
      mYChild:INT;
      bIso:BOOL;
END_VAR
VAR
END_VAR
```

Defines the scaling of a reference with the right lower corner of the reference and the left upper and right lower corner of the envisaged window in the visualisation.

bIso defines whether the reference is to be distorted or not.

After this command all commands have to be executed adjusted in size.

The scaling is cancelled by pop transformation (receives the same parameters).

Note: If bIso is activated it is difficult to count back.

Suggested codes in C:

> The variables X/YOffset, ScaleX/YMul, ScaleX/Ydiv are static and would be used for the following methods.

Sample code:

Note: Regard the adaptations of the coordinates which are marked red.

```
char RTGraphicPushTrans(short mXMother, short mYMother, short nXChild,
short nYChild, short mXChild, short mYChild, char ISo)
{
	short nXChild = min(nInXChild, mInXChild);
	short mXChild =   max(nInXChild, mInXChild);
	short nYChild = min(nInYChild, mInYChild);
	short mYChild = max(nInYChild, mInYChild);

	if (mXMother == 0 || mYMother == 0)
		return 0;
	if(mXChild - nXChild == 0 || mYChild - nYChild == 0)
	{
		mXChild = mXChild + 1;
		mYChild = mYChild + 1;
	}

	XOffset = RoundShort(XOffset + (nXChild*(ScaleXMul / ScaleXDiv)))+1;
	YOffset = RoundShort(YOffset + (nYChild*(ScaleYMul / ScaleYDiv)))+1;
	if (ISo != 0)
	{
		double dnXChild = nXChild;
		double dnYChild = nYChild;
		double dmXChild = mXChild;
		double dmYChild = mYChild;
		double dmYMother = mYMother;
		double dmXMother = mXMother;

		if    ((dmXChild - dnXChild) / dmXMother <= (dmYChild -
dnYChild) / dmYMother)
		{
			ScaleXMul = (dmXChild - dnXChild - 2)  * ScaleXMul;
			ScaleXDiv = dmXMother * ScaleXDiv;
			ScaleYMul = (dmXChild - dnXChild - 2)  * ScaleYMul;
			ScaleYDiv = dmXMother * ScaleYDiv;
		}
		else
		{
			ScaleXMul = (dmYChild - dnYChild - 2)  * ScaleXMul;
			ScaleXDiv = dmYMother * ScaleXDiv;
			ScaleYMul = (dmYChild - dnYChild - 2)  * ScaleYMul;
			ScaleYDiv = dmYMother * ScaleYDiv;

		}

	}
	else
	{
		ScaleXMul = (mXChild - nXChild - 2) * ScaleXMul;
		ScaleXDiv = mXMother * ScaleXDiv;
```

```
            ScaleYMul = (mYChild - nYChild - 2) * ScaleYMul;
            ScaleYDiv = mYMother * ScaleYDiv;
      }
      return 1;
}

char RTGraphicPopTrans(short mXMother, short mYMother, short nXChild, short
nYChild, short mXChild, short mYChild, char ISo)
{
      short nXChild = min(nInXChild, mInXChild);
      short mXChild =max(nInXChild, mInXChild);
      short nYChild = min(nInYChild, mInYChild);
      short mYChild = max(nInYChild, mInYChild);

      if (mXMother != 0 && mYMother != 0)
      {
            if(mXChild - nXChild == 0 || mYChild - nYChild == 0)
            {
                  mXChild = mXChild + 1;
                  mYChild = mYChild + 1;
            }

            if (ISo != 0)
            {
                  double dnXChild = nXChild;
                  double dnYChild = nYChild;
                  double dmXChild = mXChild;
                  double dmYChild = mYChild;
                  double dmYMother = mYMother;
                  double dmXMother = mXMother;
                  if    ((dmXChild - dnXChild) / dmXMother <= (dmYChild -
dnYChild) / dmYMother)
                  {
                        ScaleXMul = ScaleXMul / (dmXChild - dnXChild - 2);
                        ScaleXDiv = ScaleXDiv / dmXMother;
                        ScaleYMul = ScaleYMul / (dmXChild - dnXChild - 2);
                        ScaleYDiv = ScaleYDiv / dmXMother;
                  }
                  else
                  {
                        ScaleXMul = ScaleXMul / (dmYChild - dnYChild - 2);
                        ScaleXDiv = ScaleXDiv / dmYMother;
                        ScaleYMul = ScaleYMul / (dmYChild - dnYChild - 2);
                        ScaleYDiv = ScaleYDiv / dmYMother;
                  }
            }
            else
            {
                  ScaleXDiv = ScaleXDiv / mXMother;
                  ScaleXMul = ScaleXMul / (mXChild - nXChild - 2);
                  ScaleYDiv = ScaleYDiv / mYMother;
                  ScaleYMul = ScaleYMul / (mYChild - nYChild - 2) ;
```

```
            }
            XOffset = RoundShort(XOffset - (nXChild * (ScaleXMul /
ScaleXDiv))) - 1;
            YOffset = RoundShort(YOffset - (nYChild * (ScaleYMul /
ScaleYDiv))) - 1;
      }
}
```

2.27 FUNCTION ChangePassword: BOOL

```
VAR_INPUT
      pCurrentUserLevel:POINTER TO INT;
      nCurrentNumOfUsers:INT;
      pCurrentUsers:POINTER TO ARRAY [0..NUMOFUSERS] OF STRING;
      pCurrentPasswords:POINTER TO ARRAY [0..NUMOFUSERS] OF STRING;
      pCurrentLevels:POINTER TO ARRAY [0..NUMOFUSERS] OF INT;
END_VAR
```

Opens a dialog to change the password of a specific user level.

2.28 FUNCTION ChangeUserLevel: BOOL

```
VAR_INPUT
      pCurrentUserLevel:POINTER TO INT;
      nCurrentNumOfUsers:INT;
       pCurrentUsers:POINTER TO ARRAY [0..NUMOFUSERS] OF STRING;
      pCurrentPasswords:POINTER TO ARRAY [0..NUMOFUSERS] OF STRING;
      pCurrentLevels:POINTER TO ARRAY [0..NUMOFUSERS] OF INT;
END_VAR
```

Opens a dialog to change the user level.

2.29 FUNCTION GetSurroundRect: BOOL

```
VAR_INPUT
      pnXCorner:POINTER TO INT;
      pnYCorner:POINTER TO INT;
      pmXCorner:POINTER TO INT;
      pmYCorner:POINTER TO INT;
      nNumberOfPoints:INT;
      pnXPoints:POINTER TO INT;
      pnYPoints:POINTER TO INT;
END_VAR
```

Gets the surrounding rect of a polygon.

2.30 FUNCTION MovePolygon: BOOL

```
VAR_INPUT
      nNumberOfPoints:INT;
      pnXPoints:POINTER TO INT;
      pnYPoints:POINTER TO INT;
      nXCenter:INT;
      nYCenter:INT;
      nScale:DINT;
      nAngle:DINT;
      nXOffset:DINT;
      nYOffset:DINT;
END_VAR
```

Moves a polygon.

2.31 FUNCTION CalcRotation: BOOL

```
VAR_INPUT
      pnXCorner:POINTER TO INT;
      pnYCorner:POINTER TO INT;
      pmXCorner:POINTER TO INT;
      pmYCorner:POINTER TO INT;
      nXCenter:INT;
      nYCenter:INT;
      nAngle:DINT;
END_VAR
```

Rotates a rectangle.

2.32 FUNCTION SendEnum: INT

```
VAR_INPUT
      stDescription:STRING;
END_VAR
```

Registers the enum values (string and number) in a buffer and returns the buffer index.

2.33 FUNCTION EnumToString: BOOL

```
VAR_INPUT
      nId:INT;
      pstEnum:POINTER TO STRING;
      nEnum:INT;
END_VAR
```

Converts the enum number to a enum string.

2.34 FUNCTION StringToEnum: BOOL

```
VAR_INPUT
      nId:INT;
      nEnum:POINTER TO INT;
      stEnum:STRING;
END_VAR
```

Converts the enum string to the enum number.

2.35 FUNCTION VesionXXXX : BOOL

```
VAR_INPUT
      Version:INT;
END_VAR
```

Describes the version number of the lib.

2.36 FUNCTION RegisterVariable : SINT

```
VAR_INPUT
      stName: STRING;
      stValue: STRING;
      dwAdr: DWORD;
      nType: INT;
END_VAR
```

Registers a variable by stName, the default value stValue, the address dwAdr and the type nType. These information would be stored in a buffer.

2.37 FUNCTION ExecuteCommand : SINT

```
VAR_INPUT
      stCommand:STRING;
      nType:INT;
END_VAR
```

Executes one of the following commands

#define EC_INIT	1
#define EC_START_REGISTER_DYNAMIC_TEXT_FILES	10
#define EC_REGISTER_DYNAMIC_TEXT_FILE	11
#define EC_END_REGISTER_DYNAMIC_TEXT_FILES	12
#define EC_CHANGE_DYNAMIC_TEXT_LANGUAGE	13
#define EC_LOAD_DYNAMIC_TEXTS	14
#define EC_DYNAMIC_TEXT_HIDE	15
#define EC_EXECUTEPROGRAM	20
#define EC_PRINT	23
#define EC_START_REGISTER_WATCH_LISTS	30
#define EC_START_REGISTER_WATCH_LIST	31

```
#define EC_END_REGISTER_WATCH_LIST          32
#define EC_END_REGISTER_WATCH_LISTS         33
#define EC_DEFINERECEIPT                    34
#define EC_READRECEIPT                      35
#define EC_WRITERECEIPT                     36
#define EC_SAVEWATCH                        37
#define EC_LOADWATCH                        38
```

2.38 FUNCTION IsMovedInRect : SINT

```
VAR_INPUT
      nXCorner:INT;
      nYCorner:INT;
      mXCorner:INT;
      mYCorner:INT;
      pnX:POINTER TO INT;
      pnY:POINTER TO INT;
      nType:INT;
      bVisible:BOOL;
END_VAR
```

The Function IsMovedInRect returns the result of mouse movements on this rectangle.

Return value:

0 : No mouse movement

1 : Mouse movement

In pnX und pnY would be stored the result of the current X/Y position.

2.39 FUNCTION IsMovedInPolygon : SINT

```
VAR_INPUT
      nNumberOfPoints:INT;
      pnXPoints:POINTER TO INT;
      pnYPoints:POINTER TO INT;
      pnX:POINTER TO INT;
      pnY:POINTER TO INT;
      nType:INT;
      bVisible:BOOL;
END_VAR
```

The Function IsMovedInPolygon returns the result of mouse movements on this polygon.

Return value

0 : No mouse movement

1 : Mouse movement

In pnX und pnY would be stored the result of the current X/Y position.

2.40 FUNCTION GetText : BOOL

```
FUNCTION GetText : BOOL
VAR_INPUT
      stResult: STRING(256);
      nResultLength:INT;
      stPrefix: STRING;
      dwID: DWORD;
END_VAR
```

The function GetText can return a language-dependant text from the dynamic textlists. Parameter „stResult“ serves as IN_OUT parameter and gets assigned the text which was found via the prefix string „stPrefix“ and the ID „dwID“. In parameter „nResultLength“ the maximum length of string „stResult“ should be handed over, if that is shorter than 256 characters.

Return value:

FALSE – No matching text found for „stPrefix“ and „dwID”.

TRUE – Matching text found for „stPrefix“ and „dwID“.

2.41 FUNCTION GetTextByStringId: BOOL

```
FUNCTION GetText : BOOL
VAR_INPUT
      stResult: STRING(256);
      nResultLength:INT;
      stPrefix: STRING;
      stID: STRING;
END_VAR
```

Function GetTextByStringId – like function GetText - can return a language-dependant text from the dynamic textlists. The difference is that parameter „stID“ serves for handing over a string defining the ID. Thus it is possible to use an ID like “Text4711”.

Return value:

FALSE – No matching text found for „stPrefix“ and „dwID”.

TRUE – Matching text found for „stPrefix“ and „dwID“.

2.42 FUNCTION GetLastMouseDownEvent : BOOL

```
FUNCTION GetLastMouseDownEvent : BOOL
VAR_INPUT
      pMouseEvent : POINTER TO MOUSEEVENT;
END_VAR
VAR
END_VAR
```

Function GetLastMouseDownEvent returns the last given MouseDown event. Structure MouseEvent is described in the following:

```
TYPE MouseEvent :
STRUCT
      dwCounter : DWORD;
      nXPos : INT;
      nYPos : INT;
END_STRUCT
END_TYPE
```

Parameter “dwCounter” gives the number of MouseDownEvents since system start. With this parameter you can evaluate, whether no, one or multiple events have been created since the last request. Parameters „nXPos“ und nYPos“ return the last screen coordinates.

Return value

No value returned.

2.43 FUNCTION GetLastMouseMoveEvent : BOOL

```
FUNCTION GetLastMouseMoveEvent : BOOL
VAR_INPUT
      pMouseEvent : POINTER TO MOUSEEVENT;
END_VAR
VAR
END_VAR
```

Function GetLastMouseMoveEvent returns the last created MouseMove event. Structure MouseEvent is described above with function GetLastMouseDownEvent.

Return value:

No value returned.

2.44 FUNCTION GetLastMouseUpEvent : BOOL

```
FUNCTION GetLastMouseUpEvent: BOOL
VAR_INPUT
      pMouseEvent : POINTER TO MOUSEEVENT;
END_VAR
VAR
END_VAR
```

Function GetLastMouseUpEvent returns the last created MouseUp event. Structure MouseEvent is described above with function GetLastMouseDownEvent.

Return value:

No value returned.

3 Further requirements of the runtime system

3.1 Mouse handling:

The mouse handling also has to meet certain special requirements. As the visualisation code can not be executed often, mouse actions have to be saved so that later on, during the next run through the corresponding code area, they can be processed. For this purpose an array should be occupied which is able to save an even number of mouse-click actions. The same applies for all mouse movements after the activation of a tap (toggle FALSE) button, they too have to be saved.

3.2 Scanner for the EditControl:

The entries into the EditControl should be checked so that in case of an erroneous entry the EditControl remains open and is not passed on to the input which in this case delivers 0, empty strings or the like.

At least the following types should be supported (compare DrawEditC):

TYPE_BOOL,
TYPE_INT,
TYPE_BYTE,
TYPE_WORD,
TYPE_DINT,
TYPE_DWORD,
TYPE_REAL,
TYPE_TIME,
TYPE_STRING,
TYPE_SINT,
TYPE_USINT,
TYPE_UINT,
TYPE_UDINT,
TYPE_DATE,
TYPE_TOD,
TYPE_DT,
TYPE_LREAL,
TYPE_LINT,
TYPE_ULINT,
TYPE_LWORD,

Change History

Version	Description	Editor	Date
1.0	Rework of the existing version according to German version; New functions: GetText GetTextByStringId, GetLastMouseDownEvent, GetLastMouseMoveEvent, GetLastMouseUpEvent (with CoDeSys V2.3.6.0, #5119)	MN	30.01.2006
1.0	Release	MN	30.01.2006