Definitions of types and constants for C language

```
//*****************************************************************************
//                                          (C) IpeSoft s.r.o. (Ltd.) ZILINA
//   PROJECT :  D2000
//   FILE    :  Imp_def.h
//
//   DESCRIPTION : Constants and types for the import of graphic formats
//
//*****************************************************************************

  //***************************************************************************
  // Constants of gr. object's types
  //***************************************************************************
  #define cLine    =  0 // Line
  #define cPLine   =  1 // Multiline
  #define cDLine   =  2 // Disjointed multiline
  #define cArc     =  3 // Arc
  #define c3Arc    =  4 // 3-point arc
  #define cBox     =  5 // Rectangle
  #define cPAngle  =  6 // Polygon
  #define cCircle  =  7 // Circle
  #define cPiArc   =  8 // Circle sector
  #define cElipse  = 11 // Ellipse
  #define cText    = 12 // ext
  #define cGroup   = 32 // Group of objects

  // extra types
  #define cLineCombined   = -1  // it combines moer lines, containing the same attributes, into one object
                                // - either PolyLine or PolyLineDisjoint
  #define cPLineAdd       = -2  // it adds a set of parameters to the Polyline
                                // (if it has not been created, create a new one)

  #define cFontStyle      = -3  // it creates a text style + automatic creating

                                // when creating some texts - since the ver. 5.00
                                // (one style only can correspond with the given name)

  //***************************************************************************
  // the constants that define the permitted types for some parameters
  //---------------------------------------------------------------------------
  // label in comments
  // (*) - default parameter
  //***************************************************************************

  //***************************************************************************
  // color definition, implicitly the colors from logical palette which is used during import
  //***************************************************************************


  // color base
  #define CLR_BASE_WHITE   =  0
  #define CLR_BASE_YELLOW  = 16
  #define CLR_BASE_CYAN    = 32
  #define CLR_BASE_GREEN   = 48
  #define CLR_BASE_RED     = 64
  #define CLR_BASE_PINK    = 80
  #define CLR_BASE_BLUE    = 96


  // color base - shortcuts
  #define W = CLR_BASE_WHITE
  #define Y = CLR_BASE_YELLOW
  #define C = CLR_BASE_CYAN
  #define G = CLR_BASE_GREEN
  #define R = CLR_BASE_RED
  #define P = CLR_BASE_PINK
  #define B = CLR_BASE_BLUE

  type TColorArr  is ARRAY (0.. 15) OF integer // bright..dark
  // color index
  CLR_WHITE    : constant TColorArr := (W+0,W+1,W+2,W+3,W+4,W+5,6,W+7,W+8,W+9,W+10,W+11,W+12,W+13,W+14,W+15)
  CLR_YELLOW   : constant TColorArr := (Y+0,Y+1,Y+2,Y+3,Y+4,Y+5,6,Y+7,Y+8,Y+9,Y+10,Y+11,Y+12,Y+13,Y+14,Y+15)
  CLR_CYAN     : constant TColorArr := (C+0,C+1,C+2,C+3,C+4,C+5,6,C+7,C+8,C+9,C+10,C+11,C+12,C+13,C+14,C+15)
  CLR_GREEN    : constant TColorArr := (G+0,G+1,G+2,G+3,G+4,G+5,6,G+7,G+8,G+9,G+10,G+11,G+12,G+13,G+14,G+15)
  CLR_RED      : constant TColorArr := (W+0,W+1,W+2,W+3,W+4,W+5,6,W+7,W+8,W+9,W+10,W+11,W+12,W+13,W+14,W+15)
  CLR_PINK     : constant TColorArr := (P+0,P+1,P+2,P+3,P+4,P+5,6,P+7,P+8,P+9,P+10,P+11,P+12,P+13,P+14,P+15)
  CLR_BLUE     : constant TColorArr := (B+0,B+1,B+2,B+3,B+4,B+5,6,B+7,B+8,B+9,B+10,B+11,B+12,B+13,B+14,B+15)
```

```
// some examples of colors
#define i_CLR_WHITE         =  0
#define i_CLR_PALEGRAYLIGHT =  2
#define i_CLR_PALEGRAY      =  4
#define i_CLR_DARKGRAY      = 10
#define i_CLR_DARKGRAYDARK  = 12
#define i_CLR_BLACK         = 15

#define i_CLR_YELLOW        = 22
#define i_CLR_DARKYELLOW    = 26
#define i_CLR_CYAN          = 38
#define i_CLR_DARKCYAN      = 42
#define i_CLR_GREEN         = 54
#define i_CLR_DARKGREEN     = 58
#define i_CLR_RED           = 70
#define i_CLR_DARKRED       = 74
#define i_CLR_PINK          = 86
#define i_CLR_DARKPINK      = 90
#define i_CLR_BLUE          =102
#define i_CLR_DARKBLUE      =106


//*****************************************************************************
// line style
// lineStyles - type TPenStyle is (Solid, Alternate, Dash, Dot, DashDot, DashDotDot, Invisible)
//*****************************************************************************
#define tLS_Solid       = 0  // -----------  (*)
#define tLS_Alternate   = 1  // . . . . . . - only thin line, do not use
#define tLS_Dash        = 2  // - - - - - -
#define tLS_Dot         = 3  // . . . . . .
#define tLS_DashDot     = 4  // - . - . - .
#define tLS_DashDotDot  = 5  // - . . - . .
#define tLS_Invisible   = 6  // unsupported

//*****************************************************************************
// line end
// lineEnd - type TLineEnd  is (Flat,  Square, Round)
//*****************************************************************************
#define tLE_Flat        = 0  // flat (*)
#define tLE_Square      = 1  // square
#define tLE_Round       = 2  // rounded

//*****************************************************************************
// line join
// lineJoin - type TLineJoin is (Bevel, Round,  Miter)
//*****************************************************************************
#define tLJ_Bevel       = 0  // ending at the endpoint (*)
#define tLJ_Round       = 1  // rounded ending behind the endpoint
#define tLJ_Miter       = 2  // square ending behind the endpoint


//*****************************************************************************
// the patterns in Windows
// brushStyle - type TBrushStyle is (Solid, Hollow, BDiagonal, Cross, DiagCross,
//                  FDiagonal,  Horizontal, Vertical)
//*****************************************************************************
#define tBS_Solid       = 0  // solid pattern
#define tBS_Hollow      = 1  // hollow pattern (*)
#define tBS_BDiagonal   = 2
#define tBS_Cross       = 3
#define tBS_DiagCross   = 4
#define tBS_FDiagonal   = 5
#define tBS_CHorizontal = 6
#define tBS_Vertical    = 7


//*****************************************************************************
// text position
// cTextPos - type tTextPos   is (tpAtPos,tpInBox,tpIntoBox)
//*****************************************************************************
#define tTP_ATPOS       = 0  // defined position of the text (*)
#define tTP_INBLOCK     = 1  // text placed in a rectangle
#define tTP_INTOBLOCK   = 2  // text fills the rectangle

//*****************************************************************************
// horizontal alignment of text in rectangle
// cTextCenterH -horizontal center
//*****************************************************************************
#define tTHC_LEFT       = 0  // left
#define tTHC_MIDDLE     = 1  // center (*)
#define tTHC_RIGHT      = 2  // right
```

```
  //*****************************************************************************
  // vertical alignment of text in rectangle
  // cTextCenterV - vertical center
  //*****************************************************************************
  #define tTVC_TOP        = 0  // top
  #define tTVC_MIDDLE     = 1  // middle (*)
  #define tTVC_BOTTOM     = 2  // bottom


  //*****************************************************************************
  // parameters of graphic objects - param's types constants
  //-----------------------------------------------------------------------------
  // comments for the implementation
  //   1. the parameters must be placed in such procedures as they are mentioned in the comment behind the
parameter,
  //      this ensures the parameters will be accepted
  //   2. some of the parameters will not be executed
  //   3. you can set maximum 1000 position points, however version V4.5 accepts at the most first 1+30
  //      if there are more points for polyline, more objects of this type are generated
  //   4. a color index for import index to the default color palette
  //   5. RGB color - from the version V5.0, the colors fill the table with 128 elements; the object Palette will
be created from this table,
  //      the colors in object will be saved in the form of indexes into this table
  //*****************************************************************************

  //-----------------------------------------------------------------------------
  // positions and sizes
  //-----------------------------------------------------------------------------
  #define cPosXY              =1 // 2 * int/float - point position
  #define cPosDXY             =2 // 2 * int/float - a distance from the previous point

  //-----------------------------------------------------------------------------
  // line params
  //-----------------------------------------------------------------------------
  #define cLineColorRGB        =10 // int  - RGB color of object - from the version V5.0
  #define cLineColorIdx        =11 // int  - color index in the local palette
  #define cLineStyle           =12 // int  - line style - lineStyles
  #define cLineWidth           =13 // int  - line width 1..
  #define cLineEnd             =14 // int  - end of thick line - lineEnd
  #define cLineJoin            =15 // int  - polyline join - lineJoin

  //-----------------------------------------------------------------------------
  // fill params
  //-----------------------------------------------------------------------------
  #define cFillColorRGB        =30 // int - RGB color of object - from the version V5.0
  #define cFillColorIdx        =31 // int - color index in the local palette
  #define cFillPattern         =32 // int  - line style - brushStyle

  //-----------------------------------------------------------------------------
  // circles
  //-----------------------------------------------------------------------------
  #define cCircleRadial        =50 // int/float - radius

  #define cCircleAngleDegStart =56 // int/float - degree
  #define cCircleAngleDegEnd   =57 // int/float - degree
  #define cCircleAngleDegSize  =58 // int/float - degree

  #define cCircleAngleRadStart =59 // int/float - radiant
  #define cCircleAngleRadEnd   =60 // int/float - radiant
  #define cCircleAngleRadSize  =61 // int/float - degree

  //-----------------------------------------------------------------------------
  // text params
  //-----------------------------------------------------------------------------
  #define cTextText            =70 // text
  #define cTextColorRGB        =71 // int - RGB color of object - from the version V5.0
  #define cTextColorIdx        =72 // int - color index in the local palette

  #define cTextPos             =73 // int - position in a rectangle, tTP_ATPOS,..
  #define cTextCenterH         =76 // int - horizontal alignment in rectangle
  #define cTextCenterV         =77 // int - vertical alignment in rectangle
```

```
//*****************************************************************************
// creating of the text style
// text style is an object with parameter CFont or when creating a text, there are these conditions
//  * font will not be created unless all the parameters are set
//  * if cFontStyleName style exists, the new one will not be created, the existing one will be used
//  * if the text style is developed when creating the text, it will be used
//  * if cFontStyleName is set when creating the text, the text will be drawn by this style (if it exists)
//*****************************************************************************
#define cFontStyleName       =200 //  string  - font style name
#define cFontName            =201 //  string  - font name
#define cFontSize            =202 //  Integer - font size
#define cFontBold            =203 //  Boolean - attribute for bold
#define cFontItalic          =204 //  Boolean - attribute for italic
#define cFontUnderline       =205 //  Boolean - attribute for underline
#define cFontStrikeOut       =206 //  Boolean - attribute for strikeout
#define cFontCharSet         =207 //  Integer - character set (Western, Central European)

//*****************************************************************************
// types of obj.actions
//*****************************************************************************
type tObjAction  is (closeFigure,// close polyline , a change to polygon
                     closeObject,// close object
                     closeGroup, // close object and group
                     closeAll)   // close object and all groups
                                 // called internal, after ending the import

//*****************************************************************************
// access to functions - definitions
//*****************************************************************************

void _stdcall (*CreateObj)(int)
void _stdcall (*ObjAtion)(tObjAction)

void _stdcall (*Set_string )(int,char far * )
void _stdcall (*Set_boolean)(int,bool)
void _stdcall (*Set_integer)(int,int)
void _stdcall (*Set_float)(int,real)
void _stdcall (*Set_integer2)(int,int,int)
void _stdcall (*Set_float2)(int,real,real)

// info text that occurs during import
void _stdcall (*ShowInfo)(char far * )

//*****************************************************************************
// access to functions
//*****************************************************************************
createObj    : tCreateObj    := null;
objAction    : tObjAtion     := null;


set_string   : tset_string   := null;
set_boolean  : tset_boolean  := null;
set_integer  : tset_integer  := null;
set_float    : tset_float    := null;
set_integer2 : tset_integer2 := null;
set_float2   : tset_float2   := null;

// info text that occurs during import
ShowInfo     : tShowInfo     := null;

//*****************************************************************************
// access to import functions
//*****************************************************************************

#define pShowInfo     = 0
#define pcreateObj    = 1
#define pobjAction    = 2
#define pset_string   = 3
#define pset_boolean  = 4
#define pset_integer  = 5
#define pset_float    = 6
#define pset_integer2 = 7
#define pset_float2   = 8

// pointer to 1.char of null terminating string
#define maxResStr     = 10000

// import name and file type
void _stdcall (*GetFileType)(char far * description,char far * extension)

// initialization of call-back procedures
void _stdcall (*ImportConnect)(int procType,int procAddr )
```

```
  // load autocad dxf file
  void _stdcall (*ImportConnect)(int procType,int procAddr )



//***************************************************************************

//---------------------------------------------------------------------------
// Revisions History --
//---------------------------------------------------------------------------
//  0.00     28.02.01 - creating of module
//---------------------------------------------------------------------------
```

Related pages:

Import of vector formats into pictures of D2000 System