JPCAD Control Help File

Methods, Events and Constants
OLE type aliases

Notes for Visual Basic users:
When debugging application written in Visual Basic you need to load Visual Basic executable (VB.EXE) instead of your application. It also means that you cannot debug more than one Visual Basic application at a time. When you stop debugging, you need to unload Visual Basic executable from JPCAD otherwise you will not be able to re-start debugging session

A_StatusE

```
enum A_StatusE
        {
        A_CMD_CALL          = 0,
        A_LOAD              = 1,
        A_UNLOAD            = 2,
        A_NEW           = 3,
        A_OPEN_BEFORE       = 4,
        A_OPEN_AFTER        = 5,
        A_SAVE_BEFORE       = 6,
        A_SAVE_AFTER        = 7,
        A_DISCARD           = 8,
        A_END               = 100
        };
```

This is parameter to Status event.

A_CMD_CALL	JPCAD calls your command, command code is second parameter
A_LOAD	JPCAD loads your application, do the initialization and define commands here
A_UNLOAD	JPCAD unloads your application, do the close up here, but do not quit your application (see A_END)
A_NEW	new drawing open, prototype loaded
A_OPEN_BEFORE	new drawing will be loaded
A_OPEN_AFTER	new drawing was loaded
A_SAVE_BEFORE	old drawing is to be saved
A_SAVE_AFTER	old drawing was saved
A_DISCARD	old drawing is to be discarded
A_END	Quit your application upon receiving this event

A_DrawMethodE

```
enum A_DrawMethodE
        {
        A_DRAW_NORMAL     = 0,
        A_DRAW_ERASE      = 1,
        A_DRAW_HIGHLIGHT  = 2,
        };
```

Defines type of color of entity for Draw function.

A_DRAW_NORMAL

Draw entity with original color

A_DRAW_ERASE

Erase entity from screen (draw entity with color of background)

A_DRAW_HIGHLIGHT

Highlite enity color

A_DisplayE

```
enum    A_DisplayE
        {
        A_DISPLAY_STOP      = 0,
        A_DISPLAY_FLUSH     = 1,
        A_DISPLAY_REDRAW    = 2
        };
```

A_DISPLAY_STOP	stop display
A_DISPLAY_FLUSH	flush al pending changes
A_DISPLAY_REDRAW	redraw entire drawing

A_UndoE

```
enum    A_UndoE
        {
        A_UNDO_STEP                     = 0,
        A_REDO_STEP                     = 1,
        A_UNDO_BEGIN            = 2,
        A_UNDO_END              = 3,
        A_UNDO_REMOVE_ALL= 4
        };
```

A_EntTypeE

```
enum    A_EntTypeE
        {
        A_ENT_UNKNOWN     = 0,
        A_ENT_ARC         = 1,
        A_ENT_ATTR        = 2,
        A_ENT_BLOCK       = 3,
        A_ENT_CIRCLE              = 4,
        A_ENT_INSERT              = 5,
        A_ENT_LAYER       = 6,
        A_ENT_LINE        = 7,
        A_ENT_SOLID       = 8,
        A_ENT_TEXT        = 9,
        A_ENT_TEXTSTYLE   = 10,
        };
```

Return value from GetEntType function.

A_V_TypeE

```
enum   A_V_TypeE
       {
       A_V_STRING   = 1,
       A_V_DOUBLE   = 2,
       A_V_INTEGER = 3,
       A_V_POINT    = 4,
       };
```

Type of variable. Used in Variables.

A_V_NONE
Variable has no value.
A_V_STRING
Variable of type char*.
A_V_DOUBLE
Variable of type double.
A_V_INTEGER
Variable of type int.
A_V_POINT
Variable of type A_PointS.

A_V_LocationE

```
enum    A_V_LocationE
        {
        A_V_INI                 = 1,
        A_V_NOUNDO = 2,
        A_V_UNDO      = 3
        };
```

User define vaiable location. See V_Registry.

A_V_INI
Variable will be placed in JPCAD.INI file. It will be persistent between JPCAD sessions.
A_V_NOUNDO
Variable will be placed in drawing file without undo information.
A_V_UNDO
Variable will be placed in drawing file with undo information. You can define a variable of type A_V_INTEGER and place type of A_V_UNDO to manage consistency between JPCAD drawing and external information during UNDO/REDO calls. Any time you make a change in your extrenal data increase this variable by one, when you recieve notification, that UNDO/REDO was called, you can check the value of your variable and make necessary changes to your data.

A_A_TypeE

```
enum    A_A_TypeE
                {
                A_A_ENTITY    = 0
                };
```

Type of element array. Currently only A_A_ENTITY is supported.

A_X_TypeE

```
enum A_X_TypeE
        {
        A_X_ARRAY    = 1,
        A_X_STRING   = 2,
        A_X_CHAR     = 3,
        A_X_SHORT    = 4,
        A_X_LONG     = 5,
        A_X_DOUBLE   = 6,
        A_X_POINT    = 7,
        A_X_LENGTH   = 8,
        A_X_ANGLE    = 9,
        A_X_MIRROR   = 10,
        A_X_POSITION         = 11,
        A_X_VECTOR   = 12,
        A_X_DIRECTION        = 13,
        A_X_ENTITY   = 14,
        A_X_VCHUNK   = 15,
        A_X_CCHUNK   = 16
        }
```

Type of X-data.

DefCmd

long DefCmd(
BSTR CmdName,
long CmdCode,
long bTransparent)

Define new application command

Parameters

(I) pCmdName Name of the new JPCAD command. Only the A_MAX_NAME characters are recognized
(I) CmdCode ID of new command. This is the ID you will get when ADK calls your cbStatus function
(I) bTransparent TRUE if this command can be called transparently (possible reentrancy!)

Return

0 Success

Description

Define new external JPCAD command. The pCmdName string will be used to call this function.

Note

If you use a name of already defined function, you will redefine its behaviour because the newly defined functions are placed on top of serach order. When you undefine this function, the old one will be restored.

UnDefCmd

long UnDefCmd(
long CmdCode)

Undefine previously defined command.

Parameters

(I) CmdCode ID of command to undefine

Return

0 Success

Description

Undefine function defined by DefCmd.

Prompt

long Prompt(
BSTR Prompt)

Display prompt.

Parameters

(I) pPrompt Prompt

Return

0 Success

ARC_Make

```
long        ARC_Make(
        A_PointS    Center,
        double      Radius,
        double      StartA,
        double      EndA,
        long        Layer,
        long        Color,
        long        LineType,
        double      Width,
        long        bReferenced,
        long        *pArc)
```

Create arc entity.

Parameters

(I) Center	Center
(I) Radius	Radius
(I) StartA	Start angle (from X axis counterclockwise)
(I) EndA	End angle
(I) Layer	Layer
(I) Color	Color
(I) LineType	LineType
(I) Width	Line width
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pArc	Arc

Return

0	Success
-1	Error

ARC_Get

```
long        ARC_Get(
        long        Arc,
        A_PointS    *pCenter,
        double      *pRadius,
        double      *pStartA,
        double      *pEndA,
        long        *pLayer,
        long        *pColor,
        long        *pLineType,
        double      *pWidth)
```

Get arc data.

Parameters

(I) Arc	Arc
(O) pCenter	Center
(O) pRadius	Radius
(O) pStartA	Start angle (from X axis counterclockwise)
(O) pEndA	End angle
(O pLayer	Layer
(O) pColor	Color
(O) pLineType	LineType
(O) pWidth	Line width

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

ARC_Change

```
long        ARC_Change(
        long        Arc,
        A_PointS    Center,
        double      Radius,
        double      StartA,
        double      EndA,
        long        Layer,
        long        Color,
        long        LineType,
        double      Width)
```

Change arc entity.

Parameters

(I) Arc	Arc
(I) Center	Center
(I) Radius	Radius
(I) StartA	Start angle (from X axis counterclockwise)
(I) EndA	End angle
(I) Layer	Layer
(I) Color	Color
(I) LineType	LineType
(I) Width	Line width

Return

0	Success
-1	Error

ATTR_Make

```
long        ATTR_Make(
        BSTR        Tag,
        BSTR        Prompt,
        BSTR        Text,
        A_PointS    Point,
        double      Height,
        double      Angle,
        long        TextStyle,
        long        Align,
        long        Layer,
        long        Color,
        long        fFlags,
        long        bReferenced,
        long        *pAttr)
```

Create attribute definition entity.

Parameters

(I) pTag	Tag (only A_MAX_NAME characters are considered)
(I) pPrompt	Prompt
(I) pText	Text
(I) Point	Point
(I) Height	Height
(I) Angle	Angle
(I) TextStyle	Text style
(I) Align	Align
(I) Layer	Layer
(I) Color	Color
(I) fFlags	Flags
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pAttr	Attribute

Return

0	Success
-1	Error

ATTR_Get

```
long        ATTR_Get(
        long        Attr,
        BSTR        *pTag,
        BSTR        *pPrompt,
        BSTR        *pText,
        A_PointS    *pPoint,
        double      *pHeight,
        double      *pAngle,
        long        *pTextStyle,
        long        *pAlign,
        long        *pLayer,
        long        *pColor,
        long        *pfFlags)
```

Get attribute data.

Parameters

(I) Attr	Attribute
(O) pTag	Tag
(O) ppPrompt	Prompt
(O) ppText	Text
(O) pPoint	Point
(O) pHeight	Height
(O) pAngle	Angle
(O) pTextStyle	Text style
(O) pAlign	Align
(O) pLayer	Layer
(O) pColor	Color
(O) pfFlags	Flags

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

ATTR_Change

```
long        ATTR_Change(
    long        Attr,
    BSTR        Tag,
    BSTR        Prompt,
    BSTR        Text,
    A_PointS    Point,
    double      Height,
    double      Angle,
    long        TextStyle,
    long        Align,
    long        Layer,
    long        Color,
    long        fFlags)
```

Change attribute data.

Parameters

(I) Attr	Attribute
(I) pTag	Tag (only A_MAX_NAME characters are considered)
(I) pPrompt	Prompt
(I) pText	Text
(I) Point	Point
(I) Height	Height
(I) Angle	Angle
(I) TextStyle	Text style
(I) Align	Align
(I) Layer	Layer
(I) Color	Color
(I) fFlags	Flags

Return

0	Success
-1	Error

BLOCK_Make

long BLOCK_Make(
A_MatrixS Trans,
A_MatrixS TransInverse,
BSTR Name,
long Array,
long *pBlock)

Create block entity.

Parameters

(I) Trans	Transformation
(I) TransInverse	Inverse transformation to Trans
(I) pName	Name (only A_MAX_NAME characters are considered)
(I) Array	Array of block entities
(O) pBlock	Block

Return

0	Success
-1	Error

BLOCK_Get

```
long        BLOCK_Get(
        long        Block,
        A_MatrixS   *pTrans,
        A_MatrixS   *pTransInverse,
        BSTR        *pName,
        long        *pArray,
        long        *pReferenced)
```

Get block data.

Parameters

(I) Block	Block
(O) pTrans	Transformation
(O) pTransInverse	Inverse transformation to Trans
(O) pName	Name
(O) pArray	Array of block entities
(O) pReferenced	Number of references to this block

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

BLOCK_Change

long BLOCK_Change(
long Block,
A_MatrixS Trans,
A_MatrixS TransInverse,
BSTR Name,
long Array)

Change block data.

Parameters

(I) Block	Block
(I) Trans	Transformation
(I) TransInverse	Inverse transformation to Trans
(I) pName	Name (only A_MAX_NAME characters are considered)
(I) Array	Array of block entities

Return

0	Success
-1	Error

BLOCK_GetEnt

long BLOCK_GetEnt(
BSTR Name,
long *pBlock)

Get block by name.

Parameters

(I) pName	Block name (only A_MAX_NAME characters are considered)
(O) pBlock	Block

Return

0	Success
-1	Error

BLOCK_Num

long BLOCK_Num(void)
Get number of blocks.

Return
Number of blocks.

BLOCK_GetNth

```
long        BLOCK_GetNth(
    long        nIndex,
    long        *pBlock)
```

Get block ny index.

Parameters

(I) nIndex	Block index (zero based)
(O) pBlock	Block

Return

0	Success
-1	Error

CIRCLE_Make

```
long        CIRCLE_Make(
        A_PointS    Center,
        double      Radius,
        long        Layer,
        long        Color,
        long        LineType,
        double      Width,
        long        bReferenced,
        long        *pCircle)
```

Create circle entity.

Parameters

(I) Center	Center
(I) Radius	Radius
(I) Layer	Layer
(I) Color	Color
(I) LineType	LineType
(I) Width	Width
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pCircle	Circle

Return

0	Success
-1	Error

CIRCLE_Get

```
long        CIRCLE_Get(
        long        Circle,
        A_PointS    *pCenter,
        double      *pRadius,
        long        *pLayer,
        long        *pColor,
        long        *pLineType,
        double      *pWidth)
```

Get circle data.

Parameters

(I) Circle	Circle
(O) pCenter	Center
(O) pRadius	Radius
(O) pLayer	Layer
(O) pColor	Color
(O) pLineType	LineType
(O) pWidth	Width

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

CIRCLE_Change

```
long        CIRCLE_Change(
    long        Circle,
    A_PointS    Center,
    double      Radius,
    long        Layer,
    long        Color,
    long        LineType,
    double      Width)
```

Change circle data.

Parameters

(I) Circle	Circle
(I) Center	Center
(I) Radius	Radius
(I) Layer	Layer
(I) Color	Color
(I) LineType	LineType
(I) Width	Width

Return

0	Success
-1	Error

INSERT_Make

```
long        INSERT_Make(
        long        Block,
        A_MatrixS   Trans,
        A_MatrixS   TransInverse,
        long        Layer,
        long        Color,
        long        LineType,
        long        Array,
        long        bReferenced,
        long        *pInsert)
```

Create insert entity.

Parameters

(I) Block	Block
(I) Trans	Transformation
(I) TransInverse	Inverse transformation to Trans
(I) Layer	Layer
(I) Color	Color
(I) LineType	Line type
(I) Array	Array of attributes
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pInsert	Insert

Return

0	Success
-1	Error

INSERT_Get

```
long        INSERT_Get(
        long        Insert,
        long        *pBlock,
        A_MatrixS   *pTrans,
        A_MatrixS   *pTransInverse,
        long        *pLayer,
        long        *pColor,
        long        *pLineType,
        long        *pArray)
```

Get data of insert.

Parameters

(I) Insert	Insert
(O) pBlock	Block
(O) pTrans	Transformation
(O) pTransInverse	Inverse transformation to Trans
(O) pLayer	Layer
(O) pColor	Color
(O) pLineType	Line type
(O) pArray	Array of attributes

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

INSERT_Change

```
long        INSERT_Change(
    long        Insert,
    long        Block,
    A_MatrixS   Trans,
    A_MatrixS   TransInverse,
    long        Layer,
    long        Color,
    long        LineType,
    long        Array)
```

Change insert data.

Parameters

(I) Insert	Insert
(I) Block	Block
(I) Trans	Transformation
(I) TransInverse	Inverse transformation to Trans
(I) Layer	Layer
(I) Color	Color
(I) LineType	Line type
(I) Array	Array of attributes

Return

0	Success
-1	Error

INSERT_AttrGet

```
long            INSERT_AttrGet(
        long            Block,
        A_MatrixS       *pTrans,
        long            *pArray)
```

Get all attributes from block definition.

Parameters

(I) Block	Block entity
(O) pTrans	Transformation of block
(O) pAttr	Array of attribute entities of block

Return

0	Success
-1	Error

INSERT_AttrMake

```
long        INSERT_AttrMake(
    long        Attr,
    A_MatrixS   Trans,
    BSTR        Text,
    long        *pAttr)
```

Create attribute entity from attribute definition

Parameters

(I) Attr	Attribute definition
(I) Trans	Block transformation
(I) pText	Text of attribute
(O) pAttr	Attribute

Return

0	Success
-1	Error

LAYER_Make

```
long        LAYER_Make(
    BSTR        Name,
    long        Color,
    long        LineType,
    long        State,
    long        *pLayer)
```

Create layer entity.

Parameters

(I) pName	Name (only A_MAX_NAME characters are considered)
(I) Color	Color
(I) LineType	Line type
(I) State	State
(O) pLayer	Layer

Return

0	Success
-1	Error

LAYER_Get

```
long        LAYER_Get(
    long        Layer,
    BSTR        *pName,
    long        *pColor,
    long        *pLineType,
    long        *pState,
    long        *pReferenced)
```

Get layer data.

Parameters

(I) Layer	Layer
(O) pName	Name
(O) pColor	Color
(O) pLineType	Line type
(O) pState	State
(O) pReferenced	Number of references to this layer

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

LAYER_Change

```
long        LAYER_Change(
    long        Layer,
    BSTR        Name,
    long        Color,
    long        LineType,
    long        State)
```

Change layer data.

Parameters

(I) Layer	Layer
(I) pName	Name (only A_MAX_NAME characters are considered)
(I) Color	Color
(I) LineType	Line type
(I) State	State

Return

0	Success
-1	Error

LAYER_GetCurrent

long LAYER_GetCurrent(
long *pLayer)

Get current layer.

Parameters

(O) pLayer Layer

Return

0 Success

LAYER_SetCurrent

long LAYER_SetCurrent(
long Layer)

Set current layer

Parameters

(I) Layer Layer

Return

0 Success
-1 Error

LAYER_GetEnt

```
long        LAYER_GetEnt(
    BSTR        Name,
    long        *pLayer)
```

Get layer by name.

Parameters

(I) pName	Layer name (only A_MAX_NAME characters are considered)
(O) pLayer	Layer

Return

0	Success
-1	Error

LAYER_Num

long LAYER_Num(void)

Get number of layers.

Return

Number of layers.

LAYER_GetNth

```
long        LAYER_GetNth(
        long        nIndex,
        long        *pLayer)
```

Get layer by index.

Parameters

(I) nIndex	Index of layer
(O) pLayer	Layer

Return

0	Success
-1	Error

LINE_Make

```
long        LINE_Make(
        A_PointS    StartPoint,
        A_PointS    EndPoint,
        long        Layer,
        long        Color,
        long        LineType,
        double      Width,
        long        bReferenced,
        long        *pLine)
```

Create line entity.

Parameters

(I) StartPoint	Start point
(I) EndPoint	End point
(I) Layer	Layer
(I) Color	Color
(I) LineType	Line type
(I) Width	Width
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pLine	Line

Return

0	Success
-1	Error

LINE_Get

```
long        LINE_Get(
        long        Line,
        A_PointS    *pStartPoint,
        A_PointS    *pEndPoint,
        long        *pLayer,
        long        *pColor,
        long        *pLineType,
        double      *pWidth)
```

Get line data.

Parameters

(I) Line	Line
(O) pStartPoint	Start point
(O) pEndPoint	End point
(O) pLayer	Layer
(O) pColor	Color
(O) pLineType	Line type
(O) pWidth	Width

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

LINE_Change

```
long        LINE_Change(
    long        Line,
    A_PointS    StartPoint,
    A_PointS    EndPoint,
    long        Layer,
    long        Color,
    long        LineType,
    double      Width)
```

Change line data.

Parameters

(I) Line	Line
(I) StartPoint	Start point
(I) EndPoint	End point
(I) Layer	Layer
(I) Color	Color
(I) LineType	Line type
(I) Width	Width

Return

0	Success
-1	Error

A_SOLID_Make

```
long        SOLID_Make(
        A_PointS    Point1,
        A_PointS    Point2,
        A_PointS    Point3,
        A_PointS    Point4,
        long        Layer,
        long        Color,
        long        bReferenced,
        long        *pSolid)
```

Create solid entity.

Parameters

(I) Point1	Point (1)
(I) Point2	Point (2)
(I) Point3	Point (3)
(I) Point4	Point (4)
(I) Layer	Layer
(I) Color	Color
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pSolid	Solid

Return

0	Success
-1	Error

SOLID_Get

```
long        SOLID_Get(
        long        Solid,
        A_PointS    *pPoint1,
        A_PointS    *pPoint2,
        A_PointS    *pPoint3,
        A_PointS    *pPoint4,
        long        *pLayer,
        long        *pColor)
```

Get solid data.

Parameters

(I) Solid	Solid
(O) pPoint1	Point (1)
(O) pPoint2	Point (2)
(O) pPoint3	Point (3)
(O) pPoint4	Point (4)
(O) pLayer	Layer
(O) pColor	Color

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

SOLID_Change

```
long        SOLID_Change(
    long        Solid,
    A_PointS    Point1,
    A_PointS    Point2,
    A_PointS    Point3,
    A_PointS    Point4,
    long        Layer,
    long        Color)
```

Change solid data.

Parameters

(I) Solid	Solid
(I) Point1	Point (1)
(I) Point2	Point (2)
(I) Point3	Point (3)
(I) Point4	Point (4)
(I) Layer	Layer
(I) Color	Color

Return

0	Success
-1	Error

TEXT_Make

```
long        TEXT_Make(
    BSTR        String,
    A_PointS    Point,
    double      Height,
    double      Angle,
    long        Style,
    long        Align,
    long        Layer,
    long        Color,
    long        bReferenced,
    long        *pText)
```

Create text entity.

Parameters

(I) pString	String
(I) Point	Point
(I) Height	Height
(I) Angle	Angle
(I) Style	Style
(I) Align	Align
(I) Layer	Layer
(I) Color	Color
(I) bReferenced	If TRUE, the entity will be created but not displayed. This is useful when creating blocks
(O) pText	Text

Return

0	Success
-1	Error

TEXT_Get

```
long        TEXT_Get(
        long        Text,
        BSTR        *pString,
        A_PointS    *pPoint,
        double      *pHeight,
        double      *pAngle,
        long        *pStyle,
        long        *pAlign,
        long        *pLayer,
        long        *pColor)
```

Get text data.

Parameters

(I) Text	Text
(O) ppString	String
(O) pPoint	Point
(O) pHeight	Height
(O) pAngle	Angle
(O) pStyle	Style
(O) pAlign	Align
(O) pLayer	Layer
(O) pColor	Color

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

TEXT_Change

```
long            TEXT_Change(
        long            Text,
        BSTR            String,
        A_PointS        Point,
        double          Height,
        double          Angle,
        long            Style,
        long            Align,
        long            Layer,
        long            Color)
```

Change text data.

Parameters

(I) Text	Text
(I) pString	String
(I) Point	Point
(I) Height	Height
(I) Angle	Angle
(I) Style	Style
(I) Align	Align
(I) Layer	Layer
(I) Color	Color

Return

0	Success
-1	Error

TEXTSTYLE_Make

```
long        TEXTSTYLE_Make(
        BSTR        Name,
        double      Height,
        long        Weight,
        long        Effects,
        BSTR        Font,
        long        CharSet,
        long        *pTextStyle)
```

Create text style entity.

Parameters

(I) pName	Name (only A_MAX_NAME characters are considered)
(I) Height	Height
(I) Weight	Weight
(I) Effects	Effects
(I) pFont	Font (only A_MAX_NAME characters are considered)
(I) CharSet	Character set
(O) pTextStyle	Text style

Return

0	Success
-1	Error

TEXTSTYLE_Get

```
long        TEXTSTYLE_Get(
        long        TextStyle,
        BSTR        *pName,
        double      *pHeight,
        long        *pWeight,
        long        *pEffects,
        BSTR        *pFont,
        long        *pCharSet,
        long        *pReferenced)
```

Get text style data.

Parameters

(I) TextStyle	Text style
(O) pName	Name
(O) pHeight	Height
(O) pWeight	Weight
(O) pEffects	Effects
(O) pFont	Font
(O) pCharSet	Character set

Return

0	Success
-1	Error

Note

You can supply NULL to any of the output parameters if you don't need it.

TEXTSTYLE_Change

```
long        TEXTSTYLE_Change(
    long        TextStyle,
    BSTR        Name,
    double      Height,
    long        Weight,
    long        Effects,
    BSTR        Font,
    long        CharSet)
```

Change text style data.

Parameters

(I) TextStyle	Text style
(I) Name	Name (only A_MAX_NAME characters are considered)
(I) Height	Height
(I) Weight	Weight
(I) Effects	Effects
(I) Font	Font (only A_MAX_NAME characters are considered)
(I) CharSet	Character set

Return

0	Success
-1	Error

TEXTSTYLE_GetCurrent

long TEXTSTYLE_GetCurrent(
long *pTextStyle)

Get current text style.

Parameters

(O) pTextStyle Text style

Return

0 Success
-1 Error

TEXTSTYLE_SetCurrent

long TEXTSTYLE_SetCurrent(
long TextStyle)

Set current text style.

Parameters

(I) TextStyle Text style

Return

0 Success
-1 Error

TEXTSTYLE_GetEnt

long TEXTSTYLE_GetEnt(
BSTR Name,
long *pTextStyle)

Get text style by name.

Parameters

(I) pName	Name (only A_MAX_NAME characters are considered)
(O) pTextStyle	Text style

Return

0	Success
-1	Error

TEXTSTYLE_Num

long TEXTSTYLE_Num(void)

Get number of text styles.

Return

Number of text styles.

TEXTSTYLE_GetNth

long TEXTSTYLE_GetNth(
long nIndex,
long *pTextStyle)

Get text style by index.

Parameters

(I) nIndex	Index
(O) pTextStyle	Text style

Return

0	Success
-1	Error

GetEnt

```
long        GetEnt(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        long        *pEntity,
        A_PointS    *pPoint)
```

Prompt user to select entity.

Parameters

(I) pPrompt	User supplied prompt, standard prompt if NULL
(I) pKeywords	Keywords string
(I) pDefault	Default value
(O) pEntity	Selected entity
(O) pPoint	Picked point. NULL if not useful

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_BAD_SEL, A_GET_CANCEL

Description

Prompt user to select one entity. Standard prompt will be used.

GetSelection

long GetSelection(
long *pArray)

Prompt user to select entities

Parameters

(O) pArray Array of entities

Return

A_GET_OK, A_GET_CANCEL

Description

Prompt user to select entities. Use standard entity selection prompt.

GetString

```
long        GetString(
        BSTR        Prompt,
        BSTR        Default,
        BSTR        *pString)
```

Prompt user to enter string

Parameters

(I) pPrompt	User supplied prompt
(I) pDefault	Default value
(O) ppString	String

Return

A_GET_OK, A_GET_DEFAULT, A_GET_CANCEL

GetLong

```
long        GetLong(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        long        *pLong)
```

Prompt user to enter integer value

Parameters

(I) pPrompt	User supplied prompt
(I) pKeywords	Keywords string
(I) pDefault	Default value
(O) pLong	Long

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

GetDouble

```
long        GetDouble(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        double      *pDouble)
```

Prompt user to enter real value

Parameters

(I) pPrompt	User supplied prompt
(I) pKeywords	Keywords string
(I) pDefault	Default value
(O) pDouble	Double

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

GetPoint

```
long        GetPoint(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        A_PointS    *pPoint)
```

Prompt user to enter point

Parameters

(I) pPrompt	User supplied prompt
(I) pKeywords	Keywords string
(I) pDefault	Default value
(O) pPoint	Point

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

GetPointDrag

```
long        GetPointDrag(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        long        Array,
        const A_PointS      *pPointFrom,
        A_GetDragF  *pDrag,
        void        *pUserData,
        A_PointS    *pPoint)
```

Prompt user to select point while dragging entities

Parameters

(I) pPrompt	User supplied prompt
(I) pKeywords	Keywords string
(I) pDefault	Default value
(I) Array	Array of entities to drag (can be empty)
(I) pPointFrom	Start point of dragging (can be NULL if no startpoint is suitable)
(I) pDrag	Drag callback function (see A_GetDragF)
(I) pUserData	Pointer to user supplied data. This pointer is passed to pDrag function.
(O) pPoint	Point

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

Description

Use this function to drag entities to desired location.

GetAngle

```
long        GetAngle(
        BSTR        Prompt,
        BSTR        Keywords,
        BSTR        Default,
        long        Array,
        A_PointS    BasePoint,
        double      *pBaseAngle,
        double      *pAngle)
```

Prompt user for angle.

Parameters

(I) pPrompt	User supplied prompt, standard prompt if NULL
(I) pKeywords	Keywords string
(I) pDefault	Default value
(I) Array	Array of entities to drag
(I) BasePoint	Angle base point
X) pBaseAngle	Base angle (measured from X-axis counterclocwise in degrees). if you supply NULL, the BASEANGLE variable will be used instead
(O) pAngle	Angle (measured from base angle counterclockwise in dergrees)

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

Description

Prompt the user for angle. Standard angle selection method will be used.

GetKWord

long GetKWord(
BSTR Prompt,
BSTR Keywords,
BSTR Default)

Prompt user for keyword

Parameters

(I) pPrompt User supplied prompt
(I) pKeywords Keywords string
(I) pDefault Default value

Return

A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

DrawLine

long DrawLine(
A_PointS Start,
A_PointS End)

Draw line shape.

Parameters

(I) Start Start
(I) End End

Return

0 Success

Description

Draw additional drag data. This function can be called only from withing GetPointDrag event handler function.

DrawArc

long DrawArc(
A_PointS Center,
double Radius,
double StartAngle,
double EndAngle)

Draw line shape.

Parameters

(I) Center	Center
(I) Radius	Radius
(I) StartAngle	Start angle
(I) EndAngle	End angle

Return

0	Success

Description

Draw additional drag data. This function can be called only from withing GetPointDrag event handler function.

WriteEnt

long WriteEnt(
BSTR FileName,
long Entity)

Write entity to file.

Parameters

(I) pFileName	File name with extension
(I) Entity	Entity

Return

0	Success
-1	Error

ReadEnt

long ReadEnt(
BSTR FileName,
long *pEntity)

Read entity from file.

Parameters

(I) pFileName	File name
(O) pEntity	New entity

Return

0	Success
-1	Error

DelEnt

long DelEnt(
long Entity)

Deletes an entity from database.

Parameters

(I) Entity	Entity to delete

Return

0	success
-1	Entity cannot be deleted

GetAllEnts

long GetAllEnts(
long *pArray)

Get all entities from database.

Parameters

(O) pArray Selection set of all entities in database

Return

0 Success

Draw

long Draw(
long Entity,
long Method) /*actual type is A_DrawMethodE*/

Redraw entity.

Parameters

(I) Entity Entity to redraw
(I) Method Redraw method. See A_DrawMethodE.

Return

0 Success

Description

Use this function if you need to highlite/dehighlite an entity.

Display

long Display(
long fDisplay) /*actual type is A_DisplayE*/

Control drawing operation.

Parameters

(I) fDisplay Operation to provide

Return

0 Success

GetWinTrans

```
long        GetWinTrans(
    long        Window,
    A_MatrixS   *pTrans)
```

Get window transformation.

Parameters

(I) Window	Window index. -1 means main window
(O) pTrans	Window transformation from DCS to WCS

Return

0	Success
-1	Bad window index

SetWinTrans

long SetWinTrans(
long Window,
A_MatrixS Trans)

Set window transformation.

Parameters

(I) Window	Window index. -1 means main window
(I) Trans	Window transformation from DCS to WCS

Return

0	Success
-1	Bad window index

GetWinSize

```
long        GetWinSize(
        long        Window,
        A_PointS    *pSize)
```

Get window size.

Parameters

(I) Window	Window index. -1 means main window
(O) pSize	Size of window in DCS.

Return

0	Success
-1	Bad window index

GetBoundRect

```
long        GetBoundRect(
        long        Array,
        A_MatrixS   Matrix,
        long        bVisible
        A_RectS     *pRect)
```

Get bounding rectangle of entities.

Parameters

(I) Array	Entities
(I) Trans	Transformation of bounding box
(I) bVisible	TRUE if only thawed entities should be considered, FALSE for all entities
(O) pRect	Bounding rectangle

Return

0	Success

Undo

long Undo(
long fUndo) /*actual type is A_UndoE*/

Control of undo information

Parameters

(I) fUndo Undo control. See A_UndoE

Return

0 Success
-1 Error

GetEntType

```
long        GetEntType(
        long        Entity,
        long        *pEntityType) /*actual type is A_EntTypeE*/
```

Get entity type.

Parameters

(I) Entity Entity
(O) pEntityType Type of entity. See A_EntTypeE

Return

0 Success
-1 Entity type not recognized

V_Registry

```
long            A_V_Registry(
        BSTR            Name,
        long            nLocation,/* actual type is A_V_LocationE*/
        long            Type,/* actual type is A_V_TypeE*/
        A_V_ValueU      Value)
```

Define new JPCAD variable.

Parameters

(I) pName	Variable name (only A_MAX_NAME characters are considered)
(I) nLocation	Variable location
(I) Type	Variable type
(I) Value	Variable value

Return

>= 0	Success, variable index
-1	Error

V_UnRegistry

long V_UnRegistry(
BSTR Name)

Undefine JPCAD variable

Parameters

(I) pName Variable name (only A_MAX_NAME characters are considered)

Return

0 Success
-1 Error

V_GetIndex

long V_GetIndex(
BSTR Name)

Get index of variable.

Parameters

(I) pName Variable name (only A_MAX_NAME characters are considered)

Return

>= 0 Success, variable index
-1 Error

V_GetName

long V_GetName(
long nIndex,
BSTR Name)

Get name of variable from index.

Parameters

(I) nIndex	Variable index
(O) pName	Variable name

Return

0	Success
-1	Error

V_Get

```
long        V_Get(
    BSTR        Name,
    long        *pVarType,/* actual type is A_V_TypeE*/
    A_V_ValueU  *pVarValue)
```

Get variable value.

Parameters

(I) pName	Variable name (only A_MAX_NAME characters are considered)
(O) pVarType	Variable type
(O) pVarValue	Variable value

Return

>= 0	Success, variable index
-1	Error

V_GetByIndex

```
long            V_GetByIndex(
        long            nIndex,
        long            *pVarType, /*actual types is A_V_TypeE*/
        A_V_ValueU      *pVarValue)
```

Get variable value.

Parameters

(I) nIndex	Variable index
(O) pVarType	Variable type. See A_VAR_TypeE
(O) pVarValue	Variable value. See A_VAR_ValueU

Return

0	Success
-1	Error

V_Set

```
long        V_Set(
    BSTR        Name,
    long        VarType,/*actual type is A_V_TypeE*/
    A_V_ValueU  VarValue)
```

Set variable value

Parameters

(I) pName	Variable name (only A_MAX_NAME characters are considered)
(I) Type	Variable type
(I) Value	Variable default value

Return

>= 0	Success, variable index
-1	Error

V_SetByIndex

```
long            V_SetByIndex(
        long            nIndex,
        long            VarType,/*actual type is A_V_TypeE*/
        A_V_ValueU      VarValue)
```

Set variable value

Parameters

(I) nIndex	Variable index
(I) Type	Variable type. See A_VAR_TypeE
(I) Value	Variable default value. See A_VAR_ValueU

Return

0	Success
-1	Error

A_Alloc

```
long            A_Alloc(
        long            Type, /*actual type is A_A_TypeE*/
        long            *pArray)
```

Allocation of new array.

Parameters

(I) Type	Type of array
(O) pArray	New empty array

Return

0	Success

Description

Allocate handle for new array.

A_Free

long A_Free(
long Array)

Free array ahndle.

Parameters

(I) Array Array handle

Return

0 Success

Description

Free array associated with array handle. Call this function when you are no more need the array.

A_Length

long A_Length(
long Array)

Get number of elements in array.

Parameters

(I) Array Array handle

Return

Number of elements in array.

A_InsEnt

long A_InsEnt(
long Array,
long nIndex,
long Entity)

Insert entity to array.

Parameters

(I) Array Array handle
(I) nIndex Index of element after which the entity will be inserted. Use 0 to insert at begin and -1 to apped
(I) Entity Entity

Return

0 Success
-1 Error

A_Del

long A_Del(
long Array,
long nIndex)

Delete an element from array.

Parameters

(I) Array	Array handle
(O) nIndex	Element index

Return

0	Success
-1	Error

A_GetEnt

```
long            A_GetEnt(
        long            Array,
        long            nIndex,
        long            *pEntity)
```

Get entity from array.

Parameters

(I) Array	Array ahndle
(I) nIndex	Element index
(O) pEntity	Entity

Return

0	Success
-1	Error

X_Registry

long X_Registry(
BSTR Description,
long *pStructI)

Registry X-data structure description.

Parameters

(I) pDescription X-data description string
(O) pStructI X-data structure index

Return

0 Success
-1 Bad description

X_UnRegistry

long X_UnRegistry(
long StructI)

Unregistry X-data structure description.

Parameters

(I) StructI X-data structure description

Return

0 Success
-1 Unable to unregister (there are entities with X-data using this structure)

X_GetIndex

long X_GetIndex(
BSTR Description,
long *pStructI)

Get X-data structure description index.

Parameters

(I) pDescription X-data structure description
(O) pStructI X-data structure index

Return

0 Success
-1 unregistred structure

X_GetDesc

long X_GetDesc(
long StructI,
BSTR *pDescription)

Get X-data structure description.

Parameters

(I) StructI X-data structure index
(O) ppDescription X-data structure description

Return

0 Success
-1 Bad structure index

X_GetStruct

```
long            X_GetStruct(
        long            Entity,
        long            nIndex,
        long            *pStructI)
```

Get nth X-data structure index of entity.

Parameters

(I) Entity	Entity
(I) nIndex	Index of X-data structure (zero based)
(O) pStructI	X-data structure index

Return

0	Success
-1	Bad index

X_DeleteData

long X_DeleteData(
long Entity,
long StructI)

Remove entity X-data associated with structure index.

Parameters

(I) Entity	Entity
(I) StructI	X-data structure index

Return

0	Success
-1	Error

X_CreateData

```
long        X_CreateData(
        long        StructI,
        long        Entity,
        long        bModify,
        long        *pDataI)
```

Create/bind/copy X-data memory block.

Parameters

(I) StructI	X-data structure index
(I) Entity	Entity
(I) bModify	Modify flag
(O) pDataI	X-data data index

Return

0	Success
-1	Error

Description

Behaviour of this function depends on value of Entity and bModify:

Entity	bModification	Action
A_NO_REF	X	Create new X-data data index.This data index is not bound to any entity
an entity	FALSE	Create X-data index bound to entity X-data. Modifications will not be propagated to entity.
an entity	TRUE	Create X-data index bound to entity X-data. Modifications will be propagated to entity.

X_SetIndex

```
long        X_SetIndex(
        long        DataI,
        long        nIndex1,
        long        nIndex2,
        long        bRoot)
```

Support for array access.

Parameters

(I) DataI	X-data data index
(I) nIndex1	Item index (-1 => the same array, bRoot has no meaning)
(I) nIndex2	Array index (-1 && nIndex1 == -1 skip to parent, bRoot has no meaning)
(I) bRoot	Root flag. If TRUE, start from X-data data root

Return

0	Success
-1	Error

X_SetData

```
long        X_SetData(
        long        DataI,
        long        nIndex,
        long        Type,/* actual type is A_X_TypeE*/
        A_X_DataU   Value)
```

Set X-data data item.

Parameters

(I) DataI	X-data data index
(I) nIndex	Item index
(I) Type	Item type
(I) Value	Item value

Return

0	Success
-1	Error

X_GetData

```
long        X_GetData(
        long        DataI,
        long        nIndex
        long        *pType,/* actual type is A_X_TypeE*/
        A_X_DataU   *pValue)
```

Get X-data data item.

Parameters

(I) DataI	X-data data index
(I) nIndex	Item index
(O) pType	Item type
(O) pValue	Item value

Return

0	Success
-1	Error

X_PutData

long X_PutData(
long Entity,
long DataI)

Add/replace X-data of entity.

Parameters

(I) Entity	Entity
(i) DataI	Data index

Return

0	Success
-1	Error

X_FreeData

long X_FreeData(
long DataI)

Free X-data data index.

Parameters

(I) DataI X-data data index

Return

0 Success

G_SMulVV

double G_SMulVV(
A_VectorS V1,
A_VectorS V2)

Scalar product of two vectors.

Parameters

(I) V1 Vector (1)
(I) V2 Vector (2)

Return

Scalar product of two vectors.

G_VMulVV

double G_VMulVV(
A_VectorS V1,
A_VectorS V2)

Vector product of two vectors.

Parameters:

(I) V1 Vector (1)
(I) V2 Vector (2)

Return

Vector product of two vectors.

G_AddVV

A_VectorS G_AddVV(
A_VectorS V1,
A_VectorS V2)

Add two vectors.

Parameters

(I) V1 Vector (1)
(I) V2 Vector (2)

Return

Add two vectors.

See also

A_G_SubVV

G_SubVV

A_VectorS G_SubVV(
A_VectorS V1,
A_VectorS V2)

Subtract two vectors.

Parameters

(I) V1 Vector (1)
(I) V2 Vector (2)

Return

Subtract two vectors.

See also

A_G_AddVV

G_PerpenV

A_VectorS G_PerpenV(
A_VectorS V,
long bLeft)

Find perpendicular vector.

Parameters

(I) V Vector
(I) bLeft TRUE means counterclockwise orientation

Result

Perpendicular vector.

G_MulVR

A_VectorS G_MulVR(
A_VectorS V,
double R)

Scale vector.

Parameters

(I) V Vector
(I) R Scale factor

Result

Scaled vector.

G_NormV

A_VectorS G_NormV(
A_VectorS V)

Normalize vector.

Parameters

(I) V1 Vector

Result

Normalized vector.

G_LenV

double G_LenV(
A_VectorS V)

Length of vector.

Parameters

(I) V Vector

Return

Length of vector.

G_DistPP

double G_DistPP(
A_PointS P1,
A_PointS P2)

Distance of two points.

Parameters

(I) P1 Point (1)
(I) P2 Point (2)

Return

Distance of two points.

G_MidP

```
A_PointP        G_MidP(
        A_PointS        P1,
        A_PointS        P2)
```

Midpoint between two points.

Parameters

(I) P1	Point (1)
(I) P2	Point (2)

Result

Midpoint.

G_IntersLL

```
long        G_IntersLL(
        A_PointS    P1,
        A_VectorS   V1,
        A_PointS    P2,
        A_VectorS   V2,
        double      *pT1,
        double      *pT2)
```

Compute intersection of two lines.

Parameters

(I) P1	Point on line (1)
(I) V1	Direction of line (1)
(I) P2	Point on line (2)
(I) V2	Direction of line (2)
(O) pT1	t parameter of intersection on line (1)
(O) pT2	t parameter of intersection on line (2)

Return

0	Success
-1	No intersection found

Note

t parameter comes from parametric line equasion X = P + t * V, where P, X are points on line and V is direction of line.

G_Colinear

long G_Colinear(
A_PointS P1,
A_PointS P2,
A_PointS P3,
double Epsilon)

Are the three point colinear?

Parameters

(I) P1	Point (1)
(I) P2	Point (2)
(I) P3	Point (3)
(I) Epsilon	Accuracy

Return

TRUE	Points are colinear
FALSE	Point are not colinear

G_ParsA

```
long        G_ParsA(
        A_PointS    StartP,
        A_PointS    ArcP,
        A_PointS    EndP,
        A_PointS    CenterP,
        double      *pRadius,
        double      *pStartA,
        double      *pArcA,
        double      *pEndA,
        long        *pbClockWise)
```

Find another arc parameters.

Parameters

(I) StartP	Start point
(I) ArcP	Point on arc
(I) EndP	End point
(I) CenterP	Center of arc
(O) pRadius	Radius
(O) pStartA	Start angle
(O) pArcA	Angle of point on arc
(O) pEndA	End angle
(O) pbClockWise	Arc orientation, TRUE means clockwise

Return

0	Success
-1	Bad arc points

Note

Angles are measured for X axis counterclockwise in radians.

G_GetTA

```
long        G_GetTA(
        A_PointS    StartP,
        A_PointS    EndP,
        A_PointS    CenterP,
        long        bClockWise,
        A_PointS    P,
        double      *pT)
```

Return 't' parameter of point P on arc.

Parameters

(I) StartP	Start point
(I) EndP	End point
(I) CenterP	Center point
(I) bClockWise	Arc orientation; TRUE is clockwise
(I) P	Point
(O) *pT	t parameter

Return

0	Success
-1	Error (bad arc)

Description

Return t parameter of point P on arc. t goes from 0 (StartP) to 1 (EndP). The point P need not to lay on the arc - the point on arc is counted as intersection between circle and line sector from CenterP to P.

See also

A_G_SetTA

G_SetTA

```
long        G_SetTA(
        A_PointS    StartP,
        A_PointS    EndP,
        A_PointS    CenterP,
        long        bClockWise,
        double      T,
        A_PointS    *pP)
```

Get point with 't' parameter on arc.

(I) StartP	Start point
(I) EndP	End point
(I) CenterP	Center point
(I) bClockWise	Arc orientation; TRUE is clockwise
(I) T	t parameter
(O) *pP	Point on arc

Return

0	Success
-1	Error (bad arc)

Description

Get point with 't' parameter on arc.

See also

A_G_GetTA.

G_IntersLC

```
long        G_IntersLC(
        A_PointS    StartP,
        A_PointS    EndP,
        A_PointS    CenterP,
        double      Radius,
        double      *pTL1,
        double      *pTL2)
```

Find intersection of line and circle.

Parameters

(I) StartP	Line start point
(I) EndP	Line end point
(I) CenterP	Circle center
(I) Radius	Circle radius
(O) *pTL1	t parameter of intersection (1) on line
(O) *pTL1	t parameter of intersection (2) on line

Return

0	No intersection
1	One intersection
2	Two intersections
-1	Error (bad circle or line)

Note

See A_G_IntersLL for explanation of t parameter on line.

G_IntersCC

```
long        G_IntersCC(
    A_PointS    CenterP1,
    double      Radius1,
    A_PointS    CenterP2,
    double      Radius2,
    A_PointS    *pP1,
    A_PointS    *pP2)
```

Find intersection of two circles.

Parameters

(I) CenterP1	Center (1)
(I) Radius1	Radius (1)
(I) CenterP2	Center (2)
(I) Radius2	Radius (2)
(O) pP1	Intersection point (1)
(O) pP2	Intersection point (2)

Return

0	No intersection
1	One intersection
2	Two intersections
3	Circles are identical
-1	Error

G_Polar

A_PointS G_Polar(
double Angle,
double Radius)

Transform polar point to cartesian point.

Parameters

(I) Angle Angle in radians from X axis counterclockwise oriented
(I) Radius Distance from (0,0) point

Result

Cartesian point.

G_Angle

double G_Angle(
A_VectorS V)

Get angle of vector.

Parameters

(I) V Vector

Return

Angle of vector in radians from X axis counterclockwise.

M_Ident

A_MatrixS M_Ident(void)

Create identity matrix (identity transformation).

Return

Identity matrix.

M_Move

A_MatrixS M_Move(
A_VectorS V)

Create moving matrix (move transformation).

Parameters

(I) V Move vector

Return

Move matrix.

M_Scale

A_MatrixS M_Scale(
double Scale)

Create scaling matrix (scale transformation).

Parameters

(I) Scale Scale factor

Return

Scale matrix.

M_Rotate

A_MatrixS M_Rotate(
double Theta)

Create rotating matrix (rotate transformation).

Parameters

(I) Theta Rotation angle clockwise

Return

Rotation matrix

M_Mirror

A_Matrix M_Mirror(
A_VectorS V)

Create mirroring matrix (mirror transformation).

Parameters

(I) V Mirror line

Return

Mirror matrix

M_Inverse

long M_Inverse(
A_MatrixS M,
A_MatrixS *pInvM)

Find inverse matrix.

Parameters

(I) M	Matrix
(O) *pInvM	Result

Return

0	Success
-1	Error, inverse matrix not found

Note

You cannot use M as result matrix.

M_MulMV

A_VectorS M_MulMV(
A_MatrixS M,
A_VectorS V)

Multiply matrix and vector (apply transformation to vector).

Parameters

(I) M	Transformation matrix
(I) V	Vector

Return

Result point.

M_MulMP

```
A_PointS        M_MulMP(
        A_MatrixS       M,
        A_PointS        P)
```

Multiply matrix and point (apply transformation to point).

Parameters

(I) M	Transformation matrix
(I) P	Point

Return

Result point.

M_MulMM

```
A_MatrixS       M_MulMM(
        A_MatrixS       M1,
        A_MatrixS       M2)
```

Multiply two matrices (add two transformations).

Parameters

(I) M1	Matrix (1)
(I) M2	Matrix (2)

Return

Result matrix.

Note

Multiplying two matrices is not commutative operation.
You cannot use M1 or M2 also as result matrix.

M_Decompose

```
long        M_Decompose(
    A_MatrixS   M,
    double      Epsilon,
    double      *pScale,
    long        *pbMirror,
    double      *pAngle,
    A_VectorS   *pV)
```

Decompose uniform transformation to simple transformations (scale, mirror, rotate and move).

Parameters

(I) M	Matrix
(I) Epsilon	Epsilon use to check matrix uniformity
(O) pScale	Scale factor
(O) pbMirror	If TRUE, Transformation contains mirror
(O) pAngle	Rotaton angle
(O) pV	Move vector

Return

0	Success
-1	transformation is not uniform
-2	transformation is singular

Error

void cbError(
long nReturn)

Fired when other ACC function returned negative value.

Parameters

(I) nReturn return value

Description

ACC will fire this event when any ACC function returns negative value before this function returns to caller. This event can be used to trap errors.

Status

void Status(

long	**nStatus,**
long	**nCmdCode)**

Fired when JPCAD notifies your application about various events.

Parameters

(I) nStatus see A_StatusE
(I) nCmdCode command code. Valid only when nStatus = A_CMD_CALL

Description

Fired when JPCAD notifies your application about various events.

Drag

```
void        Drag(
        VARIANT     UserData,
        VARIANT     Point,
        VARIANT     Matrix,
        long        *nResult);
```

Parameters

(I) pUserData	User data
(I) Point	Current pointer location in WCS
(O) pMatrix	Transformation matrix
(O) nResult	0 do not draw any data (invalid position) 1 draw the data

Description

Use this function to specify transformation matrix for the current pointer location. When it is impossible to transform entities to desired shape, you can draw additional data using DrawLine and DrawArc.

Methods, Events and Constants

Methods groups:

Events

Constants

Constants

Enumerations:

A_StatusE
A_DrawMethodE
A_DisplayE
A_UndoE
A_EntTypeE
A_V_TypeE
A_V_LocationE
A_A_TypeE
A_X_TypeE

Constants:

A_ERROR_xx
A_USE_CURRENT
A_NO_REF
A_MAX_NAME
A_ATTR_xx
A_LAYER_xx
A_TEXT_xx
A_TEXTSTYLE_xx
A_GET_xx
A_A_EMPTY

Events

Status
Error
Drag

Kernel Functions

This group of functions works with JPCAD database and display:

DelEnt
Display
Draw
GetAllEnts
GetEntType
GetBoundRect
Undo

Read/write block
ReadEnt
WriteEnt

Entity Functions

These group of function will create, modify and query entities:

ARC
ARC_Make
ARC_Get
ARC_Change

ATTR
ATTR_Make
ATTR_Get
ATTR_Change

BLOCK
BLOCK_Make
BLOCK_Get
BLOCK_Change
BLOCK_GetEnt
BLOCK_Num
BLOCK_GetNth

CIRCLE
CIRCLE_Make
CIRCLE_Get
CIRCLE_Change

INSERT
INSERT_Make
INSERT_Get
INSERT_Change
INSERT_AttrGet
INSERT_AttrMake

LAYER
LAYER_Make
LAYER_Get
LAYER_Change
LAYER_GetCurrent
LAYER_SetCurrent
LAYER_GetEnt
LAYER_Num
LAYER_GetNth

LINE
LINE_Make
LINE_Get
LINE_Change

SOLID
SOLID_Make
SOLID_Get
SOLID_Change

TEXT
TEXT_Make
TEXT_Get
TEXT_Change

TEXTSTYLE
TEXTSTYLE_Make
TEXTSTYLE_Get
TEXTSTYLE_Change
TEXTSTYLE_GetCurrent
TEXTSTYLE_SetCurrent
TEXTSTYLE_GetEnt
TEXTSTYLE_Num
TEXTSTYLE_GetNth

Input/Output Functions

This group of functions implements Input/Output interaction with user through JPCAD command line interface:

Input functions:

GetEnt
GetSelection
GetString
GetLong
GetDouble
GetPoint
GetPointDrag
GetAngle
GetKWord

Output functions:

Prompt

Interface Functions

These are function that handles inteface between JPCAD and your application.

Commands

DefCmd
UnDefCmd
CallCmd

Miscelaneous Functions

Draw shape:
DrawArc
DrawLine

Device transformation
GetWinSize
GetWinTrans
SetWinTrans

GetMainWin

Array Functions

Array is set of user data. Arrays are handles into memory allocated inside JPCAD. There is no automatic garbage collection on unused arrays, you need to free all arrays explicitly. Failing to do so may cause memory problems.
You cannot copy array by simple assigment, you need to loop and add each array element to new array.

This group of functions work with arays:
A_Alloc
A_Free
A_Length
A_Del
A_GetEnt
A_InsEnt

Note:
If you store an entity reference in your program (or in array) you must take into account, that the referenced entity can be no more valid if an ERASE, UNDO, REDO and PURGE command were invoked. Entities stored in arrays are automatically reindexed when PURGE/Undo is invoked.

Variables Functions

You can define, get and set JPCAD variables using this group of functions.

Registry/Unregistry, access to index:

V_Registry
V_UnRegistry
V_GetIndex
V_GetName

Get/Set value:

V_Get
V_GetByIndex
V_Set
V_SetByIndex

Geometry Functions

Geometry functions is a set of various 2D geometry functions for handling vectors, intersections and so on. Code of those functions is located in ADK.DLL, so there is no communication overhead.

Operation with vectors:
G_SMulVV, G_VMulVV
G_AddVV, G_SubVV
G_PerpenV
G_MulVR
G_NormV
G_LenV

Operation with points on line:
G_DistPP

G_MidP
G_Colinear

Operation with points on circle:
G_GetTA, G_SetTA
G_ParsA

Intersections:
G_IntersLL
G_IntersLC
G_IntersCC

Polar - cartesian transformation:
G_Angle
G_Polar

Matrix Functions

Matrix function is set of 2D transformation matrix functions.

Matrix creation:
M_Ident
M_Move
M_Rotate
M_Mirror
M_Scale

Matrix operation:
M_MulMP
M_MulMV
M_MulMM
M_Inverse

Matrix composition/decomposition:
M_Compose
M_Decompose

X-data Functions

X-data functions is a set of functions for handling Extended entity data.

Registering X-data structures:
X_Registry
X_UnRegistry

Qyerying X-data structures:
X_GetIndex
X_GetDesc
X_GetStruct

Handling X-data on entities:
X_PutData
X_DeleteData

X-data buffer creation/deletion:
X_CreateData
X_FreeData

Set/Get X-data values:
X_GetData
X_SetData
X_SetIndex

Xdata structure description.
Xdata structure description string consists of two parts separated by semicolon. First part is only identification string for user, it is array of any character excluding semicolon. It can be empty. Second part describes xdata structure. It is array of predefined letters. Each letter defines type of data item. Data items can be separated to two groups: with constant value and with modified value. Types of constant values are:

$	variable length chunk of bytes
$<num>	fixed length chunk of bytes
s	variable length string
c	char
h	16-bit integer
i	32-bit integer
f	double
p	point/vect not transformated

Modified data items are updated by JPCAD, when it is needed. Types of modified values are:

l	double length transformated
a	double angle transformated (radians)
r	32-bit integer mirror flag
m	point transformated
v	vector transformated (not moved)
d	direction transformated (not moved or scaled)
e	index of element

There is special item array. Arrays can have fixed or variable length. One element of array can be item of any type (including array) or structure consisting of any number of items of any type (including array). The only restriction is that variable length arrays can be only on the outer most level of data structure. ‘Value’

of array is number of items in array. Arrays are described by letter '#', if number following it is fixed length array, if no number following it is variable length array. Structure is closed by currled brackets.

Example1:
My XDATA 12.3.96;#20c#{se#20a}dm

It describes xdata structure consisting of fixed 20 items length array of chars, variable length array, which item consists of string element index and fixed 20 item length array of angles, third item is direction vector and fourth item is point.

How to access xdata:
1) Obtain index of structure using Registry or GetIndex or GetStruct.
2) Obtain index of xdata using CreateData.
3) Set access indexes using SetIndex (if data you want access are in array)
4) Get or set data using GetData resp. SetData.

Example 2:
This example uses xdata structure from example 1. It sets length of variable length array to 5 and fills arrays of angles by 0.0.

```
#include <axdata.h>

static void	example2 (void)
	{
	long sindex;
	long dindex;
	A_X_DataU data;
	int i, j;

	sindex = A_X_Registry ("My XDATA 12.3.96;#20c#{se#20a}dm");
	dindex = A_X_CreateData (sindex, A_NO_REFERENCE, TRUE);
	data.Num = 5;
	A_X_SetData (dindex, A_XDATAC::ARRAY, data, 1);
	data.Double = 0.0;

	for (i = 0; i < 5; i++)
		{
		A_X_SetIndex (dindex, 1, i, TRUE);
		A_X_SetIndex (dindex, 2, 0, FALSE);

		for (j = 0; j < 20; j++)
			{
			A_X_SetIndex (dindex, -1, j, FALSE);
			A_X_SetData (dindex, A_XDATAC::ANGLE, data, 0);
			}
		}
	}
```

Selection Functions

This set of functions helps you in querying JPCAD database about specific entities with specific parameters

Query database:
S_Select

Set selection parameters:
S_Reset
S_PutLong
S_PutArray
S_PutDouble
S_PutPoint
S_PutString

Application Control Functions

This set of functions controls JPCAD applications.

Get/Set application directories:
P_GetDirs
P_SetDirs

Load/Unload application:
P_Load
P_UnLoad

Querry applications:
P_Reset
P_Next

Miscelaneous:
P_FindFile
P_SetCaption

P_GetDirs

long P_GetDirs(
BSTR *pAppDirs)

Get application directories.

Parameters

(O) ppAppDirs Application directories

Return

0 Success
-1 Error

P_SetDirs

long P_SetDirs(
BSTR AppDirs)

Set application directories.

Parameters

(I) pAppDirs Application directories

Return

0 Success

P_Load

long P_Load(
BSTR AppName)

Load JPCAD application.

Parameters

(I) pAppName Application name

Return

0 Success

P_UnLoad

long P_UnLoad(
BSTR AppName)

Unload JPCAD application.

Parameters

(I) pAppName Application name

Return

0 Success

P_Reset

long P_Reset(
long bExeApp)

Reset JPCAD application iterator.

Parameters

(I) bExeApp TRUE - get *.EXE applications, FALSE get *.DLL applications

Return

0 Success

P_Next

long P_Next(
BSTR *pAppName)

Get next JPCAD application.

Parameters

(O) ppAppName Application name

Return

1 Success
0 Fail - No more applications

P_FindFile

```
long        P_FindFile(
    BSTR        Paths,
    BSTR        Name,
    BSTR        FullName,
    long        nLength)
```

Get full file name.

Parameters

(I) pPaths	Searched paths separated by ';'. Use '.' for current directory and '$' for directory of calling process
(I) pName	name of the file. Can be fulle name or partial name
(O) pFullName	buffer that wil recieve full file name
(O) nLength	length of buffer (at least 260 bytes)

Return

0	Success
-1	File not found
-2	Buffer too small

S_Reset

long S_Reset(void)

Reset selection parameters.

Return

0 Success

S_PutLong

long S_PutLong(
long Long)

Store selection parameter

Parameters

(I) Long Long value

Return

0 Success

S_PutDouble

long S_PutDouble(
double Double)

Store selection parameter.

Parameters

(I) Double Double value

Return

0 Success

S_PutString

long S_PutString(
BSTR String)

Store selection parameter.

Parameters

(I) pString String value

Return

0 Success

S_Select

```
long        S_Select(
        long        Array,
        BSTR        String,
        long        *pArray)
```

Query JPCAD data about specific entities.

Parameters

(I) Array	Input selection or A_A_EMPTY if whole database
(I) pString	Selection parameters
(O) pArray	Found entities

Return

0	Success
-1	Error

P_SetCaption

long P_SetCaption(
BSTR Caption)

Set caption of JPCAD main window.

Parameters

(I) pCaption Caption string

Return

0 Success

GetMainWin

long GetMainWin(
long *pMainWin)

Get main window of JLPCAD.

Parameters

(O) pMainWin handle of main window of JPCAD

Return

0 success

A_ERROR_xx

This constants are general return codes from any ADK functions and they signal serious interface error conditions. The error number are less or equal to -1000 to prevent conflicts with specific function return codes.

-1000 A_ERROR_ALREADY_INITIALIZED
Interface was already initialized.

-1001 A_ERROR_NOT_INITIALIZED
ADK function was called, but intrface was not initialized.

-1002 A_ERROR_NOT_REENTRANT
ADK function was called while another ADK function call is pending.

-1003 A_ERROR_INTERNAL
Internal interface error.

-1004 A_ERROR_MSG
Message found in message queue during processing ADK command.

-1005 A_ERROR_CONNECTION_TERMINATED

Connection with JPCAD was terminated.

-1006 A_ERROR_INCOMPATIBLE_IPC_INTERFACE
Interprocess communication channel is not compatible (wrong installation of JPCAD)

-1007 A_ERROR_INCOMPATIBLE_DLL_INTERFACE
ADK(16).DLL is incompatible (you need to recompile your application for new version of ADK)

-1008 A_ERROR_THUNK
16-bit thunking error

-1009 A_ERR_VARIANT_TYPE
(only for ACC) Bad type of variant

-1010 A_ERROR_FUNCTION_REQUIRED
Function parameter required A_ADK()

0 A_ERROR_OK
No error.

A_USE_CURRENT

Use current value.
A_USE_CURRENT 0x80000003l
Valid for LineType, Width, Color, TextStyle.

A_NO_REF

No valid entity.

A_NO_REF ((long)0x80000000l)

A_MAX_NAME

A_MAX_NAME 32

Maximal length of name in database. For instance name of LAYER, TEXTYLE etc.

A_ATTR_xx

Constants for A_ATTR_xx functions.

A_ATTR_HIDDEN 0x1

Set if ATTR is invisible (hidden), cleared if ATTR is visible.

A_ATTR_PRESET 0x2

Set if ATTR value is preset (fixed), cleared if user must be prompted for ATTR value.

A_LAYER_xx

Constants used in A_LAYER_xx functions.

A_LAYER_FREEZE 0x1

Set if LAYER is freezed, cleared if LAYER is thawed.

A_LAYER_LOCK 0x2

Set if LAYER is locked, cleared if layerd is unlocked

A_LAYER_SPECIAL 0x4

??

A_LAYER_COLOR_BY ((long)0xFFFFFFFF)

Color by LAYER.

A_LAYER_LINETYPE_BY -1

Linetype by LAYER

A_TEXT_xx

Constants used in A_TEXT_xx functions.

TEXT vertical alignment:

A_TEXT_ALIGN_BOTTOM	0x0
A_TEXT_ALIGN_TOP	0x1
A_TEXT_ALIGN_BASELINE	0x2
A_TEXT_ALIGN_VERTICAL	0x3

TEXT horizontal alignment:

A_TEXT_ALIGN_LEFT	0x0
A_TEXT_ALIGN_RIGHT	0x4
A_TEXT_ALIGN_CENTER	0x8
A_TEXT_ALIGN_HORIZONTAL	0xC

A_TEXTSTYLE_xx

Constants used in A_TEXSTYLE_xx functions.
Font face:
A_TEXTSTYLE_ITALIC 0x1
A_TEXTSTYLE_UNDERLINE 0x2
A_TEXTSTYLE_STRIKEOUT 0x4

A_GET_xx

Constants used in A_Getxx functions.

A_GET_BAD_SEL 0

User selects no entity when calling A_GetEntity.

A_GET_CANCEL 1

A_Getxx function cancelled by user.

A_GET_OK 2

A_Getxx was successfull.

A_GET_DEFAULT 3

Default value was entered

A_GET_KWORD 5

Keyword was entered, A_GET_KWORD is base for the first keyword

A_A_EMPTY

An empty array.

```
A_A_EMPTY     0I
```

OLE type aliases

Due to OLE limitations, the following types are passed as VARIANTs.

ADK structures are passed as objects

ADK structure	ACC object	Visual Basic example (lower bound is zero)
A_PointS, A_VectorS	ACC.Point	Dim p As Object Set p = CreateObject("ACC.Point") p.x or p.Item(0) is point x coordinate p.y or p.Item(1) is point y coordinate
A_RectS	ACC.Rect	Dim rect As Object Set r = CreateObject("ACC.Rect") r.Min is left-bottom point r.Max is right-top point
A_MatrixS	ACC.Matrix	Dim m As Object Set m = CreateObject("ACC.Matrix") m.Item(row, col) is row, col item of matrix

ADK unions are passed as VARIANTS:

ADK type	contained data	VARIANT type
A_V_ValueU	A_V_STRING	BSTR
	A_V_DOUBLE	double
	A_V_INTEGER	long
	A_V_POINT	A_PointS
A_X_DataU	A_X_STRING	BSTR
	A_X_CHAR	char
	A_X_SHORT	short
	A_X_LONG, A_X_MIRROR, A_X_ARRAY	long
	A_X_DOUBLE, A_X_LENGTH, A_X_ANGLE	double
	A_X_POINT, A_X_POSITION, A_X_VECTOR, A_X_DIRECTION	A_PointS
	A_X_ENTITY	long

M_Compose

```
A_MatrixS    A_M_Compose(
                 double      Scale,
                 long        bMirror,
                 double      Angle,
                 A_VectorS   V)
```

Compose uniform transformation matrix.

Parameters

(I) Scale	Scale factor
(I) bMirror	If TRUE, Transformation will contain mirror
(I) Angle	Rotaton angle
(I) V	Move vector

Return

Transformation matrix

CallCmd

long A_CallCmd(
BSTR CmdName)

Call JPCAD command

Parameters

(I) pCmdName Name of the JPCAD command and optional parameters

Return

0 Success

Description

Run JPCAD commands, feed in parameters or/and options required by that command separated by \\r

Note

This command is very dangerous.
Use underlined version of commands, they are locale independent.
Do no forget to use \\r (double backslash) in place of Enter key.
Do not use this command when you run transparently.
There is no garancy, that syntax of JPCAD commands will not be changed in subsequent releases.
Do not call any commands, that require floating point values - converting them to string and back can loose precision.

S_PutArray

long S_PutArray(
long Array)

Store selection parameter

Parameters

(I) Array Array value

Return

0 Success

Note

Currently not implemented

S_PutPoint

long S_PutPoint(
A_PointS Point)

Store selection parameter.

Parameters

(I) Point Point value

Return

0 Success

E_GetLayer

```
long            A_E_GetLayer(
        long            hEntity,
        long            *phLayer)
```

Get layer of entity.

Parameters

(I) hEntity	Entity
(O) phLayer	Entity layer

Return

0	Success
-1	Error

Description

Get layer of entity.

E_SetLayer

long A_E_SetLayer(
A_EntH hEntity,
A_EntH hLayer)

Set layer of entity.

Parameters

(I) hEntity	Entity
(I) hLayer	Entity layer

Return

0	Success
-1	Error

Description

Set layer of entity.

E_GetWidth

```
long            A_E_GetWidth(
        long            hEntity,
        double          *pWidth)
```

Get width of entity.

Parameters

(I) hEntity	Entity
(O) pWidth	Entity width

Return

0	Success
-1	Error

Description

Get width of entity.

E_SetWidth

long A_E_SetWidth(
A_EntH hEntity,
double Width)

Set width of entity.

Parameters

(I) hEntity	Entity
(I) Width	Entity width

Return

0	Success
-1	Error

Description

Set width of entity.

E_GetColor

long A_E_GetColor(
A_EntH hEntity,
A_COLORREF *pColor)

Get color of entity.

Parameters

(I) hEntity	Entity
(O) pColor	Entity color

Return

0	Success
-1	Error

Description

Get color of entity.

E_SetColor

long A_E_SetColor(
long hEntity,
A_COLORREF Color)

Set color of entity.

Parameters

(I) hEntity	Entity
() Color	Entity color

Return

0	Success
-1	Error

Description

Set color of entity.

E_GetLineType

```
long            A_E_GetLineType(
        long            hEntity,
        int             *pLineType)
```

Get linetype of entity.

Parameters

(I) hEntity	Entity
(O) pLineType	Entity linetype

Return

0	Success
-1	Error

Description

Get linetype of entity.

E_SetLineType

long A_E_SetLineType(
A_EntH hEntity,
int LineType)

Set linetype of entity.

Parameters

(I) hEntity	Entity
() LineType	Entity linetype

Return

0	Success
-1	Error

Description

Set linetype of entity.

E_Trans

```
long        A_E_Trans(
    A_ArrayH    hArray,
    A_MatrixS   Trans,
    long        bCopy,
    A_ArrayH    *pArrayNew)
```

(Optional) copy entities and transformate them

Parameters

(I) hArray	Array of entities
(I) Trans	Entity transformation matrix
(I) bCopy	TRUE if copy entity before transformation, FALSE no copy
(O) pArrayNew	Array of new entities (if bCopy == TRUE). You can pass NULL if you are not interested

Return

0	Success
-1	Error

Description

(Optional) copy entities and transformate them.

General Entity Properties

These functions handles general entity properties. Following functions can applied to any entity. If the entity does not have specified property, function will do nothing and return -1.

E_GetLayer
E_SetLayer
E_GetWidth
E_SetWidth
E_GetColor
E_SetColor
E_GetLineType
E_SetLineType

Transformation/Copy

A_E_Trans