JPCAD Development Kit 1.2 Help File

JPCAD Development Kit (ADK) is a high level interface for writting applications for JPCAD. Applications for JPCAD are standalone EXE applications that are driven from JPCAD. The ADK.DLL is a dynamic link library, which allows you to communicate with JPCAD using Inter Process Communication (IPC) features of Windows. You can use ADK from any language which can call 32-bit or 16-bit DLL functions.
ADK application is standalone application, but it's life controlled by JPCAD. It means, that the application can call JPCAD only when it has a token from JPCAD to do it.

A_INSERT_AttrMake

long A_INSERT_AttrMake(

```
long            A_INSERT_AttrMake(
        A_EntH          Attr,
        A_MatrixS       Trans,
        const char      *pText,
        A_EntH          *pAttr)
```

Create attribute entity from attribute definition

Parameters

(I) Attr	Attribute definition
(I) Trans	Block transformation from LCS to WCS
(I) pText	Text of attribute
(O) pAttr	Attribute

Return

0	Success
-1	Error

A_INSERT_AttrGet

```
long            A_INSERT_AttrGet(
        A_EntH          Block,
        A_MatrixS       *pTrans,
        A_ArrayH        *pArray)
```

Get all attributes from block definition.

Parameters

(I) Block	Block entity
(O) pTrans	Transformation of block from LCS to WCS
(O) pAttr	Array of attribute entities of block

Return

0	Success
-1	Error

A_ReadEnt

```
long            A_ReadEnt(
        const char      *pFileName,
        A_EntH          *pEntity)
```

Read entity from file.

Parameters

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

Return

0	Success
nonzero	Error

The value Win32 API GetLastError() is returned in the case of error.

A_WriteEnt

```
long            A_WriteEnt(
        const char      *pFileName,
        A_EntH          Entity)
```

Write entity to file.

Parameters

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

Return

0	Success
nonzero	Error

The value Win32 API GetLastError() is returned in the case of error.

A_GetBoundRect

```
long            A_GetBoundRect(
        A_ArrayH        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

A_GetWinSize

```
long            A_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

Constants

ADK defines various constants:

A_ERROR_xx
A_USE_CURRENT, A_USE_CURRENT_ENT
A_NO_REF
A_MAX_NAME
A_ATTR_xx
A_INSERT_xx
A_LAYER_xx
A_TEXT_xx
A_TEXTSTYLE_xx
A_GET_xx
A_A_EMPTY

ADK Functions and Types

Function groups:

Interface - Handling interface between ADK application and JPCAD, defining new commands
Input/Output - JPCAD input and output functions
Kernel Management - JPCAD database and display functions
Entity Creation/Querying - Handling entities: creation, modification, querying data
General Entity Properties - Handling general entity properties like layer, color etc.
Handling Arrays - Array functions
Handling Variables - JPCAD variables
X-data - Handling JPCAD X-data
Selections - Selection sets
Application control - Handling other ADK applications

Geometry - 2D geomtery utility functions
Matrix - 2D matrix utility functions

Types - ADK types
Constants - ADK constants

Debugging

You can debug ADK application using your preffered debugger. Use the following steps to start debugging session:
* run JPCAD
* setup any breakpoints and run your ADK application as standalone EXE application. Your application will wait in A_Adk function until JPCAD loads it. You need to set current directory of your debugged program to directory where JPCAD was installed. see tips & tricks
* load your application into JPCAD using *load* command. When JPCAD loads application, it check if the application is not already running. If the application is already running, JPCAD connects to this instance. Otherwise the application is started.
* debug your application

You can debug as many applications at the same time as you want.

Notes

Because in the development process there can be lot of bugs, you will find it useful to use a process viewer utility (such as PVIEW from Visual C++ package) to check, if your application is not stuck in memory.

Compatibility & Limitations

ADK interface is compatible with Windows 3.1x, Windows 95, Windows NT

16-bit support

ADK supports 16-bit applications throught universal thunk layer (UTADK.DLL) on Win32s.
You can create source compatible 16-bit and 32-bit ADK applications. When you compile 16-bit application, you have to define A_16BIT before you include adk.h.

Compatibility with version 1.1

Programs written for ADK version 1.1 are source level compatible with version 1.2. You only need to recompile them.

Kernel Functions

This group of functions works with JPCAD database and display:

A_DelEnt
A_Display
A_Draw
A_GetAllEnts
A_GetEntType
A_GetBoundRect
A_Undo

Read/write block
A_ReadEnt
A_WriteEnt

Output Device transformation
A_GetWinSize
A_GetWinTrans
A_SetWinTrans

A_GetMainWin

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_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_ATTR_INSERTED	0x80

Set if ATTR is inserted - ATTR Value is value of ATTR, cleared when not inserted - ATTR Tag is value of ATTR

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

Constants for entity color and linetype

A_LAYER_COLOR_BY ((A_COLORREF)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. You should not call any A_Getxx functions and return control to JPCAD

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_NO_REF

No valid entity.

```
A_NO_REF       ((A_EntH)0x80000000l)
```

Entity Functions

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

ARC

A_ARC_Make
A_ARC_Get
A_ARC_Change

ATTR

A_ATTR_Make
A_ATTR_Get
A_ATTR_Change

BLOCK

A_BLOCK_Make
A_BLOCK_Get
A_BLOCK_Change
A_BLOCK_GetEnt
A_BLOCK_Num
A_BLOCK_GetNth

CIRCLE

A_CIRCLE_Make
A_CIRCLE_Get
A_CIRCLE_Change

INSERT

A_INSERT_Make
A_INSERT_Get
A_INSERT_Change
A_INSERT_AttrGet
A_INSERT_AttrMake

LAYER

A_LAYER_Make
A_LAYER_Get
A_LAYER_Change
A_LAYER_GetCurrent
A_LAYER_SetCurrent
A_LAYER_GetEnt
A_LAYER_Num
A_LAYER_GetNth

LINE

A_LINE_Make
A_LINE_Get
A_LINE_Change

SOLID

A_SOLID_Make
A_SOLID_Get
A_SOLID_Change

TEXT

A_TEXT_Make
A_TEXT_Get
A_TEXT_Change

TEXTSTYLE

A_TEXTSTYLE_Make
A_TEXTSTYLE_Get
A_TEXTSTYLE_Change
A_TEXTSTYLE_GetCurrent
A_TEXTSTYLE_SetCurrent
A_TEXTSTYLE_GetEnt
A_TEXTSTYLE_Num
A_TEXTSTYLE_GetNth

A_DelEnt

```
long        A_DelEnt(
     A_EntH       Entity)
```

Deletes an entity from database.

Parameters

(I) Entity	Entity to delete

Return

0	success
-1	Entity cannot be deleted

A_Display

```
long            A_Display(
        A_DisplayE      fDisplay)
```

Control drawing operation.

Parameters

(I) fDisplay	Operation to provide

Return

0	Success

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_Draw

long A_Draw(
A_EntH Entity,
A_DrawMethodE Method)

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.

A_GetAllEnts

```
long            A_GetAllEnts(
        A_ArrayH        *pArray)
```

Get all entities from database.

Parameters

(O) pArray	Selection set of all entities in database

Return

0	Success

A_P_GetDirs

long A_P_GetDirs(
char **ppAppDirs)

Get application directories.

Parameters

(O) ppAppDirs	Application directories

Return

0	Success
-1	Error

A_GetEntType

```
long            A_GetEntType(
        A_EntH          Entity,
        A_EntTypeE      *pEntityType)
```

Get entity type.

Parameters

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

Return

0	Success
-1	Entity type not recognized

Input/Output Functions

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

Input functions:
A_GetEnt
A_GetSelection
A_GetString
A_GetLong
A_GetDouble
A_GetPoint
A_GetPointDrag
A_GetAngle
A_GetKWord

Draw drag shape:
A_DrawArc
A_DrawLine

Output functions:
A_Prompt

Note on keywords
Most of the input command has parameter marked as pKeywords. This string can contain keywords and their abbreviations. Keywords are separated by semicolon (;) and abbreviations are separated by colon (,). For example:
"l,line;c,circle"
means two keywords (line and circle) with two abbreviations (l for line and c for circle). Whenever user enters 'l' or 'line', return value from command will be A_GET_KEYWORD, and whenever user enters 'c' or 'circle', return from command will be A_GET_KEYWORD + 1.

Note
The input functions can be called only when you receive A_CMD_CALL status from cbStatus.

When you get A_GET_CANCEL from any of the input function, you *should* not call any A_Getxx functions and return the control to JPCAD. A_GET_CANCEL informs you, that user wants to cancel current command.

A_ARC_Make

```
long            A_ARC_Make(
        A_PointS     Center,
        double       Radius,
        double       StartA,
        double       EndA,
        A_EntH       Layer,
        A_COLORREF   Color,
        long         LineType,
        double       Width,
        long         bReferenced,
        A_EntH       *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

A_ARC_Get

```
long            A_ARC_Get(
        A_EntH          Arc,
        A_PointS        *pCenter,
        double          *pRadius,
        double          *pStartA,
        double          *pEndA,
        A_EntH          *pLayer,
        A_COLORREF *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.

A_ARC_Change

```
long            A_ARC_Change(
        A_EntH          Arc,
        A_PointS        Center,
        double          Radius,
        double          StartA,
        double          EndA,
        A_EntH          Layer,
        A_COLORREF 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

A_ATTR_Make

```
long            A_ATTR_Make(
        const char      *pTag,
        const char      *pPrompt,
        const char      *pText,
        A_PointS        Point,
        double          Height,
        double          Angle,
        A_EntH          TextStyle,
        long            Align,
        A_EntH          Layer,
        A_COLORREF Color,
        long            fFlags,
        long            bReferenced,
        A_EntH          *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

A_ATTR_Get

```
long            A_ATTR_Get(
        A_EntH         Attr,
        char           pTag[A_MAX_NAME + 1],
        char           **ppPrompt,
        char           **ppText,
        A_PointS       *pPoint,
        double         *pHeight,
        double         *pAngle,
        A_EntH         *pTextStyle,
        long           *pAlign,
        A_EntH         *pLayer,
        A_COLORREF *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.

A_ATTR_Change

```
long            A_ATTR_Change(
        A_EntH          Attr,
        const char      *pTag,
        const char      *pPrompt,
        const char      *pText,
        A_PointS        Point,
        double          Height,
        double          Angle,
        A_EntH          TextStyle,
        long            Align,
        A_EntH          Layer,
        A_COLORREF      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

A_S_PutArray

long A_S_PutArray(
A_ArrayH Array)

Store selection parameter

Parameters

(I) Array	Array value

Return

0	Success

Note

Currently not implemented

A_BLOCK_Make

```
long            A_BLOCK_Make(
        A_MatrixS       Trans,
        A_MatrixS       TransInverse,
        const char      *pName,
        A_ArrayH        Array,
        A_EntH          *pBlock)
```

Create block entity.

Parameters

(I) Trans	Transformation from WCS to LCS
(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

Search Engine: PEL_QUERY

PEL_QUERY search engine enhances the Default search engine. It allows to select entities in polygon AND entities of specific types AND entities on specific layers.

Syntax

P[+][N]E[N]L

Meaning

P	Entities inside polygon
+	Entities inside and crossing polygon

Polygon is given as number of points and points (or pairs of doubles). If the polygon has zero points polygon selection has no meaning

N	is NOT operator for the following

E	Entities of specific type(s)

Types of entities is given as number of types followed by type. As type please use values from A_EntTypeE. If the number of types is zero, type selection has no meaning

L	Entities on specific layer(s)

Layers are given as number of layers followed by names of layers. If the number of layers is zero, layer selection has no meaning.

Samples

1. Select all entities on layers "FIRST" and "SECOND":

```
A_S_Reset();// reset parameters (for safety reasons)

A_S_PutLong(0);// zero number of polygon point because no polygon is used
A_S_PutLong(0);// zero number of types because no types are required
A_S_PutLong(2);// two layers
A_S_PutString("FIRST");
A_S_PutString("SECOND");

A_S_Select(A_A_EMPTY, "@<PEL_QUERY>PEL", &Selection)

A_S_Reset();// free parameters
```

2. Select all lines and circles crossing or inside a polygon that are not in layer "INVISIBLE"

```
A_S_Reset();// reset parameters (for safety reasons)

A_S_PutLong(NumberOfPolygonPoints);// put number of polygon points
...
A_S_PutPoint(Point[i]);// put polygon points
...
A_S_PutLong(2);// number of entity types
A_S_PutLong(A_ENT_LINE);// line
A_S_PutLong(A_ENT_CIRCLE);// circle

A_S_PutLong(1);// number of layers
A_S_PutString("INVISIBLE");// layer name
```

```
A_S_Select(A_A_EMPTY, "P+ENL", &Selection);

A_A_Reset();// free parameters
```

Search Engine: Default

Default search engine used when no search engine prefix is recognized. The default engine selects entities in polygon. Polygon points are specified as number of points followed by pairs of X and Y coordinates.

Syntax

{}	Entities crossing and inside a polygon
[]	Entities inside a polygon

Sample

To select all entites inside or crossing a polygon do the following

```
A_S_Reset()                  // free all previous parameters
A_S_PutLong(n)               // number of polygon vertices
...
A_S_PutDouble(Pt[n],x)       // X coordinate of n-th point
A_S_PutDouble(Pt[n].y)       // Y coordinate of n-th point
...
A_S_Select(A_A_EMPTY, "{}", &Array)

A_S_Reset()                  // free the parameters
```

A_BLOCK_Get

```
long            A_BLOCK_Get(
        A_EntH          Block,
        A_MatrixS       *pTrans,
        A_MatrixS       *pTransInverse,
        char            pName[A_MAX_NAME + 1],
        A_ArrayH        *pArray,
        long            *pReferenced)
```

Get block data.

Parameters

(I) Block	Block
(O) pTrans	Transformation from WCS to LCS
(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.

A_S_PutPoint

```
long            A_S_PutPoint(
        A_PointS        Point)
```

Store selection parameter.

Parameters

(I) Point	Point value

Return

0	Success

A_BLOCK_Change

```
long            A_BLOCK_Change(
        A_EntH          Block,
        A_MatrixS       Trans,
        A_MatrixS       TransInverse,
        const char      *pName,
        A_ArrayH        Array)
```

Change block data.

Parameters

(I) Block	Block
(I) Trans	Transformation from WCS to LCS
(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

A_BLOCK_GetEnt

long ***A_BLOCK_GetEnt(***
const char ****pName,***
A_EntH ****pBlock)***

```
long            A_BLOCK_GetEnt(
        const char      *pName,
        A_EntH          *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

A_BLOCK_Num

long ***A_BLOCK_Num(void)***

Get number of blocks.

Return

Number of blocks.

A_BLOCK_GetNth

```
long            A_BLOCK_GetNth(
        long            nIndex,
        A_EntH          *pBlock)
```

Get block ny index.

Parameters

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

Return

0	Success
-1	Error

A_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.

A_E_GetWidth

```
long        A_E_GetWidth(
      A_EntH      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.

A_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.

A_CIRCLE_Make

```
long            A_CIRCLE_Make(
        A_PointS        Center,
        double          Radius,
        A_EntH          Layer,
        A_COLORREF      Color,
        long            LineType,
        double          Width,
        long            bReferenced,
        A_EntH          *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

A_E_GetLineType

```
long        A_E_GetLineType(
      A_EntH      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.

A_E_SetColor

long A_E_SetColor(
A_EntH 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.

A_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.

A_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.

A_CIRCLE_Get

```
long            A_CIRCLE_Get(
        A_EntH          Circle,
        A_PointS        *pCenter,
        double          *pRadius,
        A_EntH          *pLayer,
        A_COLORREF      *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.

A_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.

A_E_GetLayer

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

Get layer of entity.

Parameters

(I) hEntity	Entity
(O) phLayer	Entity layer

Return

0	Success
-1	Error

Description

Get layer of entity.

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.

A_E_GetLayer
A_E_SetLayer
A_E_GetWidth
A_E_SetWidth
A_E_GetColor
A_E_SetColor
A_E_GetLineType
A_E_SetLineType

Transformation/Copy
A_E_Trans

A_CIRCLE_Change

```
long            A_CIRCLE_Change(
        A_EntH          Circle,
        A_PointS        Center,
        double          Radius,
        A_EntH          Layer,
        A_COLORREF 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

A_CallCmd

long A_CallCmd(
const char *pCmdName)

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.

A_INSERT_Make

```
long            A_INSERT_Make(
        A_EntH          Block,
        A_MatrixS       Trans,
        A_MatrixS       TransInverse,
        A_EntH          Layer,
        A_COLORREF Color,
        long            LineType,
        A_ArrayH        Array,
        long            bReferenced,
        A_EntH          *pInsert)
```

Create insert entity.

Parameters

(I) Block	Block
(I) Trans	Transformation from WCS to LCS
(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

Samples: Insert BLOCK with ATTRIButes

Description

Insert BLOCK with ATTRIButes

Code (only a fragment to create ATTRIButes, also see Insert sample

```
A_EntH Block;

.. select Block

A_MatrixS        Trans;// BLOCK local transformation
A_ArrayH         AttributesDef, Attributes;

// get attributes definitions
A_INSERT_AttrGet(Block, &Trans, &AttributesDef);
// allocate array for attributes
A_A_Alloc(A_A_ENTITY, &Attributes);

// for all attributes definitions
for(int i = 0; i < A_A_Length(AttributesDef); i++)
        {
        A_EntH AttrDef, Attr;

        // get attribute definition
        A_A_GetEnt(AttributesDef, i, &AttrDef);

        char     Value[80];
        ... get value of attribute

        // create attribute
        A_INSERT_AttrMake(AttrDef, Trans, Value, &Attr);

        // insert new attribute
        A_A_InsEnt(Attributes, -1, Attr);
        }

A_INSERT_Make(..., Attributes, ...)
A_A_Free(Attributes);
A_A_Free(AttributesDef);
```

A_INSERT_Get

```
long            A_INSERT_Get(
        A_EntH          Insert,
        A_EntH          *pBlock,
        A_MatrixS       *pTrans,
        A_MatrixS       *pTransInverse,
        A_EntH          *pLayer,
        A_COLORREF      *pColor,
        long            *pLineType,
        A_ArrayH        *pArray)
```

Get data of insert.

Parameters

(I) Insert	Insert
(O) pBlock	Block
(O) pTrans	Transformation from WCS to LCS
(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.

A_INSERT_xx

Constants for entity color and linetype

A_INSERT_COLOR_BY	((A_COLORREF)0xFFFFFFFE)

Color by INSERT.

A_INSERT_LINETYPE_BY	-2

Linetype by INSERT

Samples: String

Description

Handling variable length strings in ADK

Code fragment

```
char      *pStr;

if(A_GETOK != A_GetString("Enter a string", NULL, &pString))
          // handle it

// make a copy of string!!
char      *pString = strdup(pStr);

A_Prompt("\nYou entered: ");
A_Prompt(pString);
```

Samples: Transparent commands

Description

Define transparent command and handle reentrancy when transparent comman needs to call any of the A_Getxx functions.

Code fragment

```
// define the transparent command
A_DefCmd("transparent", 0, TRUE);


// command body
void    Transparent(void)
        {
        static BOOL     bRunning = FALSE;

        if(bRunning)
                {
                A_Prompt("\nCannot reenter command.");
                return;
                }

        bRunning = TRUE;

        ... your command code

        bRunning = FALSE;
        }
```

A_INSERT_Change

```
long            A_INSERT_Change(
        A_EntH          Insert,
        A_EntH          Block,
        A_MatrixS       Trans,
        A_MatrixS       TransInverse,
        A_EntH          Layer,
        A_COLORREF      Color,
        long            LineType,
        A_ArrayH        Array)
```

Change insert data.

Parameters

(I) Insert	Insert
(I) Block	Block
(I) Trans	Transformation from WCS to LCS
(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

Samples: Dialog box

Description

Display modal dialog box on top of JPCAD

Code fragment

```
...
HWND  hMainWindow;

A_GetMainWin(&hMainWindow);

DialogBox(hInstance, "my dialog", hMainWindow, NULL);
...
```

Samples: Dragging

Description

Drag user drawn entities

Code fragment

```
long A_CALLBACK DragLine(void FAR*Start, A_PointS End, A_MatrixS FAR*n)
        {
        A_DrawLine(*(A_PointS FAR*)Start,End);
        return 1;
        }

... code in some function
A_PointS        EndPoint, StartPoint = {0.0, 0.0};

switch(A_GetPointDrag("Select point", NULL, "", A_A_EMPTY, &StartPoint, DragLine,
&StartPoint,&EndPoint))
        {
        case A_GET_OK:
                // point locadet in EndPoint
                break;
        case A_GET_DEAFULT:
                // user pressed <enter>
                break;
        case A_GET_CANCEL:
                // command cancelled
                break;
        }
```

Samples: Selection

Description

Select multiple entities

Code fragment

```
...
A_ArrayH          Array;

switch(A_GetSelection(&Array))
            {
            case A_GET_OK:
                        // selected entities are in array 'Array'
                        break;
            case A_GET_CANCEL:
                        // command cancelled
                        break;
            };

...
// when done with selected entities -> delete array
A_A_Del(Array);
```

Samples: Insert

Description

Insert BLOCK

Code fragment

```
A_EntH Block;

... select the block to insert

A_MatrixS        Trans, InvTrans;

InvTrans = A_M_Compose(Scale, bMirror, Angle, Move);
A_M_Inverse(InvTrans, &Trans);

A_INSERT_Make(
            Block,
            Trans,
            InvTrans,
            Layer,
            Color,
            A_A_EMPTY, // see Insert BLOCK with ATTRIButes sample
            FALSE,
            &Insert))
```

A_LAYER_Make

```
long            A_LAYER_Make(
        const char      *pName,
        A_COLORREF Color,
        long            LineType,
        long            State,
        A_EntH          *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

Samples: Zoom

Description

Make zoom extents to selected entities

Code fragment

```
int     ZoomExtents (
                const A_ArrayH &Ents)
        {
        A_RectS         rect;
        A_PointS        size;
        A_MatrixS       dcs2wcs;
        double          dx, dy, ratio;

        // find bounding rectangle of selected entities
        if (A_GetBoundRect(Ents, A_M_Ident (), TRUE, &rect) < 0)
                return -1;

        // get main window transformation ...
        A_GetWinTrans (-1, &dcs2wcs);

        // .. and transform window rectagle to WCS
        rect.Min = A_M_MulMP (dcs2wcs, rect.Min);
        rect.Max = A_M_MulMP (dcs2wcs, rect.Max);

        // get size of window
        A_GetWinSize (-1, &size);

        // calculate suitable scale ratio
        dx = fabs(rect.Max.x - rect.Min.x);
        dy = fabs(rect.Max.y - rect.Min.y);
        if (dx * size.y > dy * size.x)
                ratio = dx / size.x;
        else
                ratio = dy / size.y;

        // create and set new window transformation
        dcs2wcs = A_M_Scale(ratio);
        dcs2wcs = A_M_MulMM(A_M_Move(A_G_MidP(rect.Min, Rect,Max)), dcs2wcs);
        A_SetWinTrans (-1, dcs2wcs);
        }
```

Samples: Generic ADK application

Description

This is generic ADK application portable to 32/16-bit environment.

Code

```
(please define A_16BIT in your 16-bit make file)
#include<windows.h>

#include"adk.h"

// exported functions cannot be static!
void    A_CALLBACK cbStatus(long,long);
void    A_CALLBACK cbError(long);

void    Test(void);

struct CommandS
        {
        char    *pName; // name of command
        BOOL    bTransparent;// TRUE - command can be invoked transparently
        void    (cbCmd)(void); / command function
        };

// define your commands here
CommandS        Commands[] =
        {
        {"test", FALSE, Test},
        ...
        {NULL, FALSE} // last stop
        };

#ifdef  A_16BIT
int PASCAL      WinMain(
                HINSTANCE       hInstance,
                HINSTANCE,
                LPSTR,
                int)
#else
int WINAPI      WinMain(
                HINSTANCE,
                HINSTANCE,
                LPSTR,
                int)
#endif
        {
#ifdef  A_16BIT
        long    nReturn = A_Adk(A_VERSION,
                        (A_StatusF*)MakeProcInstance(FARPROC(cbStatus), hInstance),
                        (A_ErrorF*)MakeProcInstance(FARPROC(cbError), hInstance));
#else
        long    nReturn = A_Adk(A_VERSION, cbStatus, cbError);
#endif

        if(nReturn != A_ERROR_OK)
                // for more error description see A_ERROR_xx
                MessageBox(NULL, "Test app", "Cannot initialize ADK", MB_OK);

        return 0;
        }

void    A_CALLBACK cbStatus(
                long    _Status,
                long    CmdID)
        {
        switch(A_StatusE(_Status))
                {
                case A_LOAD:
                        A_Prompt("\nLoading");
                        for(int i = 0; Commands[i].pName; i++)
                                A_DefCmd(Commands[i].pName, i, Commands[i].bTransparent);
                        break;
                case A_UNLOAD:
                        A_Prompt("\nUnloading");
                        break;
                case A_CMD_CALL:
                        Commands[CmdID].cbProcedure();
                        break;
                case A_NEW:
                        A_Prompt("\nA_NEW");
                        break;
                case A_OPEN_BEFORE:
                        A_Prompt("\nA_OPEN_BEFORE");
                        break;
                case A_OPEN_AFTER:
                        A_Prompt("\nA_OPEN_AFTER");
                        break;
                case A_SAVE_BEFORE:
                        A_Prompt("\nA_SAVE_BEFORE");
                        break;
                case A_SAVE_AFTER:
                        A_Prompt("\nA_SAVE_AFTER");
                        break;
                case A_DISCARD:
                        A_Prompt("\nA_DISCARD");
                        break;
                }
        }

void    A_CALLBACK cbError(
                long    nReturnCode)
        {
        if(nReturnCode <= -1000)
                // severe problems in communication with JPCAD, quit
                exit(0);
#ifdef  _DEBUG
        A_Prompt("Negative value return from function");
#endif
        }

static void     Test(void)
        {
```

```
A_Prompt("\nIn test command");
}
```

Samples

Please check the ADK/SAMPLES directory for aditional commented samples.

Generic ADK application sample
Define/undefine commands sample
Zoom sample
Insert BLOCK sample
Insert BLOCK with ATTRIButes sample
Entity(ies) selection sample
User defined dragging sample
Windows dialog boxes sample
Transparent commands sample
String handling sample
Invoking JPCAD command sample

A_LAYER_Get

```
long            A_LAYER_Get(
        A_EntH          Layer,
        char            pName[A_MAX_NAME + 1],
        A_COLORREF *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.

Device Coordinate System - coordinate system of output device (usually window or printer)

Local Coordinate System - local coordinate system of BLOCK

World Coordinate System - coordinate system of all entities stored in database

A_LAYER_Change

```
long           A_LAYER_Change(
       A_EntH         Layer,
       const char     *pName,
       A_COLORREF     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

A_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

A_LAYER_GetCurrent

```
long            A_LAYER_GetCurrent(
        A_EntH          *pLayer)
```

Get current layer.

Parameters

(O) pLayer	Layer

Return

0	Success
-1	Error

A_GetMainWin

```
long            A_GetMainWin(
        HWND            *pMainWin)
```

Get main window of JPCAD.

Parameters

(O) pMainWin	main window of JPCAD

Return

0	success

A_LAYER_SetCurrent

long A_LAYER_SetCurrent(
A_EntH Layer)

Set current layer

Parameters

(I) Layer	Layer

Return

0	Success
-1	Error

A_LAYER_GetEnt

```
long            A_LAYER_GetEnt(
        const char      *pName,
        A_EntH          *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

A_ErrorF

```
typedef void    (A_CALLBACK A_ErrorF)(
        long            nErrorCode);
```

See cbError.

A_LAYER_Num

long ***A_LAYER_Num(void)***

Get number of layers.

Return

Number of layers.

A_LAYER_GetNth

```
long        A_LAYER_GetNth(
      long        nIndex,
      A_EntH      *pLayer)
```

Get layer by index.

Parameters

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

Return

0	Success
-1	Error

A_StatusF

```
typedef void    (A_CALLBACK A_StatusF)(
        long            nStatus,
        long            CmdCode);
```

See cbStatus.

A_LINE_Make

```
long            A_LINE_Make(
        A_PointS        StartPoint,
        A_PointS        EndPoint,
        A_EntH          Layer,
        A_COLORREF      Color,
        long            LineType,
        double          Width,
        long            bReferenced,
        A_EntH          *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

A_P_SetCaption

```
long            A_P_SetCaption(
        const char      *pCaption)
```

Set caption of JPCAD main window.

Parameters

(I) pCaption	Caption string

Return

0	Success

A_LINE_Get

```
long            A_LINE_Get(
        A_EntH          Line,
        A_PointS        *pStartPoint,
        A_PointS        *pEndPoint,
        A_EntH          *pLayer,
        A_COLORREF      *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.

A_LINE_Change

```
long            A_LINE_Change(
        A_EntH          Line,
        A_PointS        StartPoint,
        A_PointS        EndPoint,
        A_EntH          Layer,
        A_COLORREF      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            A_SOLID_Make(
        A_PointS        Point1,
        A_PointS        Point2,
        A_PointS        Point3,
        A_PointS        Point4,
        A_EntH          Layer,
        A_COLORREF      Color,
        long            bReferenced,
        A_EntH          *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

A_SOLID_Get

```
long            A_SOLID_Get(
        A_EntH          Solid,
        A_PointS        *pPoint1,
        A_PointS        *pPoint2,
        A_PointS        *pPoint3,
        A_PointS        *pPoint4,
        A_EntH          *pLayer,
        A_COLORREF      *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.

A_SOLID_Change

```
long            A_SOLID_Change(
        A_EntH          Solid,
        A_PointS        Point1,
        A_PointS        Point2,
        A_PointS        Point3,
        A_PointS        Point4,
        A_EntH          Layer,
        A_COLORREF 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

A_TEXT_Make

```
long            A_TEXT_Make(
        const char      *pString,
        A_PointS        Point,
        double          Height,
        double          Angle,
        A_EntH          Style,
        long            Align,
        A_EntH          Layer,
        A_COLORREF Color,
        long            bReferenced,
        A_EntH          *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

A_TEXT_Get

```
long            A_TEXT_Get(
        A_EntH          Text,
        char            **ppString,
        A_PointS        *pPoint,
        double          *pHeight,
        double          *pAngle,
        A_EntH          *pStyle,
        long            *pAlign,
        A_EntH          *pLayer,
        A_COLORREF      *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.

A_TEXT_Change

```
long        A_TEXT_Change(
     A_EntH          Text,
     const char      *pString,
     A_PointS        Point,
     double          Height,
     double          Angle,
     A_EntH          Style,
     long            Align,
     A_EntH          Layer,
     A_COLORREF      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

A_TEXTSTYLE_Make

```
long            A_TEXTSTYLE_Make(
        const char      *pName,
        double          Height,
        long            Weight,
        long            Effects,
        const char      *pFont,
        long            CharSet,
        A_EntH          *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

A_TEXTSTYLE_Get

```
long            A_TEXTSTYLE_Get(
        A_EntH          TextStyle,
        char            pName[A_MAX_NAME + 1],
        double          *pHeight,
        long            *pWeight,
        long            *pEffects,
        char            pFont[A_MAX_NAME + 1],
        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.

A_TEXTSTYLE_Change

```
long            A_TEXTSTYLE_Change(
        A_EntH          TextStyle,
        const char      *pName,
        double          Height,
        long            Weight,
        long            Effects,
        const char      *pFont,
        long            CharSet)
```

Change text style data.

Parameters

(I) TextStyle	Text style
(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

Return

>= 0	Success
-1	Error

A_TEXTSTYLE_GetCurrent

```
long            A_TEXTSTYLE_GetCurrent(
        A_EntH          *pTextStyle)
```

Get current text style.

Parameters

(O) pTextStyle	Text style

Return

0	Success
-1	Error

A_TEXTSTYLE_SetCurrent

long A_TEXTSTYLE_SetCurrent(
A_EntH TextStyle)

Set current text style.

Parameters

(I) TextStyle	Text style

Return

0	Success
-1	Error

A_TEXTSTYLE_GetEnt

```
long            A_TEXTSTYLE_GetEnt(
        const char      *pName,
        A_EntH          *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

A_TEXTSTYLE_Num

long ***A_TEXTSTYLE_Num(void)***

Get number of text styles.

Return

Number of text styles.

A_TEXTSTYLE_GetNth

```
long            A_TEXTSTYLE_GetNth(
        long            nIndex,
        A_EntH          *pTextStyle)
```

Get text style by index.

Parameters

(I) nIndex	Index
(O) pTextStyle	Text style

Return

0	Success
-1	Error

A_SetWinTrans

```
long            A_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

void cbError

void cbError(
long nReturn)

Callback from ADK.

Parameters

(I) nReturn	functions return value

Description

ADK will call this function when it founds, that any other ADK function will return negative value (possible error). Because the negative value can also be result of correct operation of your program, use this function carefully.
Values of the nReturn parameter equal or less than -1000 (A_ERROR_ALREADY_INITIALIZED) means severe error in communication between your application and JPCAD and your application should terminate.

Interface Functions

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

Interface Status
A_Adk
cbStatus
cbError

Commands
A_DefCmd
A_UnDefCmd
A_CallCmd

A_S_Reset

long ***A_S_Reset(void)***

Reset selection parameters.

Return

0	Success

A_GetAngle

```
long            A_GetAngle(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        A_ArrayH        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.

A_GetEnt

```
long            A_GetEnt(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        A_EntH          *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.

A_USE_CURRENT, A_USE_CURRENT_ENT

Use current value.

A_USE_CURRENT	0x80000003l

Valid for LineType, Width, Color, TextStyle.

A_USE_CURRENT_ENT	((A_EntH)0x80000003l)

Valid for Layer.

A_GetLong

```
long            A_GetLong(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        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

A_A_EMPTY

An empty array.

```
A_A_EMPTY    0l
```

A_GetKWord

```
long            A_GetKWord(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault)
```

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

A_A_TypeE

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

Type of element array. Currently only A_A_ENTITY is supported.

A_GetPoint

```
long            A_GetPoint(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        A_PointS        *pPoint)
```

Prompt user to enter point

Parameters

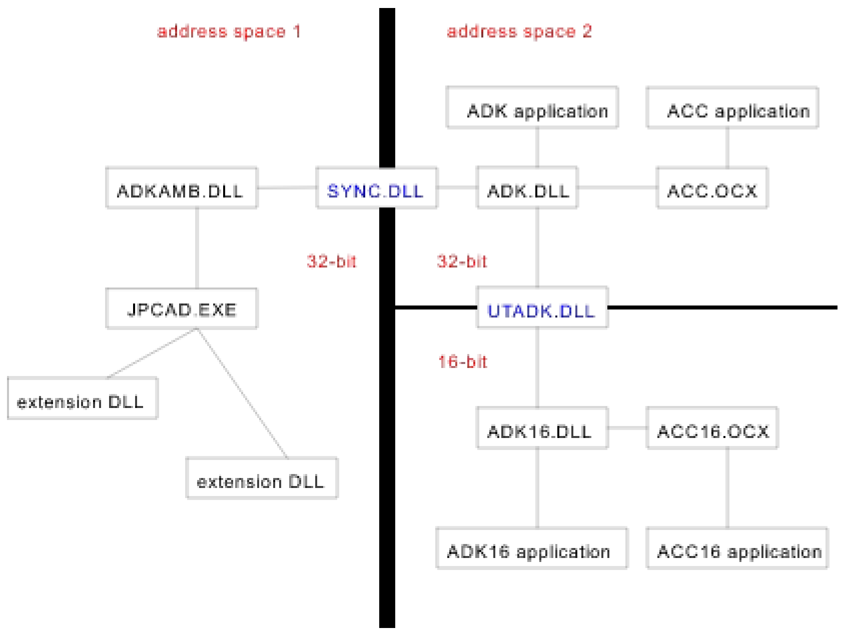
(I) pPrompt	User supplied prompt
(I) pKeywords	Keywords string
(I) pDefault	Default value
(O) pPoint	Point

Return

A_GET_OK, A_GET_DEFAULT, A_GET_KWORD, A_GET_CANCEL

Overview

Here is brief description of ADK architecture:



JPCAD.EXE	core of JPCAD
extension DLL	extension to JPCAD, like ENTITIES.DLL (defines entity), DXF.DLL (DXF import/export) and ADKAMB.DLL
ADKABM.DLL	extension to JPCAD, taking care of inter-process communication
SYNC.DLL	inter-process communication support
ADK(16).DLL	32-bit (16-bit) extension for ADK applications
ACC(16).OCX	32-bit (16-bit) OLE Control
UTADK.DLL	universal (Win32s) thunk layer
STUB32.EXE	loader of UTADK.DLL

A_GetPointDrag

```
long            A_GetPointDrag(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        A_ArrayH        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.

A_S_Select

long   ***A_S_Select(***

```
        A_ArrayH      Array,
        const char    *pString,
        A_ArrayH      *pArray)
```

Query JPCAD data about specific entities.

Parameters

(I) Array	Input selection or A_A_EMPTY if whole database
(I) pString	Selection parameters. Can be NULL to use last selection parameter
(O) pArray	Found entities

Return

0	Success
-1	Error

Note

General syntax of selection parameter is:
@<search_engine_description>search_data
where:
search_engine_prefix identifies search engine
search_data are data passed to selected search engine

Because of compatibility reasons, when no prefix is search prefix found, the default search engine is used

Implemented search engines

Default
PEL_QUERY

A_S_PutString

long A_S_PutString(
const char *pString)

Store selection parameter.

Parameters

(I) pString	String value

Return

0	Success

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.

A_GetDouble

```
long            A_GetDouble(
        const char      *pPrompt,
        const char      *pKeywords,
        const char      *pDefault,
        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

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_GetSelection

long A_GetSelection(
A_ArrayH *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.

A_S_PutDouble

long A_S_PutDouble(
double Double)

Store selection parameter.

Parameters

(I) Double	Double value

Return

0	Success

A_GetString

```
long            A_GetString(
        const char      *pPrompt,
        const char      *pDefault,
        char            **ppString)
```

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

A_S_PutLong

long A_S_PutLong(
long Long)

Store selection parameter

Parameters

(I) Long	Long value

Return

0	Success

A_Prompt

```
long          A_Prompt(
      const char    *pPrompt)
```

Display prompt.

Parameters

(I) pPrompt	Prompt

Return

0	Success

A_DrawLine

```
long            A_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 A_GetPointDrag callback function.

A_DrawArc

```
long            A_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 A_GetPointDrag callback function.

A_Undo

long ***A_Undo(***
A_UndoE ***fUndo)***

Control of undo information

Parameters

(I) fUndo	Undo control. See A_UndoE

Return

0	Success
-1	Error

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_A_Alloc
A_A_Free
A_A_Length
A_A_Del
A_A_GetEnt
A_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.

Tips & Tricks

Problem
Application dialogs are opened behind JPCAD window. JPCAD window is not disabled during command execution
Status
This is by design, when you pass NULL as parent window to MessageBox or DialogBox functions.
Solution
Use A_GetMainWin to get handle to JPCAD main window and use this handle as parent to all dialogs.

Problem
Unable to Locate DLL: Dynamic library ADK(16).DLL could not be found. This problem occurs only if you debug your program.
Status
This is by design - shared libraries ADK.DLL, SYNC.DLL, and 16-bit ADK16.DLL and UTADK.DLL are not located in System directory of Windows, but in directory of JPCAD
Solution
There are several solutions: (from most to less preffered)
1) set the current directory to directory where JPCAD is installed
2) set path to point to directory where JPCAD is located
3) copy the shared files to System directory of Windows (not recomended)

cbStatus

void ***cbStatus(***
    long ***nStatus,***
    long ***nCmdCode)***

Callback from ADK.

Parameters

(I) nStatus	Command code from JPCAD, see A_StatusE
(I) nCmdCode	command code. Valid only when nStatus = A_CMD_CALL

Description

ADK will call this function to notify your application about various status condition, that happens in JPCAD.

A_RectS

```
typedef struct
        {
        A_PointS        Min;
        A_PointS        Max;
        } A_RectS;
```

This structure holds rectangle coordinates.

A_Adk

```
long            A_Adk(
        long            Version,
        A_StatusF       *cbStatus,
        A_ErrorF        *cbError)
```

Interface to JPCAD.

Parameters

(I) Version	pass macro A_VERSION from adk.h
(I) cbStatus	address of callback status function, see cbStatus
(I) cbError	address of callback error function, see cbError. You can pass NULL

Return

A_ERROR_FUNCTION_REQUIRED	Function parameter cannot be NULL
A_ERROR_INCOMPATIBLE_DLL_INTERFACE	Incompatible version of ADK.DLL
A_ERROR_INCOMPATIBLE_IPC_INTERFACE	Incompatible version of ADKAMB.DLL
A_ERROR_CONNECTION_TERMINATED	Connection to JPCAD was unexpectedly terminated
A_ERROR_THUNK	Thunking layer error (only for 16-bit)

Description

This is the first function you have to call to establish communication with JPCAD.

Note

The cbError function should be used only in debugging process or for displaying critical errors of application

Selection Functions

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

Query database:
A_S_Select

Set selection parameters:
A_S_Reset
A_S_PutLong
A_S_PutDouble
A_S_PutString

A_DefCmd

```
long        A_DefCmd(
    const char    *pCmdName,
    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 in nCmdCode variable when ADK calls your cbStatus function
(I) bTransparent	TRUE if this command can be called transparently (possible reentrancy problem!)

Return

0	Success

Description

Define new external JPCAD command. The pCmdName string will be used to call this function. You can pass several names for the same command separated by comma (,). This is very useful for international support.

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 search order. When you undefine this function, the old one will be restored.

When your application is unloaded, JPCAD will automatically undefine all commands.

A_P_FindFile

```
long            A_P_FindFile(
        const char      *pPaths,
        const char      *pName,
        char            *pFullName,
        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

A_UnDefCmd

```
long            A_UnDefCmd(
        long            CmdCode)
```

Undefine previously defined command.

Parameters

(I) CmdCode	ID of command to undefine

Return

0	Success
-1	Command was not defined

Description

Undefine function defined by A_DefCmd.

Variables Functions

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

Registry/Unregistry, access to index:
A_V_Registry
A_V_UnRegistry
A_V_GetIndex
A_V_GetName

Get/Set value:
A_V_Get
A_V_GetByIndex
A_V_Set
A_V_SetByIndex

A_A_Alloc

```
long            A_A_Alloc(
        A_A_TypeE     Type,
        A_ArrayH      *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_A_Free

long ***A_A_Free(***
A_ArrayH ***Array)***

Free array handle.

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_V_GetIndex

long ***A_V_GetIndex(***
const char ****pName)***

Get index of variable.

Parameters

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

Return

>= 0	Success, variable index
-1	Error

A_A_Length

```
long            A_A_Length(
        A_ArrayH        Array)
```

Get number of elements in array.

Parameters

(I) Array	Array handle

Return

Number of elements in array.

A_A_InsEnt

long ***A_A_InsEnt(***
A_ArrayH ***Array,***
long ***nIndex,***
A_EntH ***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_V_GetByIndex

```
long            A_V_GetByIndex(
        long            nIndex,
        A_V_TypeE       *pVarType,
        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

A_V_GetName

long ***A_V_GetName(***
    long ***nIndex,***
    char ***pName[A_MAX_NAME + 1])***

Get name of variable from index.

Parameters

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

Return

0	Success
-1	Error

A_A_Del

```
long            A_A_Del(
        A_ArrayH        Array,
        long            nIndex)
```

Delete an element from array.

Parameters

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

Return

0	Success
-1	Error

A_P_Next

```
long        A_P_Next(
     char          **ppAppName)
```

Get next JPCAD application.

Parameters

(O) ppAppName	Application name

Return

1	Success
0	Fail - No more applications

A_A_GetEnt

long		***A_A_GetEnt(***
	A_ArrayH	***Array,***
	long	***nIndex,***
	A_EntH	****pEntity)***

Get entity from array.

Parameters

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

Return

0	Success
-1	Error

A_V_SetByIndex

```
long        A_V_SetByIndex(
      long          nIndex,
      A_V_TypeE     VarType,
      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

Types

Structures: (all the structures are DWORD aligned)

A_PointS, A_VectorS
A_RectS
A_MatrixS
A_COLORREF
A_V_ValueU
A_X_DataU

Enumerations:

A_StatusE
A_DrawMethodE
A_DisplayE
A_UndoE
A_EntTypeE
A_V_TypeE
A_V_LocationE
A_A_TypeE
A_X_TypeE

Handles:

A_ArrayH
A_EntH

Callback functions:

A_StatusF
A_ErrorF
A_GetDragF

A_V_Registry

```
long            A_V_Registry(
        const char      *pName,
        A_V_LocationE           nLocation,
        A_V_TypeE       Type,
        A_V_ValueU      Value)
```

Define new JPCAD variable.

Parameters

(I) pName	Variable name (only A_MAX_NAME characters are considered)
(I) nLocation	Variable location. See A_V_LocationE
(I) Type	Variable type. See A_V_TypeE
(I) Value	Default variable value. See A_V_ValueU

Return

>= 0	Success, variable index
-1	Error

A_P_Reset

long A_P_Reset(
long bExeApp)

Reset JPCAD application iterator.

Parameters

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

Return

0	Success

A_P_UnLoad

long A_P_UnLoad(
const char *pAppName)

Unload JPCAD application.

Parameters

(I) pAppName	Application name

Return

0	Success

A_P_Load

long A_P_Load(
const char *pAppName)

Load JPCAD application.

Parameters

(I) pAppName	Application name

Return

0	Success

A_V_UnRegistry

long A_V_UnRegistry(
const char *pName)

Undefine JPCAD variable

Parameters

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

Return

0	Success
-1	Error

A_P_SetDirs

```
long        A_P_SetDirs(
      const char    *pAppDirs)
```

Set application directories.

Parameters

(I) pAppDirs	Application directories

Return

0	Success

A_V_Get

```
long           A_V_Get(
        const char     *pName,
        A_V_TypeE      *pVarType,
        A_V_ValueU     *pVarValue)
```

Get variable value.

Parameters

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

Return

>= 0	Success, variable index
-1	Error

Application Control Functions

This set of functions controls JPCAD applications.

Get/Set application directories:
A_P_GetDirs
A_P_SetDirs

Load/Unload application:
A_P_Load
A_P_UnLoad

Querry applications:
A_P_Reset
A_P_Next

Miscelaneous:
A_P_FindFile
A_P_SetCaption

A_V_Set

```
long          A_V_Set(
        const char      *pName,
        A_V_TypeE       VarType,
        A_V_ValueU      VarValue)
```

Set variable value

Parameters

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

Return

>= 0	Success, variable index
-1	Error

A_GetWinTrans

```
long            A_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

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
	};
```

This is parameter to cbStatus function.

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
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_EntH

Variable of this type holds ID of entity. Actually entity ID is a long number. We use structure to avoid type errors.

A_ArrayH

Variable of this type holds ID of selection set. Actually entity ID is a long number. We use structure to avoid type errors. See Selections.

A_PointS, A_VectorS

```
typedef struct
        {
        double  x;
        double  y;
        } A_PointS, A_VectorS;
```

This structure holds point or vector coordinates.

A_MatrixS

```
typedef struct
        {
        double  m[3][3];
        } A_MatrixS;
```

This structure holds transfomarion matrix to define following 2D transformations: move, scale, rotation and mirror.

A_COLORREF

typedef unsigned long A_COLORREF;

This is the same as Windows COLORREF type plus special colors: A_LAYER_COLOR_BY

A_DrawMethodE

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

Defines type of color of entity for A_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_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,
        A_ENT_HATCH        = 11,
        A_ENT_HATCHSTYLE   = 12,
        A_ENT_DIMLIN       = 13,
        A_ENT_DIMANG       = 14,
        A_ENT_DIMRAD       = 15,
        A_ENT_DIMSTYLE     = 16,
        };
```

Return value from A_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 A_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_X_DataU

```
typedef struct
        {
        const void FAR* pChunk;
        long            nLength;
        } A_X_ChunkS;

typedef union
        {
        const char FAR*         pString;
        long            Long;/* also Short and Char values - no packing problems */
        double          Double;
        A_PointS        Point;
        A_EntH          Entity;
        A_X_ChunkS      Chunk;
        } A_X_DataU;
```

Holds value of X-data.

A_V_ValueU

```
struct union
          {
          const char FAR*          pString;
          double                   Double;
          long                     Integer;
          A_PointS                 Point;
          } A_V_ValueU;
```

Holds a value of Variable.

A_GetDragF

```
typedef     long       (A_CALLBACK A_GetDragF)(
            void       *pUserData,
            A_PointS   Point,
            A_MatrixS  *pMatrix);
```

Return

0	do not draw any data
1	draw the data

Parameters

(I) pUserData	Pointer to user data
(I) Point	Current pointer location in WCS
(O) pMatrix	Transformation matrix from WCS to WCS

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 A_DrawLine and A_DrawArc.

A_X_Registry

```
long            A_X_Registry(
        const char      *pDescription,
        A_X_StructH     *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

A_X_UnRegistry

```
long            A_X_UnRegistry(
        A_X_StructH   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)

A_X_GetIndex

long		***A_X_GetIndex(***
	const char	****pDescription,***
	A_X_StructH	****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

A_X_GetDesc

```
long            A_X_GetDesc(
        A_X_StructH   StructI,
        char          **ppDescription)
```

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

A_X_GetStruct

```
long           A_X_GetStruct(
        A_EntH          Entity,
        long            nIndex,
        A_X_StructH     *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

A_X_DeleteData

```
long          A_X_DeleteData(
       A_EntH        Entity,
       A_X_StructH   StructI)
```

Remove entity X-data associated with structure index.

Parameters

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

Return

0	Success
-1	Error

A_X_CreateData

long A_X_CreateData(
A_X_StructH StructI,
A_EntH Entity,
long bModify,
A_X_DataH *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	bModify	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. This data index can be used for data read only - data are copied to temporary storage
an entity	TRUE	Create X-data index bound to entity X-data. This data index can be used for both read and write.

A_X_SetIndex

```
long            A_X_SetIndex(
        A_X_DataH       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

A_X_SetData

```
long            A_X_SetData(
         A_X_DataH      DataI,
         long           nIndex,
         A_X_TypeE      Type,
         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

A_X_GetData

```
long            A_X_GetData(
        A_X_DataH       DataI,
        long            nIndex
        A_X_TypeE       *pType,
        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

A_X_PutData

```
long            A_X_PutData(
       A_EntH          Entity,
       A_X_DataH       DataI)
```

Add/replace X-data of entity.

Parameters

(I) Entity	Entity
(i) DataI	Data index

Return

0	Success
-1	Error

A_X_FreeData

long A_X_FreeData(
A_X_DataH DataI)

Free X-data data index.

Parameters

(I) DataI	X-data data index

Return

0	Success
-1	Error

A_G_SMulVV

```
double        A_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.

A_G_VMulVV

```
double          A_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.

A_G_AddVV

```
A_VectorS    A_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

A_G_SubVV

```
A_VectorS    A_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

A_G_PerpenV

```
A_VectorS       A_G_PerpenV(
        A_VectorS       V,
        long            bLeft)
```

Find perpendicular vector.

Parameters

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

Result

Perpendicular vector.

A_G_MulVR

```
A_VectorS    A_G_MulVR(
        A_VectorS    V,
        double       R)
```

Scale vector.

Parameters

(I) V	Vector
(I) R	Scale factor

Result

Scaled vector.

A_G_NormV

A_VectorS   A_G_NormV(
***   A_VectorS   V)***

Normalize vector.

Parameters

(I) V1	Vector

Result

Normalized vector.

A_G_LenV

double A_G_LenV(
A_VectorS V)

Length of vector.

Parameters

(I) V	Vector

Return

Length of vector.

A_G_DistPP

```
double          A_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.

A_G_MidP

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

Midpoint between two points.

Parameters

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

Result

Midpoint.

A_G_IntersLL

```
long            A_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.

A_G_Colinear

```
long            A_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

A_G_ParsA

```
long            A_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.

A_G_GetTA

```
long            A_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

A_G_SetTA

```
long            A_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.

A_G_IntersLC

long A_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.

A_G_IntersCC

```
long            A_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

A_G_Polar

```
A_PointS    A_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.

A_G_Angle

double A_G_Angle(
A_VectorS V)

Get angle of vector.

Parameters

(I) V	Vector

Return

Angle of vector in radians from X-axis counterclockwise.

A_M_Ident

A_MatrixS   A_M_Ident(void)

Create identity matrix (identity transformation).

Return

Identity matrix.

A_M_Move

A_MatrixS A_M_Move(
A_VectorS V)

Create moving matrix (move transformation).

Parameters

(I) V	Move vector

Return

Move matrix.

A_M_Scale

A_MatrixS A_M_Scale(
double Scale)

Create scaling matrix (scale transformation).

Parameters

(I) Scale	Scale factor

Return

Scale matrix.

A_M_Rotate

```
A_MatrixS    A_M_Rotate(
      double       Theta)
```

Create rotating matrix (rotate transformation).

Parameters

(I) Theta	Rotation angle clockwise

Return

Rotation matrix

A_M_Mirror

A_Matrix A_M_Mirror(
A_VectorS V)

Create mirroring matrix (mirror transformation).

Parameters

(I) V	Mirror line

Return

Mirror matrix

A_M_Inverse

```
long            A_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.

A_M_MulMV

```
A_VectorS    A_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.

A_M_MulMP

```
A_PointS      A_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.

A_M_MulMM

```
A_MatrixS       A_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.

A_M_Decompose

```
long            A_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

Samples: Define and undefine commands

Description

This sample will define new command called "_test" and its locale name "test". When you run this command from JPCAD, new command called "test2" will be defined or undefined.

Code fragment

```
// place this code in initialization code
A_DefCmd("_test,test", 0, FALSE);

// code to handle "test" command
void	Test(void)
	{
	static int		Test2CommandID = -1;

	if(Test2CommandID >= 0)
		{
		A_UnDefCmd(Test2CommandID);
		Test2CommandID = -1;
		}
	else
		{
		Test2CommandID = 10;
		A_DefCmd("test2", Test2CommandID, FALSE);
		}
	}
```

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:
A_G_SMulVV, A_G_VMulVV
A_G_AddVV, A_G_SubVV
A_G_PerpenV
A_G_MulVR
A_G_NormV
A_G_LenV

Operation with points on line:
A_G_DistPP
A_G_MidP
A_G_Colinear

Operation with points on circle:
A_G_GetTA, A_G_SetTA
A_G_ParsA

Intersections:
A_G_IntersLL
A_G_IntersLC
A_G_IntersCC

Polar - cartesian transformation:
A_G_Angle
A_G_Polar

Matrix functions

Matrix function is set of 2D transformation matrix functions.

Matrix creation:
A_M_Ident
A_M_Move
A_M_Rotate
A_M_Mirror
A_M_Scale

Matrix operation:
A_M_MulMP
A_M_MulMV
A_M_MulMM
A_M_Inverse

Matrix composition/decomposition:
A_M_Compose
A_M_Decompose

X-data functions

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

Registering X-data structures:
A_X_Registry
A_X_UnRegistry

Querying X-data structures:
A_X_GetIndex
A_X_GetDesc
A_X_GetStruct

Handling X-data on entities:
A_X_PutData
A_X_DeleteData

X-data buffer creation/deletion:
A_X_CreateData
A_X_FreeData

Set/Get X-data values:
A_X_GetData
A_X_SetData
A_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)
        {
        A_X_StructH sindex;
        A_X_DataH 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);
                        }
                }
        }
```

Samples: Invoking JPCAD command

Description

This sample will invoke zoom/extents command.

Code fragment

```
// uses locale independent name of command and command option
A_CallCmd("_zoom\\r_e\\r");
```