README.ADS - February 16, 1993

****

AutoCAD Release 12 for Windows "Read Me" for
AutoCAD Development System (ADS)

****

This document contains last minute information that could not be included in the normal AutoCAD documentation regarding programming for the AutoCAD Development System (ADS).

==========

Contents

==========

==========

1.0 - ADS for Windows Files

1.1 - ADS

The following is a list of the Windows ADS files that may be useful while building a list for a programmer's workbench (or IDE) for the various compilers supported for ADS Windows development:

winads.lib	ADS library for WATCOM C9.01d/386 (32-bit).

winadshc.lib	ADS library for MetaWare High C/C++ 3.0 (32-bit).
winadsc6.lib	ADS library for Microsoft C 6.0.
winadsc7.lib	ADS library for Microsoft C/C++ 7.0 and Microsoft Quick C for Windows.
winadsbc.lib	ADS library for Borland C++ 3.1 or Turbo C++ 3.0.
ddeml.lib	For Microsoft's DDEML.DLL--DDE is used to start the link to ADS apps. SendMessage is used thereafter for faster data transfer.
winads.c	Source for WinMain--allows special customization.
adslib.h	Standard ADS header file.
ads.h	Standard ADS header file.
adscodes.h	Standard ADS header file.
adsdlg.h	Standard ADS header file.
ol_errno.h	Standard ADS header file.
windde.h	Special Windows header file for convenient globals.
ddeml.h	Microsoft header file for making calls to ddeml.lib.
shellapi.h	Microsoft header file for making calls to shell.lib (Registration Database).

1.2 - ADS\WIN

For Tool Palette Example Program:

atoolbox.c	Sample ADS program for toolboxes.
atoolbox.def	Link definition file.
atoolbox.rc	Resources such as bitmaps, strings.

For DDE Example Program:

shell.lib	For calls to Windows Registration Database
winutil.c	Low level Windows utilities.
ddewin.c	DDE Manager. See "dde.txt".
spreadsh.c	Spreadsheet transfer functions.
ddeconv.c	AutoCAD Entity Conversion.
ddeads.c	Edit, DDE Menu.
ddeads.rc	Resource file.
ddeads.def	Link definitions.

1.3 - Resource and Definition Files

*.rc	Resource files that define resources needed, such as icons, dialogs, strings, etc.
*.def	Link definition files that define exported functions, attributes such as "movable", starting heap size, etc.

There are small files required by Windows ADS programs that are not required by other platforms: the .def and .rc files. These files are normally placed in the win subdirectory of the ADS directory. These are required to enable the internal message passing from AutoLISP to ADS, as well as to specify resources and link options. When creating new ADS programs, copy one of the existing .def and .rc files to your new program name:

```
copy sqr.def myname.def
copy sqr.rc  myname.rc
```

Then edit the new files and change the names. In the .def file, change the comment, the NAME, and the DESCRIPTION. In the .rc file, change the name in the STRINGTABLE. If you want your ADS program to have an icon, put its name in the .rc file, include "windde.h" in your C program, and add these lines in your main():

```
/* Display ADS window as icon */
ShowWindow(adsw_hWnd, SW_SHOWMINNOACTIVE);
UpdateWindow(adsw_hWnd);
```

The only required resource is the program name in the STRINGTABLE. This name is used for the Window class name, title, etc. Even if an ADS program doesn't have a visible window, an invisible window is used for message passing. Many sample .rc files contain an inclusion of an icon that is not necessary for invisible ADS applications. That line can be removed.

For example, for your application you can take SQR.DEF and change the following items:

```
;-----------------------------------
; SQR.DEF module definition file                    <------
;-----------------------------------

NAME            SQR                                 <------

DESCRIPTION'ADS Square Root'                <------
EXETYPE         WINDOWS
STUB                    'WINSTUB.EXE'
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD MOVEABLE
HEAPSIZE        100
STACKSIZE               8192

EXPORTS         adsw_win_proc     @1
```

```
                        adsw_dde_callback @2
```

Next, take the SQR.RC and in your application, change the following items:

```
/*---------------------------
  SQR.RC resource script                          <------
 ---------------------------*/

#include "windows.h"

SQR ICON ads.ico                                  <------
STRINGTABLE
BEGIN
   1 ,"sqr"                                       <------
END
```

1.4 - WINADS.C

The module, winads.c, is provided to allow ADS developers complete control of the Windows message loop, message procedure, DDE initialization, and so on. For generic ADS applications that aren't supporting any special Windows features this module does not have to be altered.

The object file "WINADS.OBJ" (NOTE: the exact name is dependent on your compiler) is built by compiling winads.c. This requires the Windows Software Developers Kit. To assist you, pre-compiled objects for various supported compilers are provided:

winads.obj	WATCOM C9.01d/386
winadshc.obj	MetaWare High C/C++ 3.0
winadsbc.obj	Borland C++ 3.1 or Turbo C++ 3.0
winadsc6.obj	Microsoft C 6.0
winadsc7.obj	Microsoft C/C++ 7.0 or Quick C for Windows

1.5 - ADS_MAIN

WINADS.C provides the function WinMain required by Windows applications. WinMain then calls your main(). However, some Windows compiler/linkers didn't work with a main(), so your main is redefined to ads_main by adslib.h if WIN is defined to 1. Therefore, an error referring to a missing ads_main indicates that WIN is not defined. WIN is defined in the sample make files, and in the integrated development instructions below. All of this is to allow platform independent ADS programs to be built on Windows without source changes while also allowing full access to the Windows startup code.

1.6 - DDE Server Mode

You can use the adsw_subclass_dde function to install your own DDE message handler as described in the Using AutoCAD for Windows manual. The DDE application name the other application would use should be the name of your ADS application (defined in the .rc file STRINGTABLE). You can use any topic name you like. See ads\win\DDEBLANK.C for an example (tested with MS C/C++ 7.0).

2.0 - ADS Compilers

Some of the ADS make files supplied in the AutoCAD for Windows distribution use the -t and -30 switches with the RC.EXE resource compiler. These switches support Windows 3.0; however, AutoCAD R12 for Windows requires Windows version 3.1. Please understand that these switches are superfluous and do not hurt anything.

2.1 - Microsoft C/C++ 7.0

Microsoft C/C++ 7.0 requires a host that provides DPMI services. Examples of such hosts are a DOS session under Windows enhanced mode and Qualitas' 386MAX on DOS. If you don't have enough environment space in your Windows DOS session then you will need to increase it by adding /e:xxxx to the Optional Parameters of DOSPRMPT.PIF, where xxxx is bytes.

To build the sample ADS applications in the ADS directory, cd into the directory and modify the batch file MS7MAKE.BAT to set the environment variable MSC7 to point to wherever you installed Microsoft C/C++ 7.0, then execute the following:

```
ms7make
```

To build the Windows sample ADS applications in the ADS\WIN directory, cd into that directory, make sure your compiler environment is set up (PATH, LIB, and INCLUDE), and do the following:

```
nmake -f dde
```

To build one ads sample program, such as gravity.exe, enter "ms7make gravity". Special Microsoft C7 files are:

ms7make.bat - Batch file to build Microsoft C/C++ 7.0 ADS programs.

Usage:

"ms7make gravity".

winads.ms7 - Microsoft C/C++ 7.0 nmake file

Usage:

"nmake -f winads.ms7".

winadsc7.lib - Microsoft C/C++ 7.0 ADS library
winadsc7.obj - Created during build. C7 includes the Windows support files (headers, libraries, etc.).

win\dde - Makefile for building DDE, WINBLANK, and ATOOLBOX sample applications.

Usage: "nmake -f dde".

2.2 - Examples of How to Use the PWB (Programmers Workbench)

2.2.1 - ATOOLBOX

Set up the C7 directories in the PATH, INCLUDE, and LIB environment variables. Run Windows and start up the PWB (Programmer's Workbench). Assuming your ADS files are in \ads and your Windows ADS application source files are in ads\win:

Project

New Project

Enter "\ads\win\atoolbox"

Create? Yes.

Set Project Template (Dialog)

Runtime Support

C

Template

Windows 3.1 Exe

Edit Project (Dialog)

<Enter the following file names>

atoolbox.c
atoolbox.rc
atoolbox.def
..\winadsc7.lib
..\winads.c
..\ddeml.lib

Save List

Options

Language Options

C Compiler Options

Model

Large

Processor
80286

Additional Global Options
Defines
Add ", WIN=1"
Additional Includes
Add ".."

Additional Options
Add to list: " /Gw"
Check: Release Options
Additional Release Options
Check: "Inline 80x87 Instructions"
Check: Debug Options
Additional Debug Options
Check: "Inline 80x87 Instructions"

Link Options
Check: Release Options
Additional Release Libraries
Change LLIBC7W to LLIBCEW
Check: Debug Options
Additional Debug Libraries
Change LLIBC7W to LLIBCEW

Build Options
Use Debug Options <switch on for building debug version>

Project
Rebuild All

Copy the resulting \ads\win\atoolbox.exe to your \winacad directory to be xload'ed from AutoCAD. You can switch back and forth from the PWB and a DOS box or Windows with Alt-Tab. Hold down the Alt key and press Tab until the program you want to use appears. Then let up on the Alt key. To debug, set up a Program Manager Icon that starts CodeView and ATOOLBOX.

d:\c700\bin\cvw d:\winacad\atoolbox.exe

Start up the debugger and set a break point in the atoolbox source. "xload atoolbox" from AutoCAD.

2.2.2 - DDEADS

Same except for the file list:

ddeads.c
ddeads.rc
ddeads.def
spreadsh.c
ddeconv.c
ddewin.c
winutil.c
..\winadsc7.lib
..\winads.c
..\ddeml.lib
..\shell.lib

2.3 - Microsoft C 6.0

To build the sample ADS applications in the ADS directory,
cd into the directory and modify the batch file winmake.bat to set
the environment variable MSC6 to point to wherever you installed
Microsoft C 6.0. Note that the environment variables LIB and INCLUDE
assume the Windows SDK libraries and include files have been installed
in MSC6\wlib and MSC6\winc. Your setup may vary, in which case you
will need to modify LIB and INCLUDE in winmake.bat.

The make file used in winmake.bat, winads.mak, assumes you are
using version 6.00AX of Microsoft C, which supports the compiler flag
-EM which allows compilations of larger modules. If you do not have version
6.00AX and are not compiling large modules then you may remove the -EM
switch from winads.mak.

If, when linking, you find you are missing llibcew.lib, then you
need to install the Windows large model library from the Windows SDK.

From the ADS directory execute the following:

winmake

To build one ADS sample program, such as gravity.exe, enter
"winmake gravity". Special Microsoft C 6.0 files are:

winmake.bat - Batch file to build Microsoft C 6.0 ADS programs
winads.mak - Microsoft C nmk file
winadsc6.lib - Microsoft C 6.0 ADS library
winadsc6.obj - Microsoft C 6.0 ADS object file

The 6.00 "AX" version makes use of extended memory. Without

this version, some modules might produce an error such as "Out of far heap space." The binary files (cl.exe, etc.) were on the last disk of a Microsoft C 6.00 product update. However, even without the AX version, using Microsoft's NMK instead of NMAKE makes it possible to compile all the ADS modules provided with AutoCAD.

2.4 - Microsoft QuickC for Windows

Here's how to build the ADS program, SQR (single words are menu items or push buttons in QCWIN). Assuming \ADS is your ADS directory:

Copy ADS\WIN\ads.ico to the ADS directory.

Start Windows and start QCWIN.

Project
- Open
- Use dialog to change to ADS directory, type "SQR.MAK".
- Say Yes to "Do you want a new project?".
- In the next dialog ("Edit"), add files by double clicking:

    - winadsc7.lib
    - ddeml.lib
    - win\sqr.rc
    - win\sqr.def
    - winads.c
    - sqr.c
    - ads.ico

Options
- Project
    - Select "Windows Exe"
    - Compiler
        - Memory Model: "Large"
        - Warning level: 2
        - Options >>
            - Add to Define: "WIN=1"
    - Linker
        - For normal CodeView (not Quick C debugger),
            - under Debug Options, check "CV 3.x Format".
        - Turn OFF extended dictionary
    - Resource
        - Disable load optimization
- Directories
    - Include
        - Add ;ADS

If you get "undefined" errors for the following variables, add the definitions to the C source file or a header file. These are in the Windows 3.1 SDK windows.h file.

```
#define HWND_TOP = ((HWND)0)
#define HINSTANCE_ERROR ((HINSTANCE)32)
```

To build the DDE sample programs in the ads\win directory (such as ddeads.exe), it is necessary to use the Windows SDK 3.1 windows.h. Use the QCWIN Options, Directories dialog to set the SDK include and lib directories at the start of the lists.

```
\windev\include;\qcwin\include;\ads
\windev\lib;\qcwin\lib;\ads
```

Also, in ads\win\ddewin.c, the QCWIN compiler needs a function prototype for the function findRegValue. Just copy the function definition to the top of the file after debugPrintf and add a semicolon:

After...

```
int debugPrintf(char *format, ...);
```

Insert...

```
static int findRegValue(char *key1, char *key2, char *key3, char *key4,
                        char *szValue);
```

This function was added in release 12 to find spreadsheet application information using the Windows Registration Database.

2.5 - Borland C++ 3.1

Be sure your Borland C++ 3.1 compiler environment is set up, then enter "bcmake" to use the make file winads.bc.

bcmake.bat - Batch file to use to build Borland C++ ADS programs
winads.bc - Borland make file for 3rd party developers
winadsbc.lib - Borland ADS library
winadsbc.obj - Borland ADS object file

2.6 - Turbo C++ for Windows 3.0

Turbo C for Windows uses the Borland ADS library, WINADSBC.LIB.
Using the Turbo C++ IDE (Integrated Development Environment), this example will build ATOOLBOX.EXE (modeless icon tool palette) and DDEADS.EXE (functions invoked by the DDE menu used to send AutoCAD data to a spreadsheet). This script assumes your C files are one directory below the ads

directory, such as "ads\win".

To build your own program, just replace "atoolbox" or "ddeads" (if you're using DDE) with your program name. Words starting with uppercase below are menu items or push buttons.

2.6.1 - Build ATOOLBOX

<Run Turbo C++ for Windows>
Project-Open <Use file dialog to move to your source directory, such as ads\win.>
Enter: atoolbox <Enter>

Options-Compiler-Code Generation-Model
Large
Defines
WIN=1
Options-Directories-Include
<Add to end of line> ;..
Library
<Add to end of line> ;..
Project-Add Item...

<Enter "*.C"<Enter> to see the C files>

<Add files with the "Add" button>
atoolbox.c
..\winads.c
<Enter file names>
atoolbox.rc
atoolbox.def
..\ddeml.lib
..\winadsbc.lib

<Press "Done" push button>
Project-Close Project (save your work)
Project-Open
Click on atoolbox
Compile-Build All

If you have problems with the Turbo C++ IDE not binding resources to EXE(s), you can run a batch file from DOS:

RCTOOL.BAT
```
rc -tk atoolbox.rc atoolbox.exe
copy atoolbox.exe \winacad
```

Or you can use the Borland Resource Workshop to save a .res

file.

2.6.2 - Build DDEADS

<Run Turbo C++ for Windows>
Project-Open <Use file dialog to move to your source directory>
Enter: ddeads <Enter>

Options-Compiler-Code Generation-Model
Large
Options
Pre-compiled Headers
Unsigned characters
Defines
WIN=1
Options-Directories-Include
<Add to end of line> ;..
Library
<Add to end of line> ;..
Project-Add Item...

<Enter "*.C"<Enter> to see the C files>

<Add files with the "Add" button>
ddeads.c
ddewin.c
spreadsh.c
ddeconv.c
winutil.c
..\winads.c
<Enter file names>
ddeads.rc
ddeads.def
..\ddeml.lib
..\shell.lib
..\winadsbc.lib

<Press "Done" push button>
Project-Close Project (save your work)
Project-Open
Click on ddeads
Compile-Build All

RCDDE.BAT

```
rc -tk ddeads.rc ddeads.exe
copy ddeads.exe \winacad
```

2.7 - WATCOM C9.01d/386

To build 32-bit ADS applications for AutoCAD for Windows using WATCOM C use the following version:

WATCOM C9.01d/386 or greater, which supports the optimization switch -op.

To determine what version of the compiler you have, run wcc386p.exe and the first line will tell you what compiler level you have.

Run the batch file TO31.BAT in the WATCOM directory to turn on Windows 3.1 enhanced 32-bit support.

Currently, DDE ADS sample applications cannot be compiled by the 32bit WATCOM compiler.

One thing you should know, if you are using any of the Windows 3.1 API extensions, such as DDE, COMMDLG, and the like, you need to place the correct
#define statement before the inclusion of the windows.h header file. For example, to include DDE API functions:

```
#define INCLUDE_DDEML_H
...
#include <windows.h>
...
```

This tells WATCOM to include the header file ddeml.h within the compiler header files. You must *not* do an explicit include of ddeml.h in your source. This step is only required if you are writing your own code that uses the 3.1 functionality such as DDEML calls. It does not apply to Windows 3.0 calls nor calls to the ADS library.

To build the sample ADS applications in the ADS directory,
cd into the directory and do the following:

Either set up the WATCOM environment variables -

WATCOM
INCLUDE
LIB
PATH

or set the WATCOM variable in the .BEFORE section of the wmake file MAKESAMP.WIN to point to your WATCOM compiler path.

Initiate the make with:

\WCPATH\binb\wmake /f makesamp.win

where WCPATH is replaced with your WATCOM path (unless your PATH is already set up to point to .\binb).

When linking the applications you will get a warning message complaining about redefinitions of malloc and free for clib3s.lib. This is normal.

To build one ads sample program, such as gravity.exe, enter "\WCPATH\binb\wmake /f makesamp.win gravity".

The WATCOM C9.01d/386 ADS files are:

makesamp.win - WATCOM C9.01d/386 wmake file.
winads.lib - WATCOM C9.01d/386 ADS library.
winads.obj - Created during build.

To compile C source code, invoke the wcc386p command. Compiling an ADS application requires you to specify certain options. For example:

wcc386p foo -DWIN -DWATWIN -zq -s -j -fpi87 -3s -zW -opmaxet

In this example, the options are:

foo	The name of the source file. You don't have to enter the filename extension (.c).
-DWIN	#define WIN 1, for turning on Windows-specific code
-DWATWIN	#define WATWIN 1, for turning on WATCOM 32-bit Windows specific code.
-zq	Causes the compiler to operate quietly.
-s	Removes stack overflow checks.
-j	Signed default char type.
-fpi87	Generates in-line 80387 instructions.
-3s	Generates 80386 instructions using stack-based argument passing conventions.
-zW	Microsoft Windows compatible code sequences.
-opmaxet	Optimize for speed. Note that the optimization switch

-op is used to generate Microsoft compatible floating point code.

Link your application using the WATCOM linker, wlinkp.exe. Here's a sample link file, foo.lnk:

```
sys win386
debug all               # needed if debug info is required
name foo
option stack=10K
option maxdata=1K
option quiet
library winads
file winads.obj
file foo.obj
```

Please note that the library winads.lib should be included before the runtime libraries during linking so that the math CPU precision is properly set. (8087cw.obj, normally retrieved from the runtime libraries, is included in winads.lib so that the 80X87 control word is set to the desired precision, PC_64.)

Here's the link command:

```
wlinkp @foo.lnk
```

The next step is to bind the Windows resource file to the Windows executable:

```
wbind foo -R -I.\win win\foo.rc foo.exe
```

This assumes the files foo.def and foo.rc are in a directory called .\win off of your working directory.

You now have a Windows ADS executable file called foo.exe. You can clean up the intermediate working files now if you wish:

```
del foo.rex
del win\foo.res
```

2.8 - MetaWare High C/C++ 3.0

IMPORTANT: Before you begin you MUST acquire two updates from MetaWare.

The first update provides 32-bit DDEML support for Windows. The update should include the files

```
mwsup.exe           mwsupdll.dll
mwdllsup.exe        initwin.obj
```

mwlibw.lib ddeml.h
mwsupdll.lib

The mwsupdll.dll file will need to be supplied to the end user by you where it must be installed in either the Windows directory with all the other .DLLs or somewhere along the user's PATH.

The second update is a patch to hc386.lib that fixes a bug where two different calls to fopen(), when used with the ADK, can generate identical file handles so that the files write into each other.

The sample make file that is supplied, mw30samp.win, uses the Phar Lap linker 386|LINK to build the 32-bit executable which is bound to MetaWare's MWSUP.EXE with the MWBIND utility. If you do not have 386|LINK nor intend to buy it then you'll need to modify the make file to use your linker.

MetaWare provides a README.ADS in their distribution. Please note that their readme document is for AutoCAD 386 and not AutoCAD for Windows.

To build the sample ADS applications in the ADS directory, cd into the directory and do the following:

Modify mw30samp.bat to set the environment variable HIGHC to wherever you installed High C/C++ 3.0.

Execute the batch file mw30samp.bat

To build one ads sample program, such as gravity.exe, enter "mw30samp gravity". Special High C/C++ 3.0 files are:

mw30samp.bat - Batch file to build High C/C++ 3.0 ADS programs.
mw30samp.win - High C/C++ 3.0 mwmake file.
winadshc.lib - High C/C++ 3.0 ADS library.
winadshc.obj - Created during build. (Also provided in the product distribution).

To compile C source code, invoke the hc386 command. Compiling an ADS application requires you to specify certain options. For example:

hc386 foo.c -f387 -O3 -c -Hwin -DWIN -DHCWIN -Hoff=Align_members

In this example, the options are:

-f387 Generate in-line 80387 code.

-O3 Optimization level 3.

-c Compile only (suppress invocation of the linker).

-Hwin Used in conjunction with the MetaWare 32-bit Windows ADK.

-DWIN #define WIN 1, for turning on Windows-specific code

-DHCWIN #define HCWIN 1, for turning on High C/C++ 3.0 32-bit Windows ADK-specific code.

-Hoff=Align_members
No padding in structures.

Link your application using the Phar Lap linker, 386link.exe. Here's a sample link file, foo.lnk:

```
\highc\small\initwin.obj
foo.obj
winadshc.obj
-lib winadshc.lib \highc\small\mwlibw \highc\small\hc386
        \highc\small\hcna \highc\small\hc387
-exe foo
```

Here's the link command:

```
386link @foo.lnk
```

Next, assuming the resource file is in a directory called .\win off of the current working directory, compile foo.rc into a resource file:

```
rc -I.\win -r win\foo.rc
```

Copy mwsup.exe into the working directory and bind the application resource file to it:

```
copy \highc\bin\mwsup.exe
rc -fe mwsup.exe win\foo.res
```

Combine the modified mwsup.exe and foo.exp into foo.exe:

```
mwbind mwsup.exe foo.exp foo.exe
```

By default mwbind sets the heap to 2,048K. This is memory reserved during initialization of your program; it is freed only after the program terminates. Use MetaWare's memory-configuration utility mwmem.exe to change the value of your ADS application's heap. For example, to change the amount of heap available for myadsapp.exe to 512K:

```
mwmem myadsapp.exe 512000
```

The only way to tell how much heap space a program needs is to test it configured with various amounts of space reserved.

You now have a Windows ADS executable file. You can clean up the intermediate working files now if you wish:

```
del foo.exp
del win\foo.res
del mwsup.exe
```