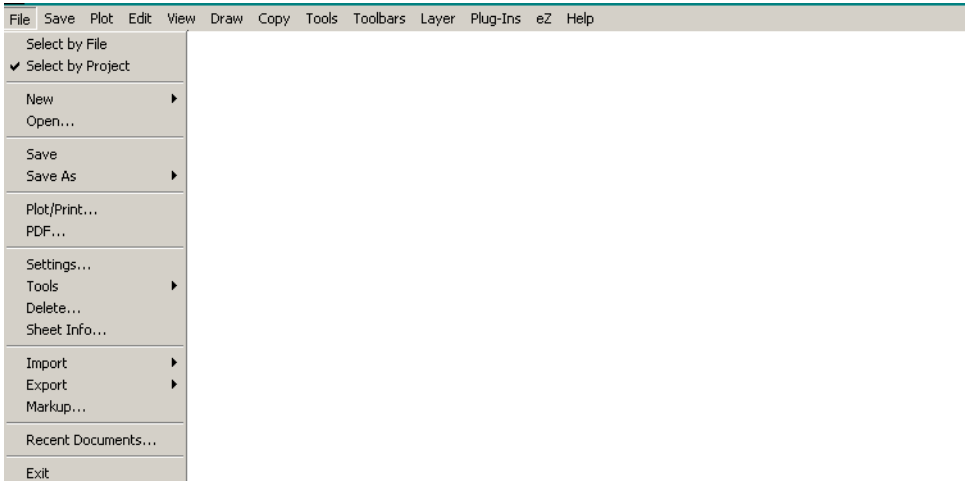


# WINDOWS MENU BAR



On Microsoft Windows platforms, **ARRIS** has a standard *Windows Menu Bar* across the top of the **ARRIS** Window. This bar allows selection of many of the functions of **ARRIS** from a standard MS Windows interface. The Windows Menu Bar duplicates most of the functions found on the **ARRIS Desktop** menu.



## Enable / Disable

The Windows Menu Bar may be turned on or off in the **Preferences** menu. Refer to the [Preferences](#) topic for more information.

The Windows Menu Bar may also be turned on and off manually using the \$menubar utility.

\$menubar('load') - Will turn the menu bar on or reload the menubar from the definition file.

\$menubar('unload') - Will turn the menu bar off.

## Customization

The Windows Menu Bar is defined in a file *sysmenu.txt* which is located in the \$ARRIS\lib directory. This file may be copied to the *Standards*, *User Home*, or *Project* directory and customized. If the file exists in either the Standards, User Home or Project directory, it is read from there and the customized version of the Windows Menu Bar is used. For customization to be used office-wide, locate the file in the *Standards* directory. For customization to be used by an individual, locate the file in the *User Home* directory. For customization to be used by every-

one working on a particular project, locate the file in the *Project* directory.

The Windows Menu Bar definition file *sysmenu.txt* is a simple ascii text file and may be edited by any text editor such as Notepad. You can add selections and submenus and also control the look of the submenus with separators and conditional selections. This is a field delimited text file so caution must be taken to preserve the formatting of each line.

A sample portion of a Windows Menu Bar definition file is shown below:

```
# Name;ID;Command or Submenu
&File;::%submenu%
  &New;::%submenu%
    &Project;::mn_newpj;init
    &Database;::mn_newdb;init
    D&rawing/Layers;::mn_dbnew;draw
    &Sheet;::mn_dbnew;sheet
    %end%
  &Open...;::mn_pjselect;init
  %sep%
  &Save;::save
  Save &As;::%submenu%
    %gray%;%path%wrk;not;%path%usr:
    Save As New Project;::mn_pjcopy;init
    %gray%;%none%
    Save As New Database;::mn_drcopy;newdb
    %gray%;#dbnam;not;noname.db
    Rename Database;::mn_dbnew;dbase;rename
    Copy Drawing/Sheet;::mn_drcopy;copy
    Rename Drawing/Sheet;::mn_drcopy;rename
    %gray%;%none%
    Briefcase - Take Out...;::mn_takeout;initcur
    %gray%;#dbnam;not;noname.db
    Briefcase - Check Out...;::mn_takeout;initcur
    Save Drawing Thumbnail;::mn_mkthumb
    %gray%;%none%
    %end%
  %sep%
  &Plot/Print...;::mnlq;PLOT
  PD&F...;::mnlq;MN_PDF
  %sep%
  Settings...;::mnlq;DBSETUP
```

### Menu Entry Line

A menu entry is of the form:

```
Name;ID;Command
```

Name is the text that appears on the menu. If the name has an ampersand, it is used as a "menu accelerator", meaning the user can sometimes type in the letter immediately following the ampersand to execute that line of the menu.

ID can be blank unless you are going to modify the menu entry in a sigmac using the \$menubar utility.

Command is the **ARRIS** command to run when the button is selected. The command may be either a sigmac command (preceded by a colon), or a \$ utility.

### Sub-Menu Entry Line

A submenu may be started as the command by placing the syntax

```
%submenu%
```

in the command field.

A submenu must have an end line:

```
%end%
```

in order to return to the previous level. Submenus are nested within the lines of the calling menu, the entire submenu following the line of the calling menu that calls the submenu. The main Windows Menu Bar must also end with the %end% line.

Note 1: When defining a sub-menu it is helpful to indent the lines defining the buttons in the sub-menu for clarity in reading the definition file. This makes multiple sub-menus that are nested easier to read and also helps insure that the %end% statement is included for each one.

### Grayed-Out or Checked Entries

You can specify that a menu entry be grayed out with the %gray% function or have a check mark next to it with the %check% function. These two functions work similarly.

```
%gray%;Parameter 1;Operator;Parameter 2  
%check%;Parameter1;Operator;Parameter 2
```

The button will be grayed out or checked according to the condition of the logical string - Parameter 1;Operator;Parameter 2 - which follows it.

%gray% items will be grayed out if the condition is False.

%check% items will be checked if the condition is True.

The Operator can be:

- equals - True if Parameter 1 equals Parameter 2
- not - True if Parameter 1 does not equal Parameter 2
- gt - True if Parameter 1 is greater than Parameter 2
- lt - True if Parameter 1 is less than Parameter 2

Parameter 1 and Parameter 2 can be:

- A system variable such as #dbnam
- A number (integer or real) such as 10 or 1.234
- A text string such as "Yes" (quotes not required)
- %path% followed by a file system path that the \$getnam utility would understand. This includes **ARRIS** shortcuts such as std: or usr:.

For example:

```
%path%std:
```

would be recognized as the path to the Standards directory.

- %glob% followed by the name of a Sigmac global variable. The value of a global variable may be checked in this way, for example:

```
%glob%imn_projmode>equals;0
```

The *Gray Out* or *Check* condition will apply to all entries that follow it. The Gray Out or Check condition must be ended by the line:

```
%gray%;%none%
```

or

```
%check%;%none%
```

In the example:

```
%gray%;#dbnam.not.noname.db
```

the button that follows will be grayed out when the database name is not noname.db.

### Separator Bar

A sub-menu may contain a separator bar by adding the %sep% function between the lines defining buttons to be separated. This allows you to visually organize buttons on the sub-menus for clarity.

### Comment Lines

A line in the Windows Menu Bar definition file which begins with the Pound sign (#) is a comment line and is ignored when the file is read for loading the menu bar.

### Menu Bar Text Conventions

By convention, on sub-menus the command text which calls another sub-menu is followed by dots in the form:

```
New...
```

This indicates that this selection will pull up another sub-menu. Text without the dots indicates that a function will be executed directly.

## See Also

[Preferences.](#)