GENSCRNX Help

By Steven Black 76200,2110

Version: 0.90 Feb.15.1994

GENSCRNX was created by Ken R. Levy at the Jet Propulsion Laboratory in Pasadena, California.

Pick a general topic below, or SEARCH from the menu bar.

GENSCRNX CONFIG.FP Commands

Set up your environment for GENSCRNX

The Most Important of All:

_GENSCRN=<File>

To specify which screen generator system to use. The default is the native FoxPro screen generator. To invoke the GENSCRNX system, use

```
_GENSCRN="<PATH\>GENSCRNX.PRG"
```

If things don't function as expected, check the FoxPro _GENSCRN system memory variable like this:

```
WAIT WINDOW _GENSCRN
```

It should point to "GENSCRNX.PRG".

_AUTOHALT=ON|OFF

Instructs the GENSCRNX process to abort ON ERROR. The default is ON. Overridden by m._AUTOHALT if it exists.

BRACES=ON|OFF

Enables (the default) or disables the automatic searching for GENSCRNX {{...}} expressions to evaluate. Setting BRACES=OFF can significantly improve the performance of screen generation. BRACES=OFF can be overridden in the screen setup snippet with a *:BRACES command. Similarly, *:NOBRACES overrides BRACES=ON. This doesn't mean GENSCRNX won't process braces, just that it won't spend time explicitly searching all the memo fields for them. If GENSCRNX sees them, they will be evaluated. To force GENSCRNX to completely ignore braces, use the GENSCRNX *:IGNOREBRACES setup snippet directive.

COMPSPR=ON|OFF

Causes auto-compilation of the .SPR when generated. The default is OFF. Specify ON to automatically create an .SPX file. Note that this has no effect when generating from a project. Overridden by m._COMPSPR if it exists, and by the *:NOCOMSPR setup snippet command.

DISPSPR=ON|OFF

Causes automatic display of the screen's .SPR file and .ERR file (if it exists) after generation. The default is OFF. Specify ON to view these automatically. Note that this has no effect when generating from a project. Overridden by m._DISPSPR if it exists, and again thereafter if *:NODISPSPR in the screen setup snippet.

_GENSCRNX <File>

Specifies the generator program to be called by GENSCRNX. The default is SYS(2004)+"GenScrn.PRG". Overridden by the m._GENSCRNX variable if it exists. Third-party screen generators, like ProGen from Pacific Data Systems, work very well with GENSCRNX. So will modified versiond of GenScrn.PRG if the modifications don't affect the textmerge code patterns..

_GENSCX=<File>
Specifies which program does the brunt of GENSCRNX pre-generation work. Defaults to GENSCX(), the workhorse procedure found in GENSCRNX.PRG.

_FOXSCX=<Table>
Specifies the repository for library objects. The default is SYS(2004)+"FOXSCX.DBF". If this table is not found, GENSCRNX will attempt to create it.

MVCOUNT <expN>
Make sure this is set to 512 or greater. Mine is set to 1000. Since this is a native FoxPro function, if you change this, you need to re-enter FoxPro for it to take effect. GENSCRNX detects if you haven't got a high enough MVCOUNT, and will issue a warning.

OUTTXT=ON|OFF
Includes the lines "** Start of inserted text" before and "** End of inserted text" after text inserted with #:INSERT. The default is OFF. Overridden by m._OUTTXT if it exists.

PLATONLY=ON|OFF
To generate code for the current platform only. Default is OFF. Overridden by m._PLATONLY if it exists. Setting PLATONLY to ON significantly improves generation performance.

SCRNOBJ=ON|OFF
Enable (the default) or disable the generation of first and last placeholder-objects at 0,0 of your screens. These objects are invisible, and allow you to generically reference the first and last objects in a creen..

_SCXDRV1 to 8=<Driver>
To specify the names of drivers to be invoked at each of eight separate "hooks" in the GENSCRNX pre-compilation process. You may specify several drivers per hook in CONFIG.FP. See the section "Understanding Hooks" below.

_SPRDRV1 to 6=<Driver>
To specify the names of drivers to be invoked at each of six post-compilation "hooks". The considerations in the _SCXDRVx paragraph above apply here also.

_UPDSPR=<File>
To specify which program does the brunt of GENSCRNX post-generation work. Defaults to UPDSPR(), a procedure within GENSCRNX.PRG.

{{<expC>}}

This executes <expC> and replaces the {{<expC>}} with what's returned by <expC>. Note that the CONFIG.FP is, at generate time, in a temporary memo field created by GENSCRNX. Beware that the {{}} calls in CONFIG.FP are evaluated very early in the GENSCRNX process, before the work files are opened or created. With {{<expC>}} in the CONFIG files, you can, among other things, update the contents of the memo containing CONFIG.FP at generate-time. You could invoke IIF() functions for calculated settings, or even return multiple lines of CONFIG.FP statements.

{{<<File>}}

Inserts the contents of a file into your CONFIG.FP at compile time. The < that follows the open braces is the command that evaluates the contents of a file and inserts the file at that location. The file inserted can contain other GENSCRNX directives and also may contain any {{<expC>}} expressions to be evaluated.

{{&.<expC>}}

Same as {{<expC> }} except the {{&.<expC>}} expression is replaced with a null string. This is another way to make a "void call" to a UDF; for instance, {{&.MYUDF()}} executes MYUDF(), discards any return value, and substitutes a null string.

GENSCRNX Memory Variables

These override CONFIG.FP settings

m._AUTOHALT "ON"|"OFF"

"ON" causes GENSCRNX process to stop ON ERROR.

m._BRACES "ON"|"OFF"

"OFF" improves performance by disabling the serach for {{}} expressions to evaluate.

m._COMPSPR "ON"|"OFF"

"ON" specifies auto-compilation of the .SPR once it is generated.

m._<Driver Name> "ON"|"OFF"

In conjunction with the GENSCRNX drvenable() function (see Other Useful Functions below), can be used to make a driver inactive. For example, the GENSCRNX 3D driver can take a long time. If you are developing and don't care for cosmetic issues at the moment, interactively setting

```
m._3d="OFF"
```

can greatly increase the throughput of GENSCRNX. For this to work, your drivers should check for drvenable()=.T. before proceeding. For example:

```
*-- First line of driver
IF ! m._<Driver Name>
   GO BOTTOM
   RETURN .F.
ENDIF
```

m._DISPSPR "ON"|"OFF"

"ON" causes automatic display of the screen's .SPR file and .ERR file. No effect when generating from a project.

m._FOXSCX <File>

To specify a FOXSCX table

m._GENSCRN <File>

Specifies which screen generator system to use. The default is the native FoxPro screen generator in the SYS(2004) directory.To invoke the GENSCRNX shell, use

```
_GENSCRN="<PATH>GENSCRNX.PRG"
```

m._GENSCRNX <File>

Specifies the generator program to be called by GENSCRNX.

m._OUTTXT "ON"|"OFF"

"ON" causes inclusion of the lines "** Start of inserted text" before and "** End of inserted text" after text inserted with *:INSERT.

m._SCNOBJ "ON"|"OFF"

Do or don't create screen placeholder objects.

m._PLATONLY "ON"|"OFF"

To generate objects for the current platform only. Marked improvement in performance when "ON".

GENSCRNX Setup Commands

Control screen-level GENSCRNX processes.

*:SET <ATTRIBUTE>

A variety of SET commands are available to control the generation options of screen sets. These are useful for assigning attributes to stand-alone screen sets, or to override the settings maintained by the project manager. These are largely self-explanatory and work just as you might expect. Please also see the *:PJXSET command.

```
*:SET OPENFILES ON|OFF
*:SET CLOSEFILES ON|OFF
*:SET DEFWINDS ON|OFF
*:SET RELWINDS ON|OFF
*:SET READCYCLE ON|OFF
*:SET MULTREADS ON|OFF
*:SET NOLOCK ON|OFF
*:SET MODAL ON|OFF
*:SET PLATONLY ON|OFF
*:SET BORDERGETS ON|OFF
*:SET ASSOCWINDS TO <Window title list>
```

*:AUTORUN ON|OFF

Automatically executes the generated file after it has been generated. This will not happen if an error occurs in the generate process. Default behavior is OFF.

*:BASLIB <Library>

Base library name for field name match. Note that the objects all come from FOXSCX.DBF. The libraries are differentiated by the contents of the FOXSCX.OBJLIB_ field. The *:BASLIB argument, like all GENSCRNX directives, is not case sensitive.

*:BRACES

Forces GENSCRNX to search for {{...}} expressions to evaluate. Overrides BRACES=OFF in CONFIG.FP.

*:COMPSPR ON|OFF

Overrides COMPSPR=OFF in CONFIG.FP and the m._COMPSPR memory variable.

*:DEFLIB <Library>

Define a library name under which to store all the objects in this screen set. All the objects are stored in FOXSCX.DBF, with the OBJLIB_ field set to the <Library> specified here. *:DEFLIB is not case sensitive. Only library screens should use the *:DEFLIB command.

*:DEFOBJ <Name>

Defines a library object name for the screen header record, which enables access to READ-level snippets and screen attributes. If the setup snippet also contains a *:DEFLIB statement, the screen

is a library screen and the header record is stored in FOXSCX.DBF records with FOXSCX.OBJNAME_ = <Name>. When this screen is generated, the FOXSCX record for this screen's header record
will be created or updated. No SPR is ever created for screens containing a *:DEFLIB statement. If the setup snippet does not contain a *:DEFLIB statement, then the *DEFOBJ <Name> provides you with an "alias" which you can use to access screen header fields. For example, if the setup snippet
contains the line

```
*:DEFOBJ Header
```

Then any object snippet in the screen could include the following command:

```
WAIT WIND "The Screen Font is " +;
          {{HEADER::FONTFACE}}+ " " + ;
          {{HEADER::FONTSIZE}}
```

*:DISPSPR ON|OFF

Overrides DISPSPR=OFF in CONFIG.FP and the m._DISPSPR memory variable.

*:DRVOFF <file>

Disables driver settings specified in the CONFIG.FP. The number of *:DRVOFF directives specified in the Setup snippet is unlimited and the files included are retained for all screens in a screen set. If *:DRVOFF is specifed in the Setup snippet in more than one screen of screen set, the list of drivers disabled is cummulative.

*:GENSCRNX <File>

Specifies a program to generate screen code for the current screen. *:GENSCRNX overrides any _GENSCRNX found in the CONFIG.FP files, and any m._GENSCRNX memory variable. If neither *:GENSCRNX or _GENSCRNX are specified, the default is GenScrn.PRG located in SYS(2004), the FoxPro startup directory. *:GENSCRNX can be used to specify a modified GenScrn needed for a particular screen.

*:IGNOREBRACES

Forces GENSCRNX to ignore all {{<expC>}} expressions. This overrides the *:BRACES and *:NOBRACES directives. This command is useful when using a screen to assemble source code that contains GENSCRNX brace functions.

*:INCLIB <Library>

Include a library in the base object search path for objects. You can have any number of *:INCLIB statements to specify many libraries in the search path. Libraries are searched in the order they are specified. Note again that all libraries are physically contained in FOXSCX.DBF.

*:INCLIB does not trigger implicit referencing to GENSCRNX library objects. Implicit referrencing, which is triggered by the *:BASLIB statement, causes library attributes to be pulled-in when the working screen's memvar matches the library screen's memvar.

#:INSERT <File>

Same as FoxPro's native #INSERT except faster, and the inserted lines may contain GENSCRNX commands and directives. Since the lines are inserted before the generator program is called, they may also contain native FoxPro directives. #:INSERT works with FoxPro 2.0 whereas, of course, native FoxPro #INSERT does not. #:INSERT is processed after SPRDRV5.

#:INSERTTOP <File>

Inserts a file at the top of .SPR code, before the cross-platform DO CASE block. If several identical #:INSERTTOP statements are found due to cross-platform duplication, the file is inserted only once. Useful for inserting files containing #DEFINE statements, copyright text, and other things that need apply just once in a generated file.

**:MEMVAR*

Replace all aliases in GET objects with "m.". The SCX fields processed are NAME, WHEN, VALID, MESSAGE, ERROR, RANGELO, and RANGEHI. This is accomplished after executing each _SCXDRV3, after each object is inserted with *:INSOBJ or *:INSSCX comment directives.

**:NAME <Snippet name>*

This directive, which can be used in any snippet, is an enhanced version of the native FoxPro #NAME directive. *:NAME automatically creates individual procedure names for snippets on each platform. The character "d" is appended to the DOS version of <Snippet name>, "w" for Windows, "m" for Mac, and "u" for Unix. If the original <Snippet name> is longer than 9 characters, GENSCRNX forces the platform character into the 10th character position. *:NAME, if used, must be on the first line of the snippet.

**:NOBRACES*

Turns off the auto braces detection for GENSCRNX. By default, GENSCRNX automatically searches all snippets for any {{<expC>}} expressions to be evaluated. This results in a few seconds overhead depending on the speed of the computer being used. Using *:NOBRACES will force GENSCRNX to only evaluate snippets if *:EVLTXT is in the Comment snippet (or Setup snippet for the header record). Overrides a CONFIG.FP setting of BRACES=ON. See "NOBRACES" in Table 1 for additional information.

**:NOCOMPSPR*

Overrides COMPSPR=ON in CONFIG.FP and the _COMPSPR memory variable.

**:NODISPSPR*

Overrides DISPSPR=ON in CONFIG.FP and the _DISPSPR memory variable.

**:NOGEN*

Prevents GENSCRNX from calling the GenScrn program, or whatever is specified with _GENSCRNX in CONFIG.FP or by the m._GENSCRNX memory variable. Use *:NOGEN in screen libraries that are never meant to execute.

*:NOSCRNOBJ

Disables the GENSCRNX placeholder invisible button. *:NOSCNOBJ can be used to override a SCNOBJ=ON setting in the CONFIG.FP files. *:NOSCNOBJ is automatic when either the #NOREAD directive exists in the Setup snippet or no GET objects exist for the screen. See the *:SCRNOBJ command below.

*:NOWCLAUSES <clause list>

Removes clauses from the DEFINE WINDOW command. To directly add any of the removed clauses, use GENSCRN's #WCLAUSES directive. Example: Add following lines in the Setup snippet to add a custom AT <row,col> SIZE <height,width>:

```
*:NOWCLAUSES AT SIZE
 #WCLAUSES AT 1,1 SIZE 10,30
```

or

```
 *:NOWCLAUSES AT SIZE
 #WCLAUSES AT {{VPOS}},{{HPOS}} SIZE
{{HEIGHT}},{{WIDTH}}
```

Example: Add following lines in the Setup snippet to add a custom FONT <fontface> STYLE <fontstyle>:

```
*:NOWCLAUSES FONT STYLE
#WCLAUSES FONT myfont STYLE mystyle
```

*:NOXGEN

Prevents GENSCRNX from updating the temporary .SCX or .SPR files.

*:PJXSET

Forces the project information to have priority over GENSCRNX *:SET commands when building from a project. This has no effect when generating stand-alone screens or on prior *:SET commands.

*:OUTFILE <File>

Specify a filename for output. You must supply a file extension or else a standard screen-file-name.SPR will be created. When generating a stand-alone screen, the FoxPro Generate dialog may give a "Filename.SPR already exists, overwrite it?" dialog even when you've given GENSCRNX other file naming instructions. GENSCRNX cannot control this dialog. This only occurs when SAFETY=ON.

*:PRG

To automatically change the .SPR extension to .PRG and add the #NOREAD PLAIN directive to the setup snippet. Using this in a screen with *:INSTXT allows building a .PRG without a READ composed of source code assembled in the screen object sequence. *:PRG is processed after _SCXDRV1.

**:SCRNOBJ*

Enables the GENSCRNX invisible button. *:SCNOBJ can be used to override a SCNOBJ=OFF setting in the CONFIG.FP files.

**:SCXDRV1 to 8 <Driver>*

Specify any number of drivers to be invoked at the eight pre-generation "hook" points. Drivers at each hook are invoked in the sequence they are specified. Please see the "Understanding Hooks" section below.

#:SECTION 3

Used like the native #SECTION 1 and #SECTION 2 directives, this will place code after the GETs are defined but before the READ. The presence of #:SECTION3 is tested before _SCXDRV1, and triggers actual insertion of a new record containing the #SECTION 3 code after _SCXDRV7.

**:SPRDRV1 to 6 <Driver>*

Specify any number of drivers to be invoked at the six post-generation "hook" points. Drivers at each hook are invoked in the sequence they are specified. Please see the "Understanding Hooks" section below.

{{<expC>}}

Executes <expC> at and replaces the {{}} command with what's returned by <expC>. {{}} statements are evaluated at a number of points in the GENSCRNX process.

{{<<File>}}

Insert a file at compile time. The < that follows the open braces is the command that evaluates the contents of a file and inserts the file at that location. {{< <file> }} can be included in the Comment snippet and the file inserted can contain other GENSCRNX directives and also may contain any {{<expC>}} expressions to be evaluated.

{{&.<expC>}}

Same as {{ }} except that the {{&.<expC>}} expression is executed and always substituted with a null string.

{{@ <expC>}}

Retrieves a GENSCRNX directive at compile time. This can be used to control default settings globally to have muliptle screens and objects use the same directives. The @ symbol triggers a wordsearch() in the Comment or Setup snippet for the directive specified by <expC>. The insertion can contain other GENSCRNX directives. Example: If the following command was in a screen's setup snippet:

```
*:SCXDRV5 3D
*:ALL3D {{MAIN.Pref_3d::@*:ALL3D}}
```

and library MAIN contains an object called PREF_3D that having *:ALL3D 4 in the Setup snippet, then the code in our screen's Setup snippet would become

```
*:SCXDRV5 3D
*:ALL3D 4
```

GENSCRNX Comment Commands

For object-level control of features

**:BASBEFORE*

Forces expressions inherited from libraries to be inserted before any existing snippet code. In the normal course of events, in the abscence of *:BASEBEFORE, expressions are appended to existing snippets.

**:BASOBJ [library.]<object>*

Specify a base library object from which this object is to draw its attributes. You may use several *:BASEOBJ commands to combine the attributes of several base objects into the current object. If the library is invoked with *:BASLIB, a *:BASOBJ statement is not necessary when the memory variable of the control matches the memory variable of a library object.

**:CLICK <Function>*

Overlays an invisible button with a WHEN snippet that calls function specified when clicked with the mouse. The () after the function name are only required if parameters are passed. In the Windows platform, *:CLICK supports text, box, and picture objects while in the MS-DOS platform, *:CLICK supports text and box objects. Example: To have a function called myfnct() exectuted from a mouse click on the object, place the following in the Comment snippet: *:CLICK myfnct

**:DEFAULT <expC>*

Replaces the DEFAULT field of the object with <expC>. Note that Lists, Invisible Buttons, and Spinners don't use the DEFAULT field. You may also append other valid GET arguments, for example: *:Default '"Quit" COLOR <color> MESSAGE "..."'

**:DEFOBJ <Object>*

Defines a library object name for the current object. If the object is in a library screen (one with a *:DEFLIB <name> setup directive), then when this screen is generated, GENSCRNX will create or update the FOXSCX record for this object.

If the object is not in a library screen, then the *:DEFOBJ serves to define an "alias" for the object from which attributes may be queried by other objects in the same screen by using brace {{::}} directives.

**:DELETE*

Removes the object at generate-time. Very handy for, among other things, applying visible comments to screens. The *:DELETE objects are visible to _SCXDRV1 drivers only, and code after _SCXDRV6 purges any *:DELETE objects introduced from libraries.

**:DELOBJ*

Deletes a screen object at compile-time after pre-processing is complete. Use *:DELOBJ for objects that need to reside in the .SCX database during pre-processing but need not be generated into the .SPR file. Objects are deleted after the first driver is run.

**:ENDTXT*

Marks the end of code added with :*INSTXT.

**:ENDFNCT*

Marks ending of the *:FUNCTION code to be added to the cleanup procedures.

**:EVLTXT*

By default, this directive is not needed. *:EVLTXT forces evaluation of any {{...}} found in any of this objects snippets. This directive is only used when either *:NOBRACES is specified in the Setup snippet or BRACES=OFF is specified in the CONFIG.FP.

**:FUNCTION <expC>*

Text between the *:FUNCTION and *:ENDFNCT commands will be automatically written to the cleanup code of the current screen. With these two directives, you can make library objects automatically inherit any necessary procedure code. Note that the <expC> is part of the actual function call of the first included function. *:FUNCTION processing is made after execution of _SCXDRV7. For example, to include two functions:

```
*:FUNCTION MyWhenCode
...
RETURN

FUNCTION MyValidCode
...
RETURN
*:ENDFNCT
```

**:IF <expL>*

Brackets the current object with an IF <expL> ... ENDIF. Very handy for selectively producing screen objects. *IF objects are processed after _SCXDRV4, and carry over to the SPR driver phase of GENSCRNX. Therefore *:IF directives increrase generation time.

#:INSERT

Same as FoxPro's native #INSERT except faster, and the inserted lines may contain GENSCRNX commands and directives. Since the lines are inserted before the generator program is called, they may also contain native FoxPro directives. #:INSERT works with FoxPro 2.0 whereas, of course, native FoxPro #INSERT does not. #:INSERT is processed after SPRDRV5.

**:INSOBJ [<Library.]<Object>*

Substitute an object from FOXSCX.DBF in place of this object. Note that you may have more than one *:INSOBJ statement per object. Objects are inserted after _SCXDRV3. This is one reason why the GENSCRNX 3D driver is not recommended for _SCXDRV3, since it may miss inserted objects. The sixth screen driver hook, _SCXDRV6, is filtered on new objects, so drivers that need to operate on new objects only should use hook 6.

*:INSSCX <File>

Substitute a screen from a library in place of this screen object. The size of the object containing the *:INSSCX directive determines the area allowed to insert objects. Note that FOXSCX has a OBJSCX_ field which is used to group items by screen. Screens are inserted after _SCXDRV3, and cause subsequent calls to _SCXDRV2 and 3. The sixth screen driver hook, _SCXDRV6, is filtered on new objects, so drivers that need to operate on new objects only should use hook 6.

*:INSTXT

Marks the beginning of code to be added in the place of this screen object. See also *:ENDTXT. *:INSTXT occurs after _SCXDRV7 and has implications in code following _SPRDRV3.

*:NOSIZE

Suppresses the SIZE clause for this screen object. *:NOSIZE is executed after _SCXDRV4 and triggers SPR maintenance after _SPRDRV2. Affects the HEIGHT, WIDTH and SPACING fields. Note that *:NOSIZE temprarily replaces all non-edit region fields the contents of the HEIGHT and WIDTH fields with (m.scxcount-100) and RECNO()-100 respectively, so drivers between *:SCXDRV5 and _SPRDRV2 shouldn't refer to these fields without applying the proper corrections. Best interupt a trial-generation with a simple SUSPEND driver to see exactly what you are dealing with here...

*:PICTURE <expC>

Replaces object PICTURE clause with <expC>. <expC> can be any character expression, including variable names or FoxPro functions. *:PICTURE directives are processed after _SCXDRV4. Example: To force a data driven PICTURE clause for an object using a variable called m.mypict, place the following in the Comment snippet:

```
*:PICTURE m.mypict
```

In the latest version of GENSCRNX (version 1.8) the *:PICTURE command can be used to create multi-state pictures in the Windows version of FoxPro. This allows different .BMP or .ICO files to be used for "On", "Off", and "Disabled" modes of a picture control.

*:REFRESH

Replaces object REFRESH clause with .T.. *:REFRESH will override the refresh setting for a SAY object and can also be used to allow a picture to be refreshed in the Read Level Show using either SHOW GETS or SHOW GETS OFF. Using both *:REFRESH and *:PICTURE <variable name> with a picture from file object can allow picture fields to be refreshed at runtime. *:REFRESH is processed after _SCXDRV8.

*:SIZE <expC>

Substitutes the object's SIZE clause with SIZE <expC>. *:SIZE is processed after _SCXDRV4, and triggers subsequent processing after _SPRDRV2.

*:SAVEPICT

Forces the PICTURE information to come from the associated library object. Processed after _SCXDRV4, and affects the PICTURE, INITIALNUM and INITIALVAL fields.

**:SAVESIZE*

Forces the SIZE information to come from the associated library object. Processed after _SCXDRV4 and affects the HEIGHT, WIDTH, INITIALNUM and SPACING fields.

**:TRNTXT <expC1> || <expC2> || <expN1>[||<ExpN2>]*

Works like STRTRAN(), except it operates on ALL memo fields in the current record, and the search is not case sensitive. *:TRNTXT is processed following _SCXDRV3.

{{@ <expC>}}

Retrieves a GENSCRNX directive at compile time. See the explanation in Table 3. Using {{@ <expC> }} in the Comment snippet can be used to retrieve other GENSCRNX comment directives. Example: If the following command was in the Comment snippet of an object:

```
*:IF {{Button1::@*:IF}}
```

then the object would use the *:IF directive specifed in another object containing *:DEFOBJ Button1 in it's Comment snippet. This could be used, for example, to generate a box around an object if the object is generated, or to coordinate the manifestation of several controls based on a single comment directive.

GENSCRNX Procedure Commands

GENSCRNX directives that go anywhere

#:INSERT

Same as FoxPro's native #INSERT except faster, and the inserted lines may contain GENSCRNX commands and directives. Since the lines are inserted before the generator program is called, they may also contain native FoxPro directives. #:INSERT works with FoxPro 2.0 whereas, of course, native FoxPro #INSERT does not. #:INSERT is processed after SPRDRV5.

**:NAME <Snippet name>*

This directive, which can be used in any snippet, is an enhanced version of the native FoxPro #NAME directive. *:NAME automatically creates individual procedure names for snippets on each platform. The character "d" is appended to the DOS version of <Snippet name>, "w" for Windows, "m" for Mac, and "u" for Unix. If the original <Snippet name> is longer than 9 characters, GENSCRNX forces the platform character into the 10th character position. *:NAME, if used, must be on the first line of the snippet.

{{<expC>}}

Executes <expC> at and replaces the {{}} command with what's returned by <expC>. {{}} statements are evaluated at a number of points in the GENSCRNX process.

{{<<File>}}

Insert a file at compile time. The < that follows the open braces is the command that evaluates the contents of a file and inserts the file at that location. {{< <file> }} can be included in the Comment snippet and the file inserted can contain other GENSCRNX directives and also may contain any {{<expC>}} expressions to be evaluated.

{{&.<expC>}}

Same as {{ }} except that the {{&.<expC>}} expression is executed and always substituted with a null string.

{{@ <expC>}}

Retrieves a GENSCRNX directive at compile time. This can be used to control default settings globally to have muliptle screens and objects use the same directives. The @ symbol triggers a wordsearch() in the Comment or Setup snippet for the directive specified by <expC>. The insertion can contain other GENSCRNX directives. Example: If the following command was in a screen's setup snippet:

```
*:SCXDRV5 3D
*:ALL3D {{MAIN.Pref_3d::@*:ALL3D}}
```

and library MAIN contains an object called PREF_3D that having *:ALL3D 4 in the Setup snippet, then the code in our screen's Setup snippet would become

```
*:SCXDRV5 3D
*:ALL3D 4
```

GENSCRNX Libraries

Working Smart With Reusable Components

GENSCRNX Libraries

An Exellent to Use Reusable Code

Among GENSCRNX's many capabilities, perhaps none is more useful than library objects. Here we will look at how to create, use, and maintain GENSCRNX library objects.

Recall that FoxPro's screen files, those with .SCX and .SCT extensions, are really just standard FoxPro tables with different extensions. The .SCX file is a .DBF, and the .SCT is a standard FoxPro memo, just like any other .FPT file. All information about a screen it's screen objects is conveniently stored this way by FoxPro's screen builder. GENSCRNX exploits FoxPro's open architecture, and allows you to create a library of reusable screen objects. This library file is called FOXSCX.DBF and is, by default, created and kept in the FoxPro startup directory. You can control the name and location of the screen object library with the _FOXSCX=<path\filename> CONFIG.FP directive. I recommend that you map the FOXSCX table to the same file for all platforms.

So a GENSCRNX library is a FoxPro table containing reusable screen objects. A library object is a record in the GENSCRNX library table. The library is built and maintained through the interface of a dummy FoxPro screens (a library screen) into which you can paste your re-usable screen objects.

Creating a GENSCRNX Library

Creating GENSCRNX libraries is easy! Just follow these steps.

- Make sure your FoxPro m._GENSCRN variable is set to GENSCRNX.

- Create a new screen, call it, say, MY_LIB.SCX. Note that you can have any number of library screens.

- Put a GENSCRNX *:DEFLIB statement in the setup snippet.The *:DEFLIB identifies the screen as a GENSCRNX library, and triggers special processing when the screen is generated with the GENSCRNX shell. "My_LibName" is any name you choose. I typically chose a name that describes the type of objects contained in this screen, like "Controls", or "Lookups", or maybe "Dates_With_Valids". GENSCRNX respects the first 10 characters of a library name. Like this:

  ```
  *:DEFLIB My_LibName
  ```

- Copy and Paste any reusable screen objects into the library screen. For each object copied to the library screen, place a *:DEFOBJ statement in the comment snippet. This object name is the alias that you will use in your working objects to refer to this library object. The GENSCRNX sample files contain many useful objects you can use to start your own libraries. Up to 24 characters are allowed for an object name.

  ```
  *:DEFOBJ <Object Name>.
  ```

- From the PROGRAM menu, chose GENERATE. Unlike working creens, generating a library screen (a screen containing a *:DEFLIB statement) doesn't create an .SPR file. Instead, GENSCRNX creates or refreshes the screen's library objects in FOXSCX.DBF. Note that, no matter which library name you use in the *:DEFLIB statement, all library objects will go into the FOXSCX table. You can therefore create many different libraries, but all library objects will be stored in the same FOXSCX library table.

Referencing Mechanics

The word "inheritance" could be used as a synonym for library referencing. Inheritance, however, has connotations in computer science that make it an inappropriate term to use here. I'll use referencing instead, which means the copy and pasting of library object properties into your working screens.To reference library objects, place a either a *:BASLIB or an *:INCLIB tatement in the setup of your working screen, like this: ·

```
*:BASLIB My_LibName
```

This causes GENSCRNX to search the My_LibName library for object library references. We'll discuss the difference between *:BASLIB and *:INCLIB in a moment. You may have mutiple *:BASLIB and *:INCLIB statements if your screen refers to objects from several different libraries.

It's helpful to remember the following points about how GENSCRNX swaps library information into your working screen:

- The GENSCRNX *:BASLIB and *:INCLIB setup directives set the sequencing of libraries for earching in the FOXSCX table. Only the libraries specified are searched.
- As a general rule, if a field in your working screen object is empty, and the library field is not, referencing (inheritance) will occur for that field.
- For snippet expressions, the library expression is ANDed and if the snippet is a procedure, the library procedure is appended. This is reversed if the library object contains a directive called *:BASBEFORE. A working object will receive library-object colors if the colors of the working screen are set to default (or automatic).
- A working object will receive the library object font information if the working object's font is set to MS Sans Serif, 8, N.
- A working object will reference the library object Picture clause if it's picture clause is either empty or set to @K (which is default in Windows). If the library object contains *:SAVEPICT, the referencing will overwrite the working object's picture clause. If the library object contains *:SAVESIZE, the referencing will overwrite the working object's height and width
- Fields that are numeric or logical are referenced if the working object's value is empty (0 or .F.) and the library object's field is not empty. For example, if RANGEHI was 0 in the surface object and 3 in the library object, the 3 would be inherited. If the DISABLED check box was not set in the surface object but was checked in the library object, the DISABLED setting would be inherited. Once a field (numeric or logical) contains a non-empty value, it will no longer inherit from any referencing operations (such as the library object referencing another library object) or when an object uses multiple referencing (multiple *:BASOBJ directives in one Comment snippet).

Explicit Referencing

*Use *:BASOBJ to bring-in properties*

You can explicitly append all the library object snippets to your working object by placing a *:BASOBJ statement in the comment snippet of your working object. For example,

```
*:BASOBJ <ObjName>
```

...will automatically assign the snippets and attributes from a previously created library object named <ObjName>.

You may aggregate the snippets from many library objects into your working objects with multiple *:BASOBJ statements, like this:

```
*:BASOBJ <Object1>
*:BASOBJ <Object2>
*:BASOBJ <Object3>
```

The referenced library objects are inherited in the order they are listed. It isn't necessary for the library object and the working object be of the same object type, since the snippet fields are found in all object records. So the VALID of a library Push Button could well be applied to a Radio Button, or any other type of screen control. The attributes of the screen itself are stored in what's called the screen header record. This is the first record in the .SCX file for each platform. The screen header record (ObjType=1) contains, among other things, the Setup, Cleanup, and the Read When, Show, Valid, Activate, and Deactivate snippets.

As far as GENSCRNX is concerned, the screen header record is a screen object like any other. There is, however, no way to reach the screen header comment snippet in the Screen Builder. GENSCRNX screen header directives are stored in the Setup snippet.

Implicit Referencing

*Trigger automatic referencing without *:BASOBJ*

If your working object variable has the same name as a library object, and the library is triggered with *:BASLIB, then inheritance is triggered automatically, even without a *:BASOBJ statement. This can lead to unexpected results; so be careful when naming a library object's variable.

For example, if you have a library object named m.My_Memvar, and a working object also to m.My_Memvar, then a link to the library happens automatically. Provided, of course, that the working screen has the appropriate *:BASLIB statement. Using *:INCLIB instead of *:BASLIB doesn't trigger implicit inheritance.

Partial Referencing

Reference selected parts of library objects

You can pull-in distinct fields from library objects into your working objects by using the GENSCRNX ":: " operator. The construct used for partial inheritance is

```
{{<Library Name>::<Snippet Name>}}
```

For example, pull the contents of the VALID clause for the My_Library_Object into the current snippet with the following statement.

```
{{My_Library_Object::VALID}}
```

These may also be combined. For example, say you have a bunch of Date library objects. Assume that one called "Date_This_Year" validates dates in this fiscal year. Another named "Date_Is_Workday" validates dates in the work week. A third called "Date_Is_Holiday" validates holidays. You could construct a compound VALID statement to validate working days for this fiscal year in the following way:

```
RETURN {{Date_This_Year::VALID}} AND ;
   {{Date_is_Workday::VALID}} AND ;
   NOT {{Date_Is_Holiday::VALID}}
```

In this way you can create complex validation criteria based on simple normalized components. The default behavior in GENSCRNX is to act on the comment snippet. So the statements below are equivalent:

```
{{My_Library_Object::}}
{{My_Library_Object::COMMENT}}
```

Methodical Referencing

Pull-in Snippet Segments

By using a refinement of the partial referencing mechanism, you can pull-in portions of library object snippets. This means you can store different "methods" associated with an action in the same snippet and refer just to the methods you need in the working object. To separate methods in the library object, use the *:METHOD directive.The construct used for methodical referencing is

```
{{<LibraryName>::<Snippet Name>::<Methodname>}}
```

For example, suppose a library GET field which has a variety of WHEN conditions checking for say security clearance and pop-up features. You could structure the library WHEN as follows:

```
*-- WHEN snippet of a library object
*-- named "Security_Check"
:*METHOD Security_MED
*-- Dissalow if below level 3
   IF m.gnSecurity < 3
      RETURN .F.
   ENDIF
*:ENDMTHD

:*METHOD Security_HIGH
*-- Dissalow if below level 4
   IF m.gnSecurity < 4
      RETURN .F.
   ENDIF
*:ENDMTHD

*-- WHEN snippet of a library object
*-- named "Date_Stuff"
:*METHOD Calendar_POPUP
  *-- Initialize a calendar popup with RIGHTMOUSE
  *-- Note that expression in braces ( {{name}} ) will be
  *-- evaluated by GENSCRNX as the memvar of the working
  *-- object at generate time.
  ON KEY LABEL RIGHTMOUSE DO calendar ;
    WITH IIF( EMPTY( {{name}}, DATE(), {{name}})
*:ENDMTHD
```

You can structure the WHEN of a working object as follows:

```
*-- WHEN of working date object
{{My_Library::Security_Check::Security_MED}}
{{My_Library::Date_Stuff::Calendar_Popup}}
```

Would produce the following generated code, assuming the working object memory variable was "ldStartDate"

```
*-- Dissalow if below level 3
   IF m.gnSecurity < 3
      RETURN .F.
   ENDIF
*-- Initialize a calendar popup with RIGHTMOUSE
*-- Note that expression in braces ( ldStartDate ) will be
*-- evaluated by GENSCRNX as the memvar of the working
*-- object at generate time.
*-- generate time.
   ON KEY LABEL RIGHTMOUSE DO calendar ;
     WITH IIF( EMPTY( ldStartDate, DATE(), ldStartDate)
```

Destructive Referencing

Object Substitution

By using the GENSCRNX *:INSOBJ command, you can replace the whole working object with a library object. In this case, the working object becomes a placeholder for the insertion of a library object. Note that you can do more than insert a library object. With the *:INSTXT command, you can insert FoxPro source code in the exact place in the .SPR where the working object would be referenced.

Recursive Referencing

Library objects can themselves contain references to other library objects. GENSCRNX is recursive, and will keep processing until all library objects are added.

Maintaining Library Objects

Making Sure the Links are Refreshed

The FOXSCX table, being a superset of the FoxPro .SCX structure, contains a PLATFORM field. Thus, it's important to generate your library screen under each platform to make sure each library object has a record for each platform you support in FoxPro.

Whenever you regenerate a library screen, GENSCRNX will refresh any existing objects in the FOXSCX table, and adds any new ones it cannot find. Remember that the actual library isn't the library screen; it's the FOXSCX table, which is created by generating the library screen. Therefore, you must re-generate the library screen for all platforms whenever you make a change to a library screen.

There you have it! I hope this compels you to try and use the power of GENSCRNX library objects. Once you get the hang of this, you will find that using the GENSCRNX library functions will both reduce the amount of source code required to produce an application, and make it much easier to maintain.

The latest available version of GENSCRNX is included on the source diskette. As always, you should periodically check the FoxForum libraries for the latest and greatest version.

GENSCRNX Internal Functions

These are available for your custom drivers to use

BASOBJ2(<expC>)

BASOBJ2(<expC>) -Places a *:BASOBJ <expC> statement in the first line of the COMMENT field of the current .SCX record.

CLEANUP()

If, for whatever reason, your drivers need to abort the generation process, like when an intractable error is detected, you can cleanly put an end to the whole process with the following:

```
=cleanup()
CANCEL
```

CLRTXT2()

Blanks the COMMENT field of the current object. Return Value - True

CONFIGFP(<expC1>[,<expC2>])

Queries CONFIG.FP. Returns the argument from the active CONFIG.FP for the <expC1> setting. If no <expC1> setting exists return <expC2>. For example, CONFIGFP("_Genmenu",SYS(2004) + "\GENMENU.PRG") would return the _GENMENU setting in CONFIG.FP. If _GENMENU is not specified in the configuration file, the second parameter is returned (or a null string is returned if no second parameter was passed). Return value - Character

DEFAULT2(<expC>)

Places the string "*DEFAULT <expC>" in first line of the comment snippet of the current record. Returns True if successful, False otherwise. Return value - Logical

DEFOBJ2(<expC>)

Places a *:DEFOBJ <expC> statement in the first line of the COMMENT field of the current .SCX record.

DELOBJ2(<expC>)

Places the string "*DELETE" in first line of the comment snippet of the current record. Return value - True

DRVENABLE(<expC>)

Is memvar/driver "ON"?. Returns True if memory variable _<expC> is "ON" or if a line exists in CONFIG.FP such that <expC> = "ON". This can be used by drivers to check for disabling instructions. See 3D.PRG for example. IF _3D='OFF', the 3D.PRG drivers are disabled via the drvenable() function. Return value - Logical

DUPREC(<expN>)
Duplicate this record. Inserts a duplicate the current .SCX record <expN> records down from the current .SCX record. Return True if successful. Return value - Logical

INSBLANK(<expN>)
Insert a blank record into the .SCX file <expN> records down from the current record. Returns .T. if successful. Return value - Logical

INSERT(<expC>)
Returns the contents of the <expC> file as a stream of characters. Return value - Character

INSREC(<expN>)
Inserts a plain null SAY object into the .SCX file, <expN> records down from the current record, at position 0,0 with height 1, width 1. Returns True if successful. Return value - Logical

INSTXT2(<expC>[,<expN>]) -
Appends <expC> to the COMMENT field <expN> records down from the current .SCX record. Returns True if successful. Return value - Logical

LIBDATA(<expC1>, <expC2>) -
Fetch library object info. Returns information stored in library object named <expC1>. If <expC2> is empty, then it returns the comment snippet of the library object. If <expC2> is not empty, it returns EVAL(<expC2>) with the pointer of the current work area at the library object record. Return value - Character

OBJFACTOR(<expL>)
If <expL> is true, returns the ratio of the object's font width to the font width of the underlying window. If <expL> is false, it returns a ratio of heights.

OBJHEIGHT()
Returns the height of the object expressed in window-font line heights. FONTMETRIC(1) + FONTMETRIC(5)).

OBJSAY()
Create a SAY. Returns an "@...SAY..." string for the EXPR field of the current object with the current HPOS and VPOS coordinates. Return value - Character

OBJWIDTH()
Returns the width of an object expressed in underlying window average font character widths.

SIZE2(<expC>)

Comment *:Size Directive. Places the string "*SIZE <expC>" in first line of the comment snippet of the current record. If <expC> is empty, the *:NOSIZE directive is placed instead of *:SIZE. Returns True if successful, False otherwise.

STRTRANC(<expC>,<expC>,<expC>[,<expN>][,<expN>])

Like native STRTRAN(), only case insensitive. Return value - Character

TRIMDELIM(<expC>)

Removes the outer delimiters from a passed string and returns the resultant string. Warning: GENSCRNX doesn't use this function anyhere, so it's continued presence is not guaranteed.

TRIMFILE(<expC>)
TRIMPATH(<expC>)
TRIMEXT(<expC>)

These handy file and path functions are found in GENSCRNX and can be used by your drivers. Their action is largely self -explanatory.

WORDSEARCH(<expC1>,[[<expL1> | <expC2>],<expL2>])

Returns the line remnant.. The character expression that's searched for is <expC1>. <expL1> | <expC2> The name of the memo field snippet to search of the current record. If <expL1> is .F., the COMMENT memo field searched. If <expL1> is .T., the SETUPCODE memo field searched. If <expC2>
is passed instead of <expL1>, the memo field <expC2> is searched. If <expL1> is .F., search uses match words. If <expL1> is .T., search does not use a "match words" comparison. Returns the balance of the first line found leading with <expC> once <expC> is stripped. If <expL2> is true, the match can be made on partial "words". If a match is not found, wordsearch() returns CHR(0) which is also equal to a GENSCRNX memory variable called m.null.

Example: If the setupcode field is equal to "*:MyDirective ON", the following calls to WORDSEARCH() on the left will return the values shown on the right:

```
WORDSEARCH("*:MYDIRECTIVE")        = ""
WORDSEARCH("*:MYDIRECTIVE",.T.)    = "ON"
WORDSEARCH("*:MYDIREC",.T.,.T.)    = "TIVE ON"
WORDSEARCH("*:MYDIREC",.T.,.F.)    = CHR(0)
```

Example:

```
m.str=wordsearch('*:3D')
```

The string m.str would contain CHR(0) if '*:3D' was not found in the COMMENT snippet of the current record. The string m.str would contain the characters following *:3D if '*:3D' was found. If the line contained '*:3D INSET', then m.str would contain 'INSET'.

Example:

```
m.str=wordsearch('#INSERT','VALID')
```

The string m.str would contain CHR(0) if '#INSERT' was not found in the VALID snippet of the current record. The string m.str would contain the characters following #INSERT if '#INSERT' was found. If the line contained '#INSERT BEEP.PRG', then m.str would contain 'BEEP.PRG'. Return value - Character

GENSCRNX Process Hooks

Sequencing Can Be Important

GENSCRNX has eight pre-generation (_SCXDRV) hooks and six post-generation (_SPRDRV) hooks to which you may attach any number of drivers. In the CONFIG.FP files, like screen setup snippets, you may specify as many drivers to each hook as you like. The CONFIG.FP drivers are executed before the drivers assigned in the setup snippet.

If the <file> parameter of a driver directive does not include a file extension, the following extensions are checked in this order: .EXE, .APP, .PRG, .FXPThe hooks are, with the exception of _SCXDRV2 and _SCXDRV3, executed in numerical order. Drivers are invoked by the hooks in the sequence they are specified. It is important to know which hooks to attach your drivers.

The _SCXDRVx hooks are each called within a SCAN loop, once for every .SCX record. If your driver processes the whole .SCX file (which is faster than making GENSCRNX call your driver for each record), a GOTO BOTTOM command placed just before your driver's RETURN statement will speed up the process by eliminating redundant calls.

As a general rule, most _SCXDRV drivers should be attached to hooks 2, 3, or 5. If your screens use objects swapped in from the FOXSCX file and you want your drivers to affect all the new objects, be sure to use _SCXDRV3 or higher.This is a good time to point out that some native GENSCRNX processes occur both in the _SCXDRV and _SPRVRV phases. This happens for insertion and the *:NAME
commands, among others.

Here are the main events in the GENSCRNX process. A complete detailed description would take many pages. I have tried to limit this to the most notable events.

1 1. Before _SCXDRV1, CONFIG.FP is loaded into a temporary memo field. Any {{}} commands therein are executed. FOXSCX.DBF (or any specified substitute) is opened. The transporter is invoked if required (and if we are building a stand-alone screen). If _GENSCRNX="OFF", GenScrn is called, and we're done. Otherwise, control is passed to the GENSCX() procedure, where the temporary screen and project files are created and the drivers are invoked.

2 _SCXDRV1 drivers are executed. Next, FOXSCX table preparation occurs.

3 _SCXDRV2 drivers are executed.

4 _SCXDRV3 drivers are executed, then library objects and screens are then swapped into your screen as required. If any library objects or library screens are inserted or altered, _SCXDRV2 and _SCXDRV3 are called again as appropriate for each new or altered object. This is why, if you want your drivers to effect all objects, it's wise to use hook _SCXDRV3 or higher.

5 _SCXDRV4 drivers are executed. After this, *:IF processing is initiated, as is *:SIZE, *:NOSIZE, *:DEFAULT.

6 _SCXDRV5 drivers are executed, after which ACTIVATE and DEACTIVATE snippets are consolidated for the screen.

7 _SCXDRV6 drivers are executed for inserted or new objects only. After which we process objects for *:DELETE, *:NAME (part 1), and *:TRNTXT. Invisible placeholder buttons added to the table as first and last items on the screen.

8 _SCXDRV7 drivers are executed. #:SECTION 3 and *:FUNCTION (part 1) effects are processed.

9 _SCXDRV8 drivers are executed, and we are done with pre-processing.

10 GENSCRNX calls GenScrn.PRG, which generates code from the modified temporary copy of your screen created by the previous steps.

11 GENSCRNX loads the code it into a temporary memo file. Any remaining {{}} statements are

evaluated.

12 _SPRDRV1 drivers are executed, followed by the final part of *:NAME processing.

13 _SPRDRV2 drivers are executed, followed by maintenance of the SHOW snippet to ensure refreshable objects refresh properly.

14 _SPRDRV3 drivers are executed, followed by some text insertion maintenanc.

15 _SPRDRV4 drivers are executed.

16 _SPRDRV5 drivers are executed, followed by more text insertion housekeeping.

17 _SPRDRV6 drivers are executed, and, finally, the source code is written to disk.

The GENSCRNX 3D Driver

Beautify your Windows Screens

Introduction

The 3D.PRG driver is designed for GENSCRNX 1.7a or later. The 3D driver is only enabled when building screens in FoxPro for Windows and is disabled when generated code for non-Windows platforms. The files contained in 3DFOX.ZIP are for use with FoxPro 2.5 for Windows, although GENSCRNX.PRG is compatible with all versions of FoxPro 2.x.

3D Topics

3D Basic Installation

The quickest way to get started with generating 3D screens is to place the following two lines of code in the Setup snippet of a screen that has a light gray background:

```
*:SCXDRV5 3D
*:ALL3D
```

To enable the 3D driver globally for all FoxPro for Windows screens, place the following in the CONFIG.FPW:

```
_SCXDRV5="<path>3D.PRG"
```

with the proper <path> of the 3D.PRG driver. Then place the following line of code in the Setup snippet of a screen that has a light gray background:

```
*:ALL3D
```

3D Setup Snippet Directives

Commands that affect an entire screen

****:SCXDRV5 <path>3D***

Specify SCX driver #5 to execute the 3D.PRG driver function.

****:SAY3D [<expN> | INSET | RAISED] [REFRESH] [COLOR <color>]***

Specify 3D effects for all SAY objects.

****:GET3D [<expN> | RAISED] [COLOR <color>]***

Specify 3D effects for all GET objects.

****:EDIT3D [<expN> | RAISED] [COLOR <color>]***

Specify 3D effects for all EDIT objects.

****:CONTROL3D [<expN> | RAISED] [COLOR <color>]***

Specify 3D effects for all control objects. This includes all CHECK BOX, LIST BOX, POPUP, and RADIO BUTTON, and SPINNER objects.

****:PICTURE3D [<expN> | RAISED] [COLOR <color>]***

Specify 3D effects for all PICTURE objects.

****:INSET3D [COLOR <color>]***

Specify 3D inset effect for all BOX and LINE objects.

****:RAISED3D [COLOR <color>]***

Specify 3D raised effect for all BOX and LINE objects.

****:BOX3D [<expN>] [REFRESH] [COLOR <color>]***

Specify 3D beveled box effect using 3DBOX.PRG for all BOX objects.

****:BOX3D [<expN>] [SHADOW] [REFRESH] [COLOR <color>]***

Specify 3D shadow effect for all BOX objects.

****:ALL3D [<expN> | RAISED] [COLOR <color>]***

Specify 3D effects for all above objects. If one of the above directives are included along with *:ALL3D, the specific directive will override the *:ALL3D directive.

:TEXT3D [<expN> | INSET | RAISED] [COLOR <color>]

Specify 3D effects for all TEXT objects.

3D Comment Snippet Directives

Control 3D qualities at the screen object level

**:NO3D*

Disable 3D effect for object. The *:NO3D directive is only used to disable the 3D effect for an object when 3D effects are specified globally in the Setup snippet.

**:3D [<expN>] [COLOR <color>]*

Specify 3D effect for object. For BOX and LINE objects, the 3D inset effect is the default.

**:3D INSET [<expN>] [COLOR <color>]*

Specify 3D inset effect for object

**:3D RAISED [<expN>] [COLOR <color>]*

Specify 3D raised effect for object.

**:3D [<expN>] BOX [SHADOW] [REFRESH] [COLOR <color>]*

Specify 3D beveled box effect for BOX object. <expN> specifies the bevel width. A positive number creates a raised box while a negative number creates an inset box. An optional parameter list can be specified for non-default 3DBOX parameters. Refer to 3DBOX.PRG source code (parameters 6 and on) for detailed information or see examples below.

When specifying *:3D BOX, be sure to use the Screen Builder's Send to Back option for the box. The *:3D BOX directive causes GENSCRNX to replace the @ row1,col1 TO row2,col2 command with a DO 3DBOX WITH <expN> command where <expN> can be a parameter list for parameters 5 through 19 of 3DBOX.PRG, although normally only the bevel width needs to be specified. The 3DBOX procedure call needs to be executed before any @ SAY, @ GET, etc. that need to appear on top of the 3D box which can be achieved by using the Send to Back option.

Examples: To specify the default 3D beveled box:

```
*:3D BOX
```

To specify a 3D box with a raised bevel of 6:

```
*:3D 6 BOX
```

To specify a 3D box with a inset bevel of 6 and to have redrawing during the Read Show clause:

```
*:3D -6 BOX REFRESH
```

To specify a 3D box with an inset bevel of 4, sun color white, and shade color of blue:

```
*:3D -4 BOX COLOR B
```

or

```
*:3D -4 BOX COLOR RGB(0,0,255)
```

or

```
*:3D -4,255,255,255,0,0,255 BOX
```

3D Common Directives

Things that apply everywhere

If <expN> is omitted, the default 3D effect is created for the object.

If <expN> is specified, a shadow 3D effect is created for the object unless the object is a BOX and the *:3D BOX directive is used (refer to *:3D BOX above). The cast of the shadow is determined by the sign of <expN>. For example, if <expN> is 4, a shadow is cast 4 pixels down and to the right of the object. If <expN> is -6, a shadow is cast 6 pixels up and to the left of the object.

If COLOR <color> is omitted, the default 3D shadow color is used for the object. If <expN> is omitted, COLOR <color> specifies the shade color of the chiseled box. If <expN> is specified, COLOR <color> specifies the color of the shadow.

Color can also be specified with a set of 3 RGB (Red Green Blue) color values separated by commas. The color values can range from 0 through 255. For example, the color blue can also be specified with RGB color RGB(0,0,255)

Examples: To specify the default 3D effect:
*:3D

To specify a shadow cast 4 pixels down and to the right using the default shadow color:
*:3D 4

To specify the default 3D effect with a blue shadow color:
*:3D COLOR B

To specify a shadow cast 6 pixels down and to the right with a dark gray shadow color:
*:3D 6 COLOR RGB(128,128,128)

3D Additional Information

Miscellaneous things about the 3D driver

Any cross-platform screens using 3D effects must be built in FoxPro for Windows for the WINDOWS code to be generated properly. The 3D driver program uses the FONTMETRIC() function to determine the row,col, and size information for the 3D objects which requires FoxPro for Windows. Generating cross-platform screens using 3D effects in a platform other than Windows will only effect the code generated for the Windows platform. Therefore, a cross-platform project can simply be re-built in FoxPro for Windows when development is complete.

The 3D driver automatically adds a SET READBORDER OFF command at the end of the Setup snippet if a SET READBORDER ON command or a *:SET BORDERGETS directive does not already exist in the Setup snippet. The easiest way to enable the border gets option when using the 3D driver is to place a SET READBORDER OFF command anywhere in the Setup snippet making sure the command is places starting in column one (not indented). Refer to GENSCRNX documentation for further information about the *:SET BORDERGETS directive.

If a public variable called _3D is set to OFF, the 3D driver will be disabled and all 3D effects will be suppressed. _3D='OFF' is useful when prototyping a project and 3D effects are not needed for testing but the time it takes for a project to re-build needs to be as short as possible. The overhead time of using the 3D driver depends on the number and type of 3D effects generated for the screen. 3D=OFF can be placed in the CONFIG.FPW file to disable 3D effects without being dependent on a public variable call _3D existing. If both a public variable _3D and 3D=ON | OFF in the CONFIG.FPW file, the public variable has precedence.

Objects containing the GENSCRNX directives *:SIZE or *:NOSIZE in the Comment snippet will not generate 3D effects.

GENSCRNX Drag and Drop

An Important Interface Innovation

Help for this not implemented yet.

Steven Black Consulting

Specializing in unique, tough development projects since 1985

2 Bay St., Kingston ON, K7K 6T7 CANADA
613.542.3293 (voice) 613.531.9481(fax) 76200,2110 (CompuServe)

The Steven Black Consulting Company has coordinated and executed xBase development and implementation projects throughout the world. We specialize in FoxPro development in the following areas:

1. Multilingual Software Development

Steven Black is the creator of the Steven Black's INTL Toolkit, a framework for FoxPro to easily create and maintain multilingual and multi-version programs from a unilingual single-version codebase.

2. Tough Projects

Steven Black Consulting has a proven track record managing and participating in delicate software projects where, for whatever reason, success is crucial and obstacles loom large. We have worked successfully and seek to continue working in the following types of situations

- "Out-of-control project" turnarounds
- Sensitive situations
- Hostile environments
- Multi-cultural applications
- Large-scale efforts
- Team efforts, as leaders or dedicated followers.

A job we can't do probably can't be done.

If you are a FoxPro expert, with excellent people and project management skills, then please call.

GENSCRNX — What is it?

It's an amazing tool...

GENSCRNX is a program that enhances FoxPro's standard screen enerator program. It was developed by Ken Levy at the Jet Propulsion Laboratory in Pasadena California, and released into the public domain. GENSCRNX is available and supported on CompuServe in the FoxForum.

GENSCRNX is a screen generator shell that plays on the inputs and utputs of GenScrn.PRG (or whatever generator program you use). To understand GENSCRNX, please refer to Figure 1 below:

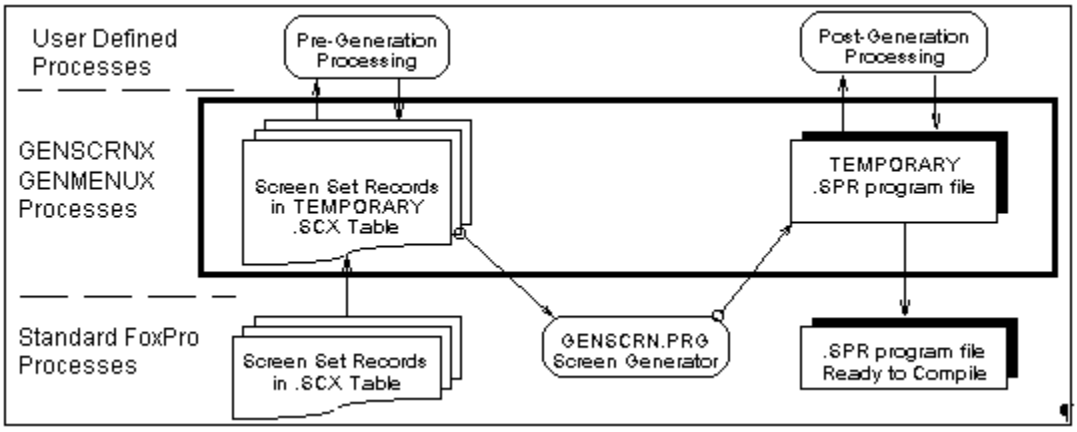

Figure 1 -- GENSCRNX first makes a COPY of the screen file, which is then processed with custom commands and "driver" programs. After standard GenScrn, the result is stored in a temporary memo for more processing before the source is finally written to disk.

*:BASLIB

Base library name for field name match.

*:BASOBJ

Specify a base library object from which this object is to draw its attributes.

*:DEFLIB

Define a library name under which to store all the objects in a library screen set.

***:DEFOBJ**

Defines a library object name

*:INCLIB

Include a library in the base object search path for objects.

***:INSOBJ**

Substitute an object from FOXSCX.DBF in place of this object.

***:INSSCX**

Substitute a screen from a library in place of this screen object.

***:INSTXT**

Marks the beginning of code to be added in the place of this screen object.

3D

A public domain driver available in the CompuServe Foxforum libraries which adds 3-D visual effects to Foxpro for Windows and FoxPro for Macintosh screens.

FOXSCX
The default GENSCRNX library object repository.

GENMENUX

A public domain GenMenu.PRG shell created by Andrew Ross MacNeill of Ottawa, Ontario Canada. You can reach Andrew at 76100,2725.

GENSCRNX

A public domain GenScrn.PRG shell created by Ken R. Levy of the Jet Propulsion Laboratory in Pasadena California. You can reach Ken at 76350,2610.