#243: Script Manager Variables

Written by: John Harvey & Peter Edberg June 1989

This Technical Note describes, in detail, the local and global script variables.

Introduction

The Script Manager maintains a number of global variables which can be read with the routine `_GetEnvirons`. These variables can be set by a corresponding routine, `_SetEnvirons`. In addition, each script interface system maintains variables of its own. These are referred to as local variables in Inside Macintosh, Volume V-293, The Script Manager, and are read by `_GetScript` and set by `_SetScript`.

Think of it like this: the Script Manager maintains an environment in which different script interfaces can run. The global variables are used to set up and maintain the environment (thus the names for the routines `_GetEnvirons` and `_SetEnvirons`), and the local variables control how the script itself works (so we have `_GetScript` and `_SetScript`).

Global Variables

When you call `_GetEnvirons` or `_GetScript`, you describe the variable you are interested in with a verb. A verb is simply an integer constant which the Script Manager uses to figure out which variable you want to read or set. The table in *Inside Macintosh*, V-313, gives incorrect names and descriptions for some of the `_GetEnvirons` and `_SetEnvirons` verbs. Table 1 provides correct descriptions.

Constant	Value	Meaning
smVersion	0	Script Manager version number
smMunged	2	Global modification count
smEnabled	4	Script count; 0 if Script Manager not enabled
smBidirect	6	Bidirectional script flag
smFontForce	8	Force font flag
smIntlForce	10	Force international utilities flag
smForced	12	Current script forced to system script
smDefault	14	Current script defaulted to Roman script
smPrint	16	Print action vector
smSysScript	18	Preferred system script
smLastScript	20	Last keyboard script
smKeyScript	22	Keyboard script
smSysRef	24	System folder volRefNum
smKeyCache	26	[Obsolete, do not use]
smKeySwap	28	Keyboard swapping resource handle
smGenFlags	30	General flags
smOverride	32	Script override flags
smCharPortion	34	Ch vs Sp Extra proportion, 4.12 fixed

Table 1–Verbs for _GetEnvirons and _SetEnvirons

The descriptions in the table are still a bit sketchy. The next section describes each variable in more detail and describes the size of each global.

Byte or word globals are mapped to the low-order byte or word of the LongInt returned by _GetEnvirons, with the high-order parts set to zero. Similarly, for these globals _SetEnvirons ignores all but the appropriate part (low-order byte or word) of its params value.

Verb Name	Bytes	Brief Description
smVersion	2	Script Manager version number

At boot time, the version global is initialized to the value SMgrVers. The high byte is the major version number and is defined in the MPW interface files. The low byte is updated when any changes are made to the Script Manager.

smMunged	2	Global modification count

The munged global is initialized to zero at boot time and incremented when:

- _KeyScript changes the key script and updates smKeyScript and smLastScript
- _SetEnvirons is used to change a Script Manager global

smEnabled	1	Script count; 0 if Script Manager not enabled

At boot time or switch-launch time, the enabled global is initialized to zero, then incremented for each script that is installed and enabled. Since the Roman script system should always be installed by the Script Manager, a value of zero indicates that the Script Manager is not enabled.

It should be noted that older versions of the Script Manager treated this as a `Boolean`. In other words, if there was more than one script installed, `_GetEnvirons(smEnabled)` would return 255 (when `_GetEnvirons` returns a `Boolean` value $FF represents true).

For this reason, when testing to see if more than one script is installed, it is best to test as follows:

```
scriptsinstalled := GetEnvirons(smEnabled);
IF scriptsinstalled > 1 THEN
{more than one script available, use Chartype, etc.}
```

`smBidirect`	1	Bidirectional script flag

The bidirectional global indicates that at least one bidirectional script is installed. It should be set to true ($FF) by the Arabic and Hebrew script systems. This is not presently done, but will be corrected in future versions of these systems.

`smFontForce`	1	Force font flag
`smIntlForce`	1	Force international utilities flag
`smForced`	1	Current script forced to system script
`smDefault`	1	Current script defaulted to Roman script

At boot time, `FontForce` and `IntlForce` are set from the `'itlc'` resource, and `Forced` and `Default` are set to zero. These are all flags with the value zero for false and $FF for true. `FontForce` and `IntlForce` control the operation of the `_FontScript`, `_Font2Script`, and `_IntlScript` routines. `Forced` and `Default` report the actions of these routines.

Setting `FontForce` to true forces Roman fonts to be interpreted as belonging to the system script. This is for compatibility with applications that hard-code font numbers.

`IntlForce` determines the behavior of the `_IUGetIntl` call. When `intlforce` is set to true, `_IUGetIntl` will return a handle to the international resources (of type `'itlx'` where `x` is 0-2) for the system script. When `IntlForce` is false, the `_IUGetIntl` will use the font of the current port to determine the appropriate resources to fetch. Thus date formats, sorting, etc. can reflect the current script.

`smPrint`	4	Print action vector

Print action routine vector; set up at boot time. See Technical Note #174, Accessing the Script Manager Print Action Routine.

`smSysScript`	2	Preferred system script
`smLastScript`	2	Last keyboard script
`smKeyScript`	2	Keyboard script

At boot time and switch-launch time, `SysScript` and `KeyScript` are set from the `SysScript` field of the `'itlc'` resource if that script is installed and enabled; otherwise, `SysScript` and `KeyScript` are set to Roman (without setting `Default`).

The KeyScript global is the current keyboard script, tested and updated by the _KeyScript routine. When _KeyScript changes KeyScript, it moves the old value to LastScript. _KeyScript can also swap the current key script with the last one, which it retrieves from LastScript. The KeyScript value is also used to get the proper keyboard script icon and to retrieve the proper 'KCHR'.

SysScript specifies the system script, and is used, for example, by _FontScript, _Font2Script, and _IntlScript.

KeyScript, LastScript, and SysScript always contain integers that correspond to a script number. Script numbers are documented in The Script Manager chapter of *Inside Macintosh*, Volume V-293.

smSysRef	2	System folder volRefNum

Set from the global BootDrive at boot time and switch-launch time. SysRef was originally a way of testing for vanilla launch versus switch launch; now the Enabled global is used for that purpose.

smKeyCache	2	[Obsolete, do not use]
smKeySwap	4	Keyboard swapping resource handle

The 'KSWP' resource handle is put here at boot time and switch-launch time. A 'KSWP' resource contains a table of key sequences that will cause the currently installed 'KCHR' (keyboard mapping table) to change to the preferred system 'KCHR', switch to the Roman 'KCHR', or rotate among the available 'KCHR' resources. The table includes the virtual key code and the modifier keys. The following is the 'KSWP' resource for the Kanji script interface system.

```
resource 'KSWP' (0, sysheap) {
       {/* array: 3 elements */
              /* [1] */
              Rotate, 49, controlOff, optionOff, shiftOff, commandOn,
              /* [2] */
              System, 70, controlOff, optionOff, shiftOff, commandOn,
              /* [3] */
              Roman, 66, controlOff, optionOff, shiftOff, commandOn
       }
};
```

The resource says rotate 'KCHR' resources if a Space–Command key occurs, switch to the system 'KCHR' on keypad plus (+)–Command key, and switch to the Roman 'KCHR' on keypad asterisk (*)–Command key.

smGenFlags	4	General flags

Only the two high-order bits are defined (in the file ScriptEqu.a), as follows:

smfShowIcon = 31	(show icon even if only one script)
smfDualCaret = 30	(use dual caret for mixed direction text)

The high-order byte of smgrGenFlags, containing these flags, should be setup from the flags byte in the 'itlc' resource. This is not presently done, but will be fixed in future versions of the Script Manager.

The following MPW Pascal procedure demonstrates how to get script 'SICN' resources to display even if there is only one script system installed.

```
PROCEDURE SetSICN;
VAR
       SICNstate: Longint;
       err: Oserr;

BEGIN
       SICNstate := GetEnvirons(smGenFlags);
       BSET(SICNstate,smfShowIcon);

       err := SetEnvirons(smGenFlags,SICNstate);
END;
```

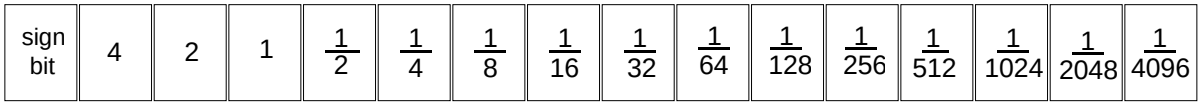
`smOverride`	4	Script override flags

At present, this is not set or used by the Script Manager. It is, however, reserved for future improvements.

`smCharPortion`	2	Ch vs Sp Extra proportion, 4.12 fixed

This is 16-bit fixed-point value in 4.12 format (e.g., 10% = $0199). It is initialized to 10 percent at boot time. It is intended to be used by script systems to allocate space among intercharacter spacing and interword spacing when justifying text.

A 16-bit fixed-point value in 4.12 format is similar to the fixed-point number type defined on page I-79 of *Inside Macintosh* . The obvious difference being that it is only 16 bits long. The integer part of the value is stored in the high four bits, and the fractional part is stored in the low 12 bits.



16-bit Fixed-Point Number in 4.12 format

Local Variables

Every script interface system has local variables. Page V-132 of *Inside Macintosh* lists verbs which are constants that indicate which variable you want to read or set. The table of constants used to access the local variable, although more accurate than the global table, does contain a few inaccuracies. In addition four new constants have been added. Table 2 gives the correct constants.

Constant	Value	Meaning
`smScriptVersion`	0	Script Interface version number
`smScriptMunged`	2	Local modification count
`smScriptEnabled`	4	Script Enabled Flag
`smScriptRight`	6	Right to Left Flag
`smScriptJust`	8	Justification Flag
`smScriptRedraw`	10	Word Redraw Flag
`smScriptSysFond`	12	Preferred System Font
`smScriptAppFond`	14	Preferred Application Font

smScriptNumber	16	Script 'itl0' ID
smScriptDate	18	Script 'itl1' ID
smScriptSort	20	Script 'itl2' ID
smScriptFlags	22	Script Flags Word (**new**)
smScriptToken	24	'itl4' ID number (**new**)
smScriptRsvd	26	Reserved
smScriptLang	28	Script's language code (**new**)
smScriptNumDate	30	Number/date representation codes (**new**)
smScriptKeys	32	Script 'KCHR' ID
smScriptIcon	34	Script 'SICN' ID
smScriptPrint	36	Script printer action routine
smScriptTrap	38	Trap entry pointer
smScriptCreator	40	Script file creator
smScriptFile	42	Script file name
smScriptName	44	Script name

Table 2–Local Variable Constants

Here again the descriptions are a little terse. The following section describes each variable in more detail and describes the size of each variable.

Verb Name	Bytes	Brief Description
smScriptVersion	4	Script Interface version number

When the script interface is loaded, this is set to the current version number.

smScriptMunged	2	Local modification count

This variable is incremented each time _SetScript is called.

smScriptEnabled	1	Script Enabled Flag

A Boolean which indicates whether the script has been enabled. Set to $FF when enabled and zero when not enabled.

smScriptRight	1	Right to Left Flag

A Boolean indicating if text should be drawn right to left or left to right. It is set to $FF for right to left text (Arabic and Hebrew scripts) and zero for left to right (Roman).

smScriptJust	1	Justification Flag

A byte flag which describes how text should be justified. The possible settings correspond to the justification flags used by TextEdit.

0 = left justification
1 = center justified
-1 = right justified

`smScriptRedraw` 1 Word Redraw Flag

A byte flag describing how much of a line should be redrawn when text is being entered.

0 Only draw a character
1 Redraw the entire word
-1 Redraw the entire line (Arabic)

`smScriptSysFond` 2 Preferred System Font

This is the font family ID for the preferred System Font. In a Roman system, `ScriptSysFond` is 0, the family ID for Chicago.

`smScriptAppFond` 2 Preferred Application Font

Font family ID for the preferred Application Font. In a Roman system, `ScriptAppFond` is 3, the family ID for Geneva.

`smScriptNumber` 4 Script `'itl0'` ID

Resource ID of `'itl0'` for this script. The `'itl0'` resource describes how numbers and times should be displayed. The resource ID should match the country version code for a given country.

`smScriptDate` 4 Script `'itl1'` ID

Resource ID of the `'itl1'` for this script. The `'itl1'` describes how dates should be displayed.

`smScriptSort` 4 Script `'itl2'` ID

Resource ID of the `'itl2'` for this script. The `'itl2'` contains routines for sorting. See Technical Note #178.

`smScriptFlags` 2 Script flags

This verb provides access to the script flags word, which contains bit flags that describe features of the script. This word is initialized from the script's `'itlb'` resource. Constants specifying the bit numbers are described in Table 3.

Constant	Bit Number	Description
`smsfIntellCP`	0	script has intelligent cut and paste
`smsfSingByte`	1	script has only single bytes
`smsfNatCase`	2	native characters have upper and lower case
`smsfContext`	3	contextual script (e.g., AIS-based)
`smsfNoForceFont`	4	will not force characters
`smsfB0Digits`	5	has alternative digits in B0-B9
`smsfForms`	13	uses contextual forms for letters
`smsfLigatures`	14	uses contextual ligatures
`smsfReverse`	15	reverses native text, right-left

Table 3–Constant Bit Numbers

`smScriptToken` 2 Script `'itl2'` ID

Resource ID of the `'itl4'` for this script. The `'itl4'` contains contains tables needed by the number formatting and conversion routines and the `_intlTokenize` routine. See Script Manager 2.0, Interim Chapter.

`smScriptRsvd` 4 Reserved

`smScriptLang` 2 Script's language code

This verb accesses a word which contains the current language code for the script. The language codes are defined in the MPW interface files.

`smScriptNumDate` 2 Number and date representation codes

This verb accesses a word containing the number and date representation codes for the script. The number representation code is in the high byte of the word, and the date code is in the low byte.

The possible values for number representations and date codes are declared as constants in the MPW interface files.

The number codes are:	and the date codes are:
intWestern = 0;	calGregorian = 0;
intArabic = 1;	calArabicCivil = 1;
intRoman = 2;	calArabicLunar = 2;
intJapanese = 3;	calJapanese = 3;
intEuropean = 4;	calJewish = 4;
	calCoptic = 5;

`smScriptKeys` 4 Script `'KCHR'` ID

Resource ID of preferred `'KCHR'` resource. The `'KCHR'` resource is used to map virtual key codes into the correct character code. See Technical Note #160.

`smScriptIcon` 4 Script `'SICN'` ID

Resource ID of the small icon that is used to represent which country specific resources (`'itl0'`, `'itl1'`, `'itl2'`, `'KCHR'`) are currently installed in the system. Presently, the Roman system does not display the `'SICN'`. Arabic, Kanji, Chinese, and Hebrew interface systems do display this icon in the upper-right corner of the menu bar.

`smScriptPrint` 4 Script printer action routine

Print action routine vector; setup when script is installed by Script Manager. See Technical Note #174, Accessing the Script Manager Print Action Routine.

`smScriptTrap` 4 Trap entry pointer

Pointer to Script dispatch routine. Script Manager routines always belong to one of two groups. The first group of routines are common to every script interface system, and the second group must be supplied by the script interface system. This variable will point to a dispatch routine for the interface-supplied routines. When you call `_ScriptUtil`, it looks at the selector that is passed and either calls a common routine or calls the routine whose address is stored in `ScriptTrap`. The routine in `smScriptTrap` will then use the selector to vector to the correct routine. In general, routines that display or measure text in some way will be supplied by the interface.

A list at the end of this Note indicates which routines are implemented by the Script Manager and which routines are supplied by a script interface system.

`smScriptCreator` 4 Script file creator

The four character creator type for the script interface's file. For Roman it is "ZSYS," the same creator as any system file has.

`smScriptFile` 4 Script file name

A pointer to the a Pascal string which contains the name of the file containing the script interface system. For the Roman SIS, it is System.

`smScriptName` 44 Script name

A pointer to a Pascal string which contains the script interface's name. For Roman it is naturally, "Roman."

Who Does What?

Table 4 breaks the documented routines into common Script Manager routines and interface specific routines.

Common Routines	Interface Supplied
_FontScript	_CharByte
_IntlScript	_CharType
_KeyScript	_Pixel2Char
_GetEnvirons	_Char2Pixel
_SetEnvirons	_Transliterate
_Font2Script	_FindWord
_Format2Str	_HiliteText
_FormatStr2X	_DrawJust
_FormaX2Str	_MeasureJust
_GetFormatOrder	_ParseTable
_InitDateCache	_VisibleLength
_IntlTokenize	_FindScriptRun
_LongDate2Secs	_PortionText
_LongSecs2Date	
_Str2Format	
_String2Date	
_String2Time	
_StyledLineBreak	
_ToggleDate	
_ValidDate	

Table 4–Script Manager Routines and Interface Specific Routines

`_GetScript` and `_SetScript`, which return the values of local script variables, are implemented by the Script Manager for some verbs and the script interface system for others.

There is also a group of Script Manager routines which don't use the `_ScriptUtil` trap, but are documented in The Script Manager chapter of *Inside Macintosh*, Volume V-293 or The Script Manager 2.0 Interim Chapter. There routines are utilities that read and write to low-memory or PRAM. It is important to use these routines when they are available. That will allow Apple to modify where global variables, etc. are stored, and your application will remain compatible. The utilities are:

```
GetDefFontSize
GetSysFont
GetAppFont
GetMBarHeight
GetSysJust
SetSysJust
ReadLocation (documented in Interim Chapter)
WriteLocation (documented in Interim Chapter)
```

Further Reference:

- *Inside Macintosh*, Volume V-293, The Script Manager
- Technical Note #160, Key Mapping
- Technical Note #174, Accessing the Script Manger Print Action Routine