Route 1 Script Editor - version 4.1

Choose one of the following topics for details and information on Script Editor's many versatile features:

If you are new to scripts, and find the thought of writing one overwhelming, press the green question mark button (initially configured upon installation of Route 1), or launch the file TUTORIAL.RS1 to run a tutorial about Script writing, or another Route 1 feature.

Script Commands

The following commands can be placed in a script in any order to perform various tasks. See Create a New Script for more information on opening the Script Editor and editing a Script. See Script Hints in Route1.HLP for some ideas. These commands can be typed in manually, or can be entered automatically by pressing the corresponding button on the command button bar. Every line in a script must be a valid command, comment, variable declaration, label, or blank space. Any line beginning with a space, such as that which has been indented will be treated like blank line, and ignored. The following are valid commands that can either be typed from the keyboard, or inserted with the buttons in the Script Editor:

Ask
Input
Goto
Type
Wait
If, If Exist
Title
List
Beep
DDE Commands
Status
Copy
Name
Delete

Exit - include the command *Exit* or *End* to stop execution of the running script and close the Script runner.

Note: ignore the ">" character shown above when typing commands; it is used to denote the insertion mechanism performed by pressing the command's button in the Script Editor.

Comments:

Any comments you wish to include in your scripts should start with the **'** or **;** or ***** characters. These comments are for the script writer's use, and will be treated as a blank line and ignored when the script is executed.

Commenting your scripts is encouraged, especially for long or complex scripts, and aids in later revision.

Using Scripts

The following topics describe different aspects of scripts and their uses with Route 1:

While entering a script, you may wish to test it for errors, or to see if it functions as desired. To test a script, first save the script you're working on, and then launch it with Route 1's [**<**] button.

With the exception of those included with Route 1, scripts are in no way the property or responsibility of the author of Route 1. However, if you have written a script that you believe is especially useful, interesting, or innovative, send it to the author for consideration of publication with future releases of Route 1.

If, If Exist

Purpose: decides whether or not to execute a particular command depending on a situation

Syntax #1: If %*n* ="*expression*" then *command*

where n is a whole number ranging from 1 to 9, signifying a variable.
where expression is whatever %n might be equal to.
where command is any valid script command.

Syntax #2: If Exist "*filename*" then *command*

where command is any valid script command.
where filename is any valid file name, including its path.

Remarks: in the above statement, *command* will be executed if and only if the variable (%*n*) is exactly equivalent to *expression*, whether *expression* is text, or a number. If *expression* is a text string, then the contents of the variable must match exactly, case included. In the case of *If Exist*, *command* will be executed if and only if *filename* exists; *filename* can include the drive and path as well. If a path is not specified, the current directory will be searched. Route 1 will not search those directories listed in your path statement (see your DOS manual), nor will it search the Windows or System directories (unless specified).

Wait

Purpose: pauses script execution for a specified amount of time

Syntax: Wait(*n*)

where n is any number greater than 0, representing seconds

Remarks: use this to wait *n* seconds while a program loads before executing the next command, or for timing functions. See the included script, TIMER.RS1 for an example. Wait has no effect on other running applications or tasks, including Route 1's clock, calendar and memory display.

Type

Purpose: sends keystrokes to the active Windows application, as though they were typed from the keyboard

Syntax: Type"*expression*"

where expression is any set of valid keystrokes to type.

Remarks: If used correctly and carefully, this command can take control of any Windows application. It can not type keys to a non-Windows application. *Expression* can consist of anything, although the following reserved characters have special meanings:

+ for SHIFT
^ for CONTROL
% for ALT

Use these symbols with others, such as ^J would send Ctrl-J to the active window. To use these characters normally, enclose them in parenthesis, such as {%}. If you use the **>Type** button, you are given a large list of some possible special combinations. Select one and press **>Insert**, or press **Close** to type your own.

For best results, use the included utility, PUTFOCUS.EXE, to transfer the focus to any desired running program before sending keystrokes. Simply add the following line to transfer the focus to "*application*":

PutFocus *application*

where application is the name that appears in the title bar of the Windows application to receive the focus.

What follows is a simple example of PUTFOCUS used in conjunction with the Type command:

```
PutFocus Program Manager
Type"% n"
```

This simple script first activates Program Manager (if it is not loaded, Route 1 will return an error), actives its system menu, and types "n" to minimize it.

Note: You cannot send keystrokes to non-Windows application. In addition, to preserve the functionality of the "Alt" keystroke code, "%", *expression* cannot contain any variables.

Note: Make sure you separate any codes in this command, so that a "}{" is not present, since this is translated as a carriage return (used mostly for the Ask and Input commands).

Goto

Purpose: transfers execution of the script to another location in the script, designated by a label

Syntax: Goto *label*

where label is a word designated elsewhere as a label by beginning and ending it with **:**

Remarks: use Goto to repeat or skip any portion of the script. Although both colons at the beginning and the end of a label are not required (at least one is necessary), they are both recommended to avoid confusion with Script Commands. *Label* should not contain a colon (:) in the Goto line, but should be included in the actual label:

```
.                    (here are normal commands to be executed)
.
Goto Skip
.                    (this portion will be ignored)
.
:Skip:
.
.                    (these commands are executed)
```

Input

Purpose: displays a message and prompts the user to enter data or text.

Syntax: Input(%*n*)"*text*"

where text is a line of text to be displayed to the user
where n is a number ranging from 1 to 9, signifying a variable in which the user's response is stored.

Remarks: This command allows the user to interact with or interrupt the running script. Input displays *text*, with two buttons: **Ok** and **Cancel**. Pressing **Ok** will continue the script, assign the user's entered data or text to *%n*, and pressing **Cancel** will abort the script. Including the code **}{** will insert a carriage return (line feed), enabling multi-line text.

Ask

Purpose: displays a message and prompts the user to continue or cancel execution of the script

Syntax: Ask"*text*"

where text is a line of text to be displayed to the user

Remarks: This command allows the user to interact with or interrupt the running script. **Ask** displays text, with two buttons: **Ok** and **Cancel**. Pressing **Ok** will continue the script, and pressing **Cancel** will abort the script. Including the code **}{** will insert a carriage return (line feed), enabling multi-line text.

Beep

Purpose: makes a beep through the PC speaker, or configured sound board.

Syntax: Beep

Remarks: use this to alert the user that something is going to happen, or something has already happened. Beep displays no visual message, but can be used in conjunction with Ask or Input to direct the users attention to the message. If using Ask or Input, place the beep first; otherwise the beep will not sound until the user has responded.

Title

Purpose: sets a title to be displayed in all dialog boxes while running the script

Syntax: Title"*text*"

where text is a line of text to be displayed to the user

Remarks: by default, the Script Runner sets the title-bar text in all message boxes, list boxes, and status windows to the name of the running script. This command changes the text to be displayed to *text*. Use this before opening a message box, list box, or status window - the *Title* command will not change the text of any boxes already opened.

Variables

All of the Route 1 script commands work with variables, which are designated by %n, where n is a whole number, ranging from 1 to 9. Variables can contain anything, such as text or numerical values. Variables can be assigned by either the Input command, or with an equals sign:

```
%1=3          (assigns the numerical value of 3 to the variable %1)
%4=+2         (adds 2 to the current value of %4)
%3=-2         (assigns -2 to %3)
%6=+-2        (subtracts 2 from the current value of %6)

%2="Howdy"    (assigns the text "Howdy" to %2)
%2=+" Doody"  (adds " Doody" to %2 to make "Howdy Doody")
```

Variables can be put in place of anything, for example:

```
Wait(%3)
```

or

```
Status "This is line %3"
```

Note: one or more variables can be placed in the arguments of any script command, except "Type", since the "%" character is used to denote "Alt".

Special Variables (functions):

There are eight special read-only variables that contain information, but cannot be assigned to anything else. They include:

%d - the current date in the format specified by Control Panel
%t - the current time in the format specified by Control Panel
%y - the current day; sun, mon, tue, wed, thu, fri, sat
%r - Route 1 directory
%w - Windows directory
%s - Windows\System directory
%0 - any command line parameters passed to the script when it was first run. This includes any files dropped onto a Route 1 button containing the script, when the drag-drop option is set to "Launch App and Open File".
%f(*filespec*) - returns a filename from the current directory matching filespec (use *.* for all files). Repeat this function recursively (over and over) until an empty string is returned to obtain a listing of all the files in the current directory. When a different filespec is specified, the "search" is reset, and begins from the top.

Note: including }{ on any line will be interpreted as a carriage return, although it is intended for use with the Ask and Input commands.

DDE Commands

The following symbolic commands are executed by sending commands through DDE (dynamic data exchange) to Route 1. Route 1 **must** be running and not busy in order to use these commands. Since these commands are issued through communications, there might be a delay in their execution. If you encounter some trouble using these commands, try issuing a **wait** command after *each* one (for perhaps a second), to give Route 1 time to process them.

> command

Purpose: used to launch an application or another script.

Syntax: >*path*|*appname*|*...*
Syntax: >*appname*

where path is any existing drive and directory
where appname is the name of an existing application to load

Remarks: Press **Browse...** and select a file, or type the command manually. This button works in the same way that the Browse button works when you add or edit a button in Route 1, such that the appropriate command is entered for you when you select a filename and path. Obviously, it is advisable to use this only on scripts that you are using for your own use, as another Route 1 user will most likely not have the same files and directories that you do. The "**|**" character is used to separate appnames from each other and from "**>**" commands. A "**>**" character on this line is used to change the working directory. For more on the **>** and **|** symbols, refer to "Button Properties: Command Strings" in the Route 1 help file.

note: although the ">" character is used to change directory here, and in Route 1 command strings, it is still required at the beginning of any line to start a program, even if there is no actual change directory command. (see second *Syntax* example)

& command

Purpose: used to switch Route 1's active button bar configuration file (INI file).

Syntax: &*filename*

where filename is an existing Route 1 configuration file that has been created in the Options Box.

Remarks: See Changing Settings in this help file and Script Hints in the Route 1 help file for help on creating and switching between different configurations. This command can also be used straight from Route 1, without using a script. See the "Make INI List" command in the Route 1 SubMenu editor.

* command

Purpose: used to add a new button to Route 1.

Syntax: **filename*iconname*

where filename is the path and filename of the application or script to associate with the new button.

where iconname is the path and filename of the icon file for the new button.

Remarks: the button is added to the end of the current ButtonBar. See Adding a new button in Route1.HLP for more information.

command

Purpose: used to change the position of the ButtonBar.

Syntax: #*n*

where n is the number, from 1 to 4, representing the position of Route 1's ButtonBar.

Remarks: this position is not automatically saved to Route1.INI when this command is invoked. The values of *n* represent the following:

1 - vertical on the right side of the screen
2 - vertical on the left side of the screen
3 - horizontal on the right side of the screen
4 - horizontal on the left side of the screen

^ and + commands

Purpose: collapse and un-collapse the ButtonBar

Syntax: ^ or +

Remarks: ^ collapses Route 1, and + un-collapses Route 1.

Q command

Purpose: quits (closes) Route 1

Syntax: Q

Remarks: this closes Route 1 instantly unless password protection is enabled. If the "Exit Route 1 Exits Windows" advanced option is set, Route 1 will exit Windows.

X command

Purpose: exits Windows

Syntax: X

Remarks: this exits Windows instantly, unless password protection is enabled, or another application refuses.

R command

Purpose: restarts Windows

Syntax: R

Remarks: this restarts Windows instantly, unless password protection is enabled, or another application refuses.

Status

Purpose: controls the Script Runner's status display, used for showing text and displaying information during script execution.

Syntax: Status ON | OFF | "*expression*"

where ON turns on the display
where OFF turns off the display
where expression is the text string to display

Remarks: use this to display to the user something without affecting script operation, such as the amount of time remaining in a count-down timer. Including the code **}{** will insert a carriage return (line feed), enabling multi-line text, although text in the status box is automatically wrapped.

Using the Script Editor

Although the Script Editor is a separate program, it is still an integral part of Route 1 Script (*.RS1) development. Choose one of the following for more details to get you started.

Inserting Commands

Editing and creating scripts with the Script Editor takes place in the text box, above which is a horizontal row of eight buttons (twelve if **Advanced** is checked in the **Tool** menu). Each button represents a script command, although not all commands have respective buttons. For more details on specific commands, see Script Commands.

To insert a command, simply press the desired button. A blank line will be inserted after the current text-cursor position for the command. With some commands, a dialog box will appear, prompting for pertinent information or data. When the appropriate command has been inserted into the script, the text-cursor will be automatically placed where more text is to be entered.

If commands are entered using only these insertion buttons, errors are less likely to occur when running the script.

Edit an Existing Script

Select **Open** from the **File** menu to open a blank script. Select **Save** from the **File** Menu to save your script. You can open as many concurrent scripts as you like until you run out of memory.

The selected script is then opened for editing in a text box, with the usual **Cut**, **Copy**, and **Paste** commands.

Create a New Script

Select **New** from the **File** menu to open a blank script. Select **Save As** from the **File** Menu to save your script. You can create as many new scripts as you like until you run out of disk space.

Hints & Suggestions

There are two scripts (*.RS1) included with Route 1, their names and descriptions follow:

TIMER.RS1 - the Countdown Timer

This is a sample script exemplifying many Script commands and techniques. An example listing follows below. The blue lines are only for clarification

```
(ask for number of seconds to time)
      :Begin:
      Input(%1)"Enter timer duration: (in seconds)"
(perform the timer)
      :TimeIt
      Status "Waiting %1 seconds..."
      Wait(%1)
(alert the user)
      Status "Time is up!"
      %2=0
      :Repeat:
      Beep
      %2=+1
      wait(0.1)
      If %2="20" Then Goto Skip
      Goto Repeat
(give three options for repeat or quit & check response)
      :Skip:
      Status "Time is up!"
      wait(1)
      List(%3)"Repeat with same time|Enter new time|Quit"
      If %3="-1" then goto Quit
      If %3="0" then goto TimeIt
      If %3="1" then goto Begin
      If %3="2" then goto Quit
(end the script)
      :Quit:
      Status OFF
```

This script first asks the user to input the desired delay, then it delays for that amount. Then, using the if command, beeps 20 times.

STARTUP.RS1 - the StartUp Script

This file is initially empty, but can be filled with any commands or application names. If the Run StartUp Script option is turned on, Route 1 will run STARTUP.RS1 when it is first loaded, and if Route 1 is the shell or specified on the LOAD= line, the script is run when Windows is first loaded.

The second line of WIN.INI begins with "LOAD = ". What follows is a list of programs that are loaded when Windows is first started. The inherent problems with this method are as follows:

1. You are limited to 127 characters for all programs. This is enough room for three or four programs. If you have After Dark®, Clock, and of course, Route 1 on this line, you probably can't fit any more.

2. You can't have any command-line parameters. If you start a program with Program Manager, like Word for Windows, you can have it automatically load a document by typing: WINWORD myfile.DOC. WIN.INI won't allow for this either.

3. You have to edit WIN.INI manually, and risk screwing up something else.

4. Windows 3.1 tries to fix this problem, but requires Program Manager, contributes to desktop clutter, and takes up more memory and disk space.

5. There are no conditional statements or multiple configurations allowed.

Route 1 has a built in function that eliminates all this hassle. Following the same powerful, simple, straight-forward command conventions as Route 1 buttons, SubMenus and scripts, you can quickly create a StartUp Script. Simply create a script, and call it StartUp.RS1. Then open Route 1's options box, and turn "Run StartUp Script" on. The script will then be run automatically, when Windows starts, and Route 1 is loaded. In addition to starting programs, you can run other script commands as well, allowing user interaction for multiple boot-ups and conditional statements.

With the inclusion of these two samples included with Route 1, scripts are in no way the property or responsibility of the author of Route 1. However, if you have written a script that you believe is especially useful, interesting, or innovative, send it to the author for consideration of publication with future releases of Route 1.

Associate with File Manager

If a launched file is not an application, it's extension is checked with the [Extensions] sections of WIN.INI for an associated application to launch with it, only if Route 1's advanced option is set. You can use File Manager is associate files with a particular extension with an application, or you can use a text editor (like Notepad) to edit WIN.INI.

Launch with Route 1

Using Route 1's **[<]** button, any script (*.RS1) can be launched like an ordinary application. Scripts can also be assigned to Buttons and SubMenus. While launching or browsing the hard disk, simply select a script file from any directory, and press **Accept**. Note: in order to run a script from an application other than Route 1 (such as File Manager or XTree for Windows) RS1 files must be associated with RS1RUN.EXE.

About Script Editor

A script (with the extension *.RS1) is a file containing a collection of commands, that can be run by Route 1. The commands are executed sequentially (by the script runner, RS1RUN.EXE), and can be assigned to a button or SubMenu, or can be launched by Route 1. Scripts can perform complex operations, simple math, and can be interactive or run in the background.

To create or edit a script, open the Script Editor by selecting Apps from the Configure menu, and pressing the "Script Editor..." button. A new window appears, allowing you to open an existing script to edit, or to create a new script.

Once the editor is visible, you can type in any valid commands, one on each line. You can have as many commands as you wish, as long as the script file's size does not exceed about 32k bytes. See Script Commands for a list and explanations of valid commands. Commands can be entered either by typing them, or by inserting them by pressing the corresponding button. Pressing the button will automatically enter a command in the correct syntax, allowing you to enter in the specifics. You will be prompted for certain parameters when applicable.

This is the third release of the Script Editor. The Script Editor is for use with Route 1 only, and this version is compatible with versions of Route 1 including and following 4.1.

Route 1 is shareware, but it's not free. If you like it, please send $18.00, with the included order form, to the address below. Your registration gets you free technical support with Route 1, and free updates, forever. If you think that Route 1 is either useful, well designed, and innovative, or just a nuisance, please drop me a line, so I know how far it has traveled. Thank you.

David Karp
P.O. Box 20024
Oakland, CA 94620

Internet Address:
Daaron@OCF.Berkeley.EDU

Delete

Purpose: deletes a file

Syntax: Delete"*filename*"

where filename is the name of the file to delete

Remarks: the best use of this is with a variable, since a file can only be deleted once. Use the **%f** variable to obtain a listing of files in the current directory, or the **%0** for possible filenames specified on the command line. For instance, you can create a one-line script that will act as a "trashcan" with the drag-drop option set to "Launch app and open file". (the included utility 1TRASH.EXE does a much more thorough job) Simply create a script (call it trashcan.RS1, for example), and enter the following line:

```
Delete"%0"
```

This will delete any file that is included as the command line parameter, as dropped files are.

Name

Purpose: renames a file

Syntax: Name"*filename1*"to"*filename2*"

where filename1 is the old filename
where filename2 is the new filename

Remarks: use this to rename one file, or a group of files (using wildcards). Files cannot be renamed from one directory to another, although the directory must be specified if it is not in the current directory. (to move a file, first Copy it, and then Delete it) Use the **%f** variable to obtain a listing of files in the current directory, or the **%0** for possible filenames specified on the command line.

Copy

Purpose: copies a file

Syntax: Copy"*filename1*"to"*filename2*"

where filename1 is the existing path & filename
where filename2 is the destination path & filename (or just path)

Remarks: to move a file, first Copy it, and then Delete it. The file can be renamed in the process be specifying a new filename as *filename2*, or by using the Name command. Use the **%f** variable to obtain a listing of files in the current directory, or the **%0** for possible filenames specified on the command line.

List

Purpose: creates a listbox from which the user can choose an item

Syntax: List(%n)"item1|item2|item3|..."

where n is a number from 1 to 9, specifying the destination variable
where item# are the items to be displayed in the list. Items are separated by the pipe "|" symbol.

Remarks: the number of the selected item is placed as a numerical value in the variable *%n*, where 0 is for the first item, 1 for the second, 2 for the third, and so on. (-1) is returned if no item was selected, or the user selected **Cancel**. As many items as desired can be displayed, each separated by the "|" character, as shown.