HexFlags DA Version 1.1

By Mark Chally

(MPW adaptation by Ken Mcleod)

Problem: Calculate the content of a menu template's "enableFlgs" field and express in hexadecimal, quickly and conveniently (correctly, at *first* attempt), while still in ResEdit or any editor.

Solution: HexFlags DA.

• Introduction

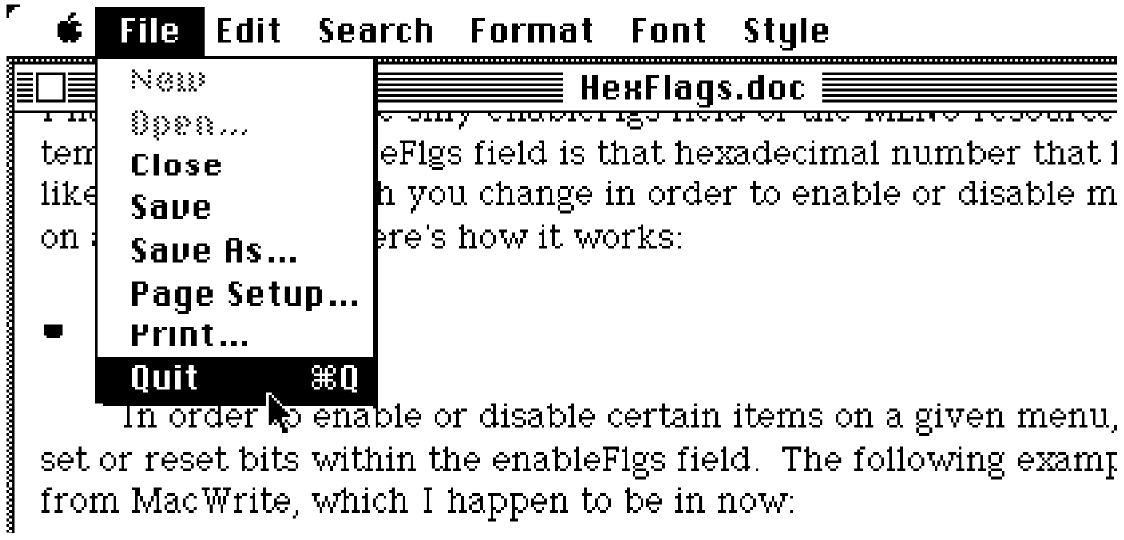
ResEdit is a great programmer's utility. It's powerful and surprisingly graceful, given its complexity. For everything it does clumsily or inadequately, it does at least ten things well. While some programmers prefer to use RMaker or Rez to create resources, I prefer to "play" with mine until I get them just the way I want them. I use ResEdit, LightspeedC, and "other stuff" running concurrently under MultiFinder for most of my development work. With such an environment, any change I want to make to my program (code, resources, bitmaps—*everything*) is just seconds away.

Given that environment, little annoyances *really* turn me off. And that's why some things about ResEdit bother me. The one thing I hate most is the silly enableFlgs field of the MENU resource template. The enableFlgs field is that hexadecimal number that looks sorta like "$FFFFFF79" which you change in order to enable or disable menu items on a given menu. Instead of putting a checkbox on each menu item in the resource, the author (Andy?) chose to use "the flags from hell" as an input field.

Correspondingly, Rez requires you to supply the "flags from hell" in a slightly different format (one bit off, in fact) for any given menu. Here's how it's done:

• Theory

In order to enable or disable certain items on a given menu, you must set or reset bits within the enableFlgs field. The following example is taken from MacWrite, which I happen to be in now:



tem eFlgs field is that hexadecimal number that l
like h you change in order to enable or disable m
on ere's how it works:

In order to enable or disable certain items on a given menu,
set or reset bits within the enableFlgs field. The following examp
from MacWrite, which I happen to be in now:

The above menu would be represented in binary and hexadecimal as the following:

ResEdit Notation

Hex:	$FFFFF**[**FF9**]**
Binary:	1111 1111 1111 1111 1111 111**[**1 1111 1001**]**

Rez Notation

Hex:	$7FFFFF**[**FC**]**
Binary:	0111 1111 1111 1111 1111 1111 **[**1111 1100**]**

I have placed brackets around the “significant” digits to indicate that they are the digits that “matter” for the above menu. That is, when the system looks at the enableFlgs field, it only looks at the bits (hence, with the hex, hexadecimal digits) that correspond to existing menu items. The rest of the field is, in effect, ignored.

By matching the binary representation of the flags with the menu, you should notice that each bit from *right to left* corresponds to a *descending* (i.e.. starting at the top, for ResEdit beginning with the *menu title* , and for Rez, beginning with the first) menu item. So for ResEdit, the rightmost binary digit corresponds to the menu title (disabling it disables all menu items), the second from the right corresponds to the first menu item, and each successive digit to the left corresponds to the next item. For Rez, the rightmost binary digit corresponds to the first item, and the second from the right, the second, etc.

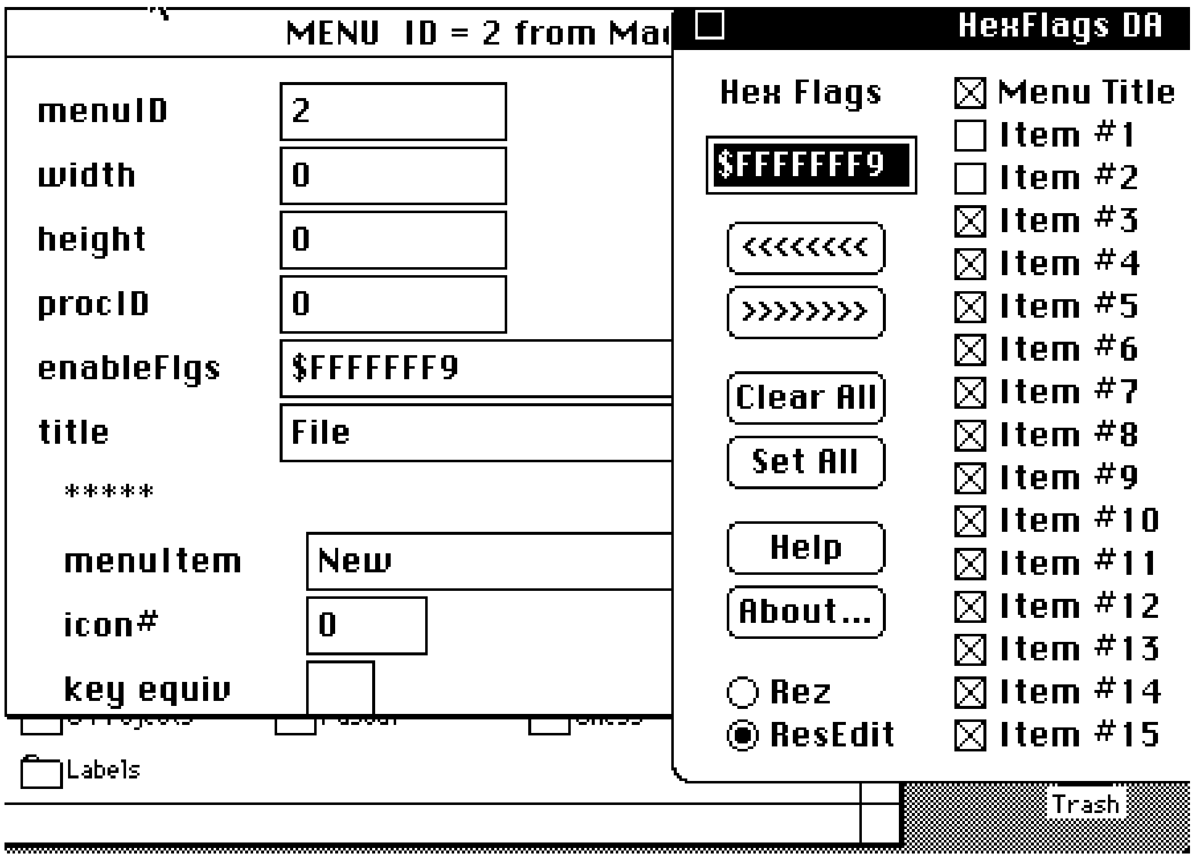
For our example then, the rightmost nine bits (there are eight menu items, plus the title makes nine necessary bits) become significant for ResEdit, and the rightmost eight for Rez. In converting that to hexadecimal, the rightmost three (the 9th bit from the right starts the third hex digit) hex digits are significant for ResEdit, and two for Rez.

So, you can see that in order to create a valid hexadecimal number for the enableFlgs field, you must first “plot” your binary number, then convert each set of four bits to a hexadecimal digit. Having done that (using hard-boiled base-swapping math, or a note sheet), you can then insert the resultant hexadecimal number into ResEdit’s enableFlgs field or use the number for Rez source (padded with ‘F’s, usually, but padding with anything will work fine—if you pad with ‘F’s, any menu items you add will “automatically” be enabled by default. Also remember that the largest number for Rez is $7FFFFFFF, because you still only manipulate 31 menu items). It isn’t a lot of work, but it’s enough to break your concentration if you want to make a quick change to a menu and get back to your programming. Plus, if you do it with the haste that I do, you’ll often get it *wrong* on the first attempt. Along comes **HexFlags**.

• Usage

HexFlags is simple to use, but not entirely obvious. First, insert it into your system file or an editor using the Font/DA mover that corresponds to your current system/finder. *As a short aside, it will not work to use the option-open command of Font/DA mover to install HexFlags into ResEdit. I originally intended to paste it into ResEdit so it would only show up within that program, but I found out that ResEdit apparently* ***does not look*** *at its resource fork when it loads a desk accessory (I guess it assumes the DA is in the system file, since it's not distributed with any DAs installed in it). HexFlags* ***does*** *work when installed into* ***other*** *programs though—if for some reason you should want to.*

Anyway, once HexFlags has been installed into your system file, you may load it from ResEdit, or any other program. It works nicely under MultiFinder, and doesn't care if you load it (as normal) into the system heap, or hold down the option key when loading it in order to load it into the application heap. If you *do* run it from ResEdit and you do have a resource file open (the one in the picture is MacWrite), it will look something like this:



MENU ID = 2 from Ma

menuID	2
width	0
height	0
procID	0
enableFlgs	$FFFFFFF9
title	File
*****	
menuItem	New
icon#	0
key equiv	

HexFlags DA

Hex Flags: $FFFFFFF9

( ) Rez
[✓ ResEdit]

- ☒ Menu Title
- ☐ Item #1
- ☐ Item #2
- ☒ Item #3
- ☒ Item #4
- ☒ Item #5
- ☒ Item #6
- ☒ Item #7
- ☒ Item #8
- ☒ Item #9
- ☒ Item #10
- ☒ Item #11
- ☒ Item #12
- ☒ Item #13
- ☒ Item #14
- ☒ Item #15

To see which items would be enabled for the hexadecimal number in the enableFlgs field, you may simply copy the number from ResEdit's enableFlgs field, and paste it into the MenuFlags field of the DA. Then press the ">>>>>>>>" button to cause the appropriate boxes to be checked. (Be sure you're in ResEdit mode for ResEdit, or Rez mode for Rez.) Each checked box corresponds to an enabled menu item, and each unchecked corresponds to a disabled one.

Going the other way is even simpler (and probably used more often). Simply set your mode (Rez or ResEdit), check the boxes that correspond to menu items you'd like enabled (use the "Set All" or "Clear All" button to start you off if you like) and uncheck the ones you'd like disabled, then just press the "<<<<<<<<" button. The hexadecimal number then appears in the MenuFlags field. Copy it and paste it into ResEdit's enableFlgs field or into your Rez source if you wish. That's *all* there is to it!

Cut, Copy, Paste and Clear are handled as expected, and undo is ignored. "Set All" sets all check boxes, "Clear All" clears them all—each updates the MenuFlags field of the DA. "Help" offers operational hints "on the fly", and "About…" gives me a minute of glory.

The "Rez" and "ResEdit" radio buttons set your mode, deciding which method will be used to translate between boxes and hex (whether the menu title is included as an "item"). Changing the radio button selection will cause the hex value to be converted from one system to the other. When the "Rez" option is selected, the "Menu Title" check-box goes away.

The other two buttons work as described above.

The window may be moved about freely to wherever you'd like, but I guessed you'd probably want it where it normally is placed, as you need room to work. (You're certainly welcomed to change ths default dialog rectangle in your own working copy if you're adept with ResEdit.) When you're through with the DA, just click its close box.

If you would like the radio button for the conversion system to default to "Rez" instead of "ResEdit", you may change the string (named "HexFlags" as are all of HexFlags' resources) in the "STR " resource to "Rez" instead of "ResEdit". (Note, HexFlags will look for "ResEdit" in that string at the time the DA is opened. If it doesn't find it, it will assume "Rez".)

• Modification History

Version 1.01 — Moved "$" character from statText field in front of hex string in MenuFlags field to part of the hex string. Version 1.01 and later adds "$" if it is missing and filters out any other non-hex characters. Now the string resembling "$FFFFFFFF" can be copied or cut directly from ResEdit's enableFlgs field and pasted directly into the MenuFlags field of HexFlags—Version 1.0 didn't respond "nicely" to a string such as "$FFFFFFFF" in the MenuFlags field, but required a string of the form "FFFFFFFF" instead. Version 1.01 and later welcomes either format and also adds "$" to the beginning (if absent) when it calculates the hex value.

Version 1.02 — Slightly changed the treatment of the edText MenuFlags field when SetIText is called. Replaced the code with a more elegant patch around SetIText's "Modeless Dialog in a D/A" bug that caused the edText field not to be updated. This was a cosmetic change.

Version 1.1 — Added support for MPW "Rez" utility, including "Mode" buttons to change between ResEdit and Rez modes, allowing the format preferred by each program to be used. Added "automatic" resetting of the enableFlgs DA field when "Clear All" and "Set All" buttons are used. Final compile under MPW by Ken Mcleod (kudos, Ken) instead of under Turbo Pascal, so the DA is only about 7k instead of about 15k.

• About Support, Appreciation, and the Hacker Ethic.

I believe in the *Hacker Ethic*. As I have come to understand it, the hacker ethic indicates that programmers should freely provide each other utilities, programming methods, concepts, and most importantly: support and understanding. If we work together and help each other solve problems that apply to us all, we can significantly enhance our world.

Where does that come into this? Well, I've done my part. After using a variety of "neat" shareware and public domain utilities (I have actually **paid** for shareware that I use), I've felt compelled to offer a utility such as this freely to all for the common good.

What do you owe me? HexFlags is "**WhoCaresWare**". I'm not gonna send a guy named "Guido" with a violin case after you so that you send me money or stop using this DA, nor am I going to suggest an amount of money or even make you feel guilty for not sending any. If you don't want to send me money, don't get all hot and sweaty worrying about it. If you do, then go ahead and send me what you think it's worth (no coins, fruit, or vegetables please).

Okay, so what am I getting at? Just this: if my work has benefitted you, do something sporting about it: send me comments, bug reports, literary pats on the back, etc. Better yet, *write something useful yourself!*

• Now that you're filled in

I really would like to hear from you (even hate-mail), especially the bug reports. You can send me your comments, suggestions, (money,) etc. at any of the following:

Chally Micro Solutions
P.O. Box 4600
West Covina, CA 91791

The AppleBus BBS
Sysop, Mark Chally (me)
818-919-5459

GEnie, Chally.Micro
MCI Mail, ChallyMicro
MacNet, ChallyMicro

The Macintosh world is something really special, and I'm happy to be a part of it. Thanks to all of you who have "done your part".

Mark Chally