Pokémon Gen III ROM Hacking Suite

aka “Lord of the Tools” — MrDollSteak

At the time when tools like YAPE and A-map were created, nobody was thinking about advances in hacking technique that would enable the addition of new types, new moves, and even, an increase in the number of Pokémon. However, those times are gone. Now, we have several tutorials on adding new types and moves, a solid research foundation for new Pokémon, as well as a lot more advances that have rendered tools useless, like expanded OWs. In addition to this, the support for operating systems other than Windows is little to none. Being a Linux user, this got me thinking that we needed a new set of tools for the future.

Shinyquagsire has already begun taking care of scripting and mapping with Script Editor Advance (SEA) and Map Editor of Happiness (MEH), both written in Java. My forte is Python, which is also cross-platform. So, in December 2013, I began creating a cross platform hacking suite. The first tab is a Pokémon data editor, designed for the future. It can support new types, new moves, new Pokémon and so much more. Eventually, it will have many more tabs and pop-out windows to edit all sorts of things. So, let's begin.

The ini

This ini for this suite is a new breed. This ini actually customizes itself to a specific ROM. At an address specified in it (default 0xFFFFFE) a set of 2 bytes are read from the ROM when it is loaded. The bytes correspond to a number. If the bytes read are FFFF, a new, unused number is assigned to the ROM and written to that address. Then, the base ROM is checked (for example, FireRed (E) is BPRE) and the data from the base ROM's section of the ini is copied into a new section with the ROM's new assigned number. I know that sounds confusing, so let me show an example:

Example: PokeRoms.ini

```
[ALL]
offsetthatcontainssecondromid = 0xFFFFFE
# This is the offset of the ROM id.
justusestandardini = False
# If this is "True", the program will just the read base ROM's
# section and use that!

[BPRE]
name = Pokemon FireRed (E)
gamecode = BPRE
numberofpokes = 411
pokebasestats = 0x2547A0
pokebasestatslength = 0x1C
#-----snipped for length------#
```

Okay, so the above is an example ini. Now, let's say you load a FireRed ROM. When you load it, the ini becomes this:

Example 2: PokeRoms.ini

```
[ALL]
offsetthatcontainssecondromid = 0xFFFFFE
justusestandardini = False

[BPRE]
name = Pokemon FireRed (E)
gamecode = BPRE
numberofpokes = 411
pokebasestats = 0x2547A0
pokebasestatslength = 0x1C
#-----snipped for length------#

[0000]
name = Pokemon FireRed (E)
gamecode = BPRE
numberofpokes = 411
pokebasestats = 0x2547A0
pokebasestatslength = 0x1C
#-----snipped for length------#
```

So now, your ROM has its own section in the ini. You know what that means, right? You can

expand and re-point things *without* affecting other roms that are loaded. Cool, huh? Also, you can use decimal, hexadecimal, or even binary in the ini.😄 Just use the prefixes "0x" for hexadecimal and "0b" for binary.

Please note that booleans (True or False) are case sensitive. They *must* have capitalized first letter and lowercase the rest. (This is a Python standard.) Also, lists like egg-groups must not have any spaces between the list objects, but they may have spaces in the names. Examples:

```
Wrong: 0x7888, 0x87988, 0x885938
Right: 0x7888,0x87988,0x885938

Wrong: Egg Group 1, Egg Group 2
Right: Egg Group 1,Egg Group 2
```

Now, I know that the names I use in my ini can be confusing. So, made a list of every single name I used in the ini and what it does. It is at the end of this document, just after the character map.

If you are using MrDollSteak's ROM base, then I have included built-in support for this too! Using a hex editor, go to 0xAC in your ROM. On the side, you will see text (BPRE would hope, lol). Type over this with MrDS, then load your ROM up in the suite. It will read from MrDollSteak's section of the ini and load everything just fine. After the first loading, it is okay to change it back to BPRE.

Now that you know what everything in the ini does, there is only one more thing I want to talk about. If you know that your game has repointed data, prepare your ini before hand. If the program tries to load data at an offset that has been repointed and no longer exists, it will most likely error (as with almost any hacking tool :p. So, before hand, set up the ini for your ROM. You can do this, by first copying the ini section of your base ROM to the very end of the ini. Then, assign the ROM an unused number to replace its section header (for a FR ROM, [BPRE] would become something like [0003]). Then, navigate to the "offsetthatcontainssecondromid" offset in a hex editor and type that number in exactly as is. Then, you can fill out that new ini section with your custom offsets.

Usability Notes

This program is designed with a level of flexibility that is above many of the old ROM hacking tools. This means that it isn't going to prevent you from entering an ATK value of 9000. Instead, it will simply cut it down to the max if it is over the maximum possibility on saving. This goes for most of the program, if you enter a value that is too large, or too small the values will not be adjusted unless they absolutely have to be.

Another usability feature that I need to discuss are the combo boxes (the drop down menus like "Type"). You can type in them to make your section, however you need to make sure you hit enter/return to lock in your choice. Otherwise, it will error on saving when it tries to load a custom option. All of the combo boxes support auto-complete so when the suggestion you want comes up, just hit enter to select it!

In this program, no data is saved until you click a save button or repoint something. On the Pokémon Data Editing tab, there are two save buttons: "Save Tab" and "Save All". "Save Tab" will just save the current tab you are in. So if you just want to save the stats tab, switch to it and click

"Save Tab". The "Save All" button will save everything in the Pokémon Data Editing tab. There is also an autosave feature that can be turned on in the ALL section of the ini. This will autosave Pokémon data when switching Pokémon.

The repoint functions all work differently to cope for the different data, so I will discuss them all individually. The only thing they completely have in common is that all offsets are in Hex:

Move Repointer

This repointer allows you to change the size of all of the movesets. When you repoint, it will change all of the pointers to the move table to match the new offset. Then, it writes a temp table of all 00s to reserve the new table. However, it writes nothing else. Only when you click the "Save all" button, the moveset actually written to the ROM and the old table cleared out with FFs (this can be changed by unchecking the box in the move tab for the purpose of preserving shared movesets).

Egg Move Repointer

This one is only called when you click save. What happens, is when the egg move table is saved to the ROM, it checks to see if the new table is longer. If it is, it will ask you to repoint. This repointer will take care of everything. All you need to do is select the offset.

Evolution Expander

This window does a lot. First, it loads the entire evolution table as is in the ROM (the unsaved version) and expands each entry to the new number you have chosen. Them, it jumps all over the ROM and changes each of the loading routines to all for the expanded table. Then, it writes in the new expanded table and fills the old one with FFs. Then it reloads the evolutions from the new table. This means, that if you change evolutions, but then expand, you will lose your changes, so please save before doing this.

Pokédex Entry Repointer

Like the Egg Move Repointer, this one is only called by the save button, if needed. It compares the new 'dex entry to the old one and then if it is longer, it will ask you to repoint. It checks and does both entries separately for Ruby and Sapphire. Unless the box for overwriting entries is checked, entries will not be overwritten with FF. (This is the default) Uncheck if the entry is shared.

Sprite Repointer

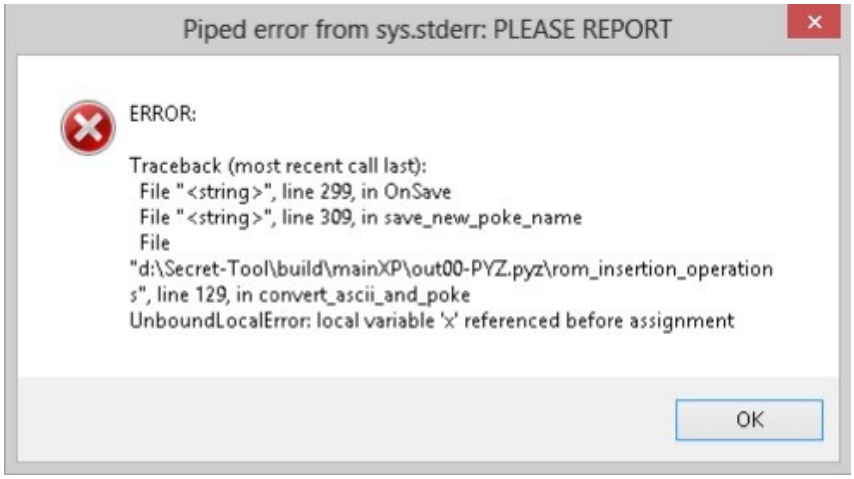
The sprite repointer is called on saving when the images are larger or when you click to repoint the icon in the case of shared icons. When icons are repointed, the are not filled with free space since every icon is the same size and repointing means that it is shared and you are unlocking it from being shared. For the sprites, if the box to fill old sprites with free space is checked, the old sprites will be filled. There is also a check box to repoint each sprite individually. If this is checked, you will be asked to repoint each sprite and palette individually. If it is unchecked, you will only be asked for a single offset and all of the sprites will be placed one after another at that offset.

One last note one usability. Buttons only pertain to the tab that contains them. This means that the "Save all" button on the Pokémon Data Editor tab only saves the data in that tab and its children tabs. It will not save the Move Editor tab or any other tab I plan add.

Please feel free to use the contact feature in the help menu if you want to drop some suggestions or comments.😄

Errors

No matter how good of a programmer you are, there will always be some type of bugs in your program. Whether they are small (like hitting tab and it switching to the wrong box) or large (like crashing the program when you hit ctrl-c), it would be helpful for you to report them all. Any errors that look like this:



are usually pretty serious. What I did here, was I forgot to assign variable x a number before entering a loop where it was used as a counter. However, on the larger scale, I ended up redirecting all errors from the console in Python to be sent to an error window like this one. Python errors consist of a traceback that has information like the line number that the error occurs on, what file it occurs in, what function called the function that errored, and other stuff like that. This makes them extremely useful for fixing bugs. So, if you ever get an error like this (and you didn't do something crazy like fill all of the boxes with 999999999), submit it to me and I will fix it.:D

Now, as of version 1.2, there is a built-in ini recovery feature that recover broken inis for English roms.

Let's say you expand Pokémon on FR but then delete your ini. Whoops! In the past, you had to work really hard to restore the ini or just start over. Now, a little box will pop up after every error and ask if you think it was related to the ini and whether or not you want to try and recover it. If you think so, click yes and give it a try. G3HS has an internal ini that is a list of pointers where it can find almost every offset in the ini by going to that location in the ROM. It will set you up a new ini section and hopefully make your ROM loadable again!

Explanation of Tabs and their Features

Pokémon Data Editor

Stats Tab

This tab is to edit all of the stats of each Pokémon. It supports expanded types/moves/etc.

Moves Tab

This tab edits all of the moves and TM/HMs a Pokémon can learn. It also supports the repointing of movesets to allow expanding them. You can double click on a move to copy its properties into the edit boxes. If you are using Jambo51's move table hack, make sure you change the ini accordingly. (Move length to 3 and change his hack settings to True.)

Evo Tab

This tab is to edit the evolutions of each Pokémon. By double clicking on an evolution you can copy it into the edit boxes. It supports the ability expand the number of evolutions per Pokémon. It does this using the offsets in the ini to write the changes that allow the game to read the expanded evolution sets.

For v1.2, I added the ability to use Pokémon, moves, and locations as arguments for evolution. See the ini guide for move information.

For v1.3, I added the ability to use specifics maps and types as arguments also.

Pokédex Tab

This tab can edit all of the things Pokédex except for footprints. (I will get to those.) A small note that you can not position things negatively like you can in YAPE. DoesntKnowHowToPlay informed me that doing so breaks several moves. For FireRed, there is a button that applies the fix for the Pokédex not displaying names that have more than one word properly. This tab also supports editing footprints. For these, just load a 16x16 image and I will convert it to 2 colors and import it. I am working on a way to draw them but it isn't a priority.

Move Tutor Tab

The sole purpose of this tab is to, you guessed it, edit move tutors. On the left, you can edit each Pokémon's individual ability to learn move tutor moves. On the right, you can edit the actual moves themselves, so the moves that are offered to be taught.

Egg Moves Tab

This tab edits the table of egg moves in its entirety. You can add Pokémon to the table, remove Pokémon, and change their moves that they can learn.

Sprites Tab

I had a lot to live up to when making this tab. I did my best to give it all if the functionalities of Wichu's Sprite Editors and then some. If you click on the sprites or colors you can change them.

Remember that icon palettes are shared! When loading the full 256x64 sheet, it uses the Wichu format of Front, Front Shiny, Back, Back Shiny. (NOT the order that my editor displays them in!) There is also a graphic position pditor that you can control by changing the Player Y, Enemy Y, and Enemy Alt dials.

Unique Features of this sprite editor:

- Formes like Castform and Deoxys are loaded based upon the size of the image, meaning that nothing is hard coded so you can add more if you are skilled enough.
- Emerald animations are read the same way as formes but directly from the main sprite table. (The secondary leftover table from RS is ignored.) Because of this, if you manage to port Emerald animations to FireRed or Ruby, they will still work perfectly.
- The ability to repoint each sprite individually or together as one.
- Fill old images with Free Space!
- Edit icon palettes.
- Icon animation preview.

For v1.2, I did my best to code an auto-select palette feature for the icons and it seems to work fine. I would always suggest using GIMP or something similar to align the colors of the image properly to one of the in-game palettes before importing it if you still have issues. Remember: programs don't have brains, it is only math. Only a real human can really determine if the colors are right.

Habitat Tab

(New in version 1.2) Yes, the first ever habitat editor. This will allow you to edit those posky habitat tables that are a mess of pointers and offsets. Just select a habitat and you can move Pokémon around, remove and add them, then save them back to the ROM. Don't worry about forgetting any, all unused pokés are kept track of in the 3rd column.:)

Note: Because the tables are such a mess and I am not that good of a programmer, when you click save, the table is then built all as one and you are asked to repoint. It will take the new block table and save it to the new offset you pick and the old one will be filled with free space. So yeah, don't edit the table until you have everything else set unles you want to be constantly moving it.:P

Expand Pokémon

Yes, you read that right. While this button only exists for FireRed BPRE, it still does exist. (It only exists for BPRE because it has only ever been done on that. The code follows DoesntKnowHowToPlay's tutorial to the letter.) When you use this, it will apply JPAN's Save Block Hack to your ROM. If you already have his hack installed, don't worry, it will simply do it again, just make sure you change the RAM offset in the window to something different if you used that area. Check DoesntKnowHowToPlay's tutorial for more info on this. You will be asked for the number of New Pokemon you want to add and the Dex size you want after it is complete. (The dex size starts at 387 because you have to count Missingno) After providing that, just search for a free space offset or provide one. The rest will be done for you in under a second.:D This program also accounts for the Unown and Egg that have sprites are such at the end of the table, so don't worry about those factoring into your numbers.

Please make note that this is still kind of experimental. It has been shown to work, but it has not been 100% tested. (This actually goes for most of this tool, haha)

Hex Editor

This is just a simple hex editor. Over time, I will try and add more features. I did NOT code most of the editor. I did make personalizations, but here is the info for the main code:

```
Copyright 2013 Gully Chen

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS"
BASIS,WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied.

See the License for the specific language governing permissions and
limitations under the License.

Author:  CChen
Purpose: Hex editor based on wxPython
Created: 04/16/2012
```

Updating

Before version 1.2, you had to go to GitHub and download the new update a deal with it yourself. Well, from now on, you can choose to have me update for you. If you choose “Auto-update” from the updater dialog in G3HS, it will download and unpack the correct zipfile for you and then merge you ini with the new one. All of you custom ROM sections will be preserved, and any new entries (like things I need for new features) will be added to each section. Always back-up your ini first though!!! In case anything goes wrong. If you have any issues, please let me know.

Ini for moving to v1.2?

If at any point you need to merge inis on the fly like the updater does (in the case of moving from anything less than or equal to v1.1.6 to v1.2) use the ini merger found in the Tools menu. This will guide you through the process of merging the new ini in.

Linux and Mac

Currently, I am running Windows XP and Ubuntu 13.10 on my computer and openSUSE on a separate machine. This means that I have the ability to create binaries only for Windows and those two types of Linux. So the binaries I make may not work on Arch, Fedora, old Ubuntu, Mac, and most other operating systems. This is where you come in: if you run one of the unsupported OSs, let

me know. I can teach you how to package and make binaries that you can help me distribute. The reddit mod browniebiznatch has agreed to help me make a package for Mac. Once I set you up and teach you what to do, it will be as simple as pulling down the latest version, running a single command, and then uploading. Now bad.:)

If you just want to run it without packaging, you should familiarize yourself with how Python works and then install these python libraries:

- Python 2.7.6
- Python Imaging Library (PIL)
- wxPython v2.8 (v3.0 won't load this for some reason)

Then, you can run my code from command line via "python main.py" when inside of the folder that contains the source code.

Character Map

Because I am using Latin-1 and UTF-8 encoding (to be as cross-platform as possible) not every character that you might need is completely supported. So, I had to create my own character map to make up for the lost ones. Here is a table of characters, what you have to type to get them, and their hex values in the ROM. Characters that don't exist under UTF-8 (Œ) or multi characters (PO, Ké) use the XSE style of brackets to be supported. So those characters would use [OE], [PO], & [Ké], respectively.

Please note, this is not perfect. I still have a lot of work to do on this chart. If you find any characters that aren't working, just send me a message.

Please note that I made this chart by hand. That is why the "(?)" characters in the middle are not filled out. I have no idea what they are (If you do know, tell me and I can fill this chart out more. I think they may be the Japanese chartset... Maybe...) but you can still access them by type the "Hex Value in the ROM" without the "0x" inside of brackets. So 0x49, which is a "(?)", would become [49].

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
(Space)	(Space)		0x00
À	À		0x01
Á	Á		0x02
Â	Â		0x03
Ç	Ç		0x04
È	È		0x05
É	É		0x06
Ê	Ê		0x07
Ë	Ë		0x08

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
Ì	Ì		0x09
(?)((	Í		0x0A
î	Î		0x0B
Ï	Ï		0x0C
Ò	Ò		0x0D
Ó	Ó		0x0E
Ô	Ô		0x0F
Œ	[OE]		0x10
Ù	Ù		0x11
Ú	Ú		0x12
Û	Û		0x13
Ñ	Ñ		0x14
ß	ß		0x15
à	à		0x16
á	á		0x17
(?)	[18]		0x18
ç	ç		0x19
è	è		0x1A
é	é		0x1B
ê	ê		0x1C
ë	ë		0x1D
ì	ì		0x1E
(?)	[1F]		0x1F
î	î		0x20
ï	ï		0x21
ò	ò		0x22
ó	ó		0x23
ô	ô		0x24
œ	[oe]		0x25
ù	ù		0x26
ú	ú		0x27
û	û		0x28
ñ	ñ		0x29

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
º	º		0x2A
ª	ª		0x2B
¹	¹		0x2C
&	&		0x2D
+	+		0x2E
(?)	[2F]		0x2F
(?)	[30]		0x30
(?)	[31]		0x31
(?)	[32]		0x32
(?)	[33]		0x33
Lv	[Lv]		0x34
=	=		0x35
;	;		0x36
(?)			0x37
(?)			0x38
(?)			0x39
(?)			0x3A
(?)			0x3B
(?)			0x3C
(?)			0x3D
(?)			0x3E
(?)			0x3F
(?)			0x40
(?)			0x41
(?)			0x42
(?)			0x43
(?)			0x44
(?)			0x45
(?)			0x46
(?)			0x47
(?)			0x48
(?)			0x49

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
(?)			0x4A
(?)			0x4B
(?)			0x4C
(?)			0x4D
(?)			0x4E
(?)			0x4F
(?)			0x50
¿	¿		0x51
¡	¡		0x52
PK	[PK]		0x53
MN	[MN]		0x54
PO	[PO]		0x55
Ké	[Ké]		0x56
BL	[BL]		0x57
OC	[OC]		0x58
K	[K]		0x59
Í	Í		0x5A
%	%		0x5B
(	(		0x5C
)	)		0x5D
(?)			0x5E
(?)			0x5F
(?)			0x60
(?)			0x61
(?)			0x62
(?)			0x63
(?)			0x64
(?)			0x65
(?)			0x66
(?)			0x67
â	â		0x68
(?)			0x69
(?)			0x6A

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
(?)			0x6B
(?)			0x6C
(?)			0x6D
(?)			0x6E
í	í		0x6F
(?)			0x70
(?)			0x71
(?)			0x72
(?)			0x73
(?)			0x74
(?)			0x75
(?)			0x76
(?)			0x77
(?)			0x78
(?)			0x79
(?)			0x7A
(?)			0x7B
(?)			0x7C
(?)			0x7D
(?)			0x7E
(?)			0x7F
(?)			0x80
(?)			0x81
(?)			0x82
(?)			0x83
(?)			0x84
(?)			0x85
(?)			0x86
(?)			0x87
(?)			0x88
(?)			0x89
(?)			0x8A
(?)			0x8B

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
(?)			0x8C
(?)			0x8D
(?)			0x8E
(?)			0x8F
(?)			0x90
(?)			0x91
(?)			0x92
(?)			0x93
(?)			0x94
(?)			0x95
(?)			0x96
(?)			0x97
(?)			0x98
(?)			0x99
(?)			0x9A
(?)			0x9B
(?)			0x9C
(?)			0x9D
(?)			0x9E
(?)			0x9F
(?)			0xA0
0	0		0xA1
1	1		0xA2
2	2		0xA3
3	3		0xA4
4	4		0xA5
5	5		0xA6
6	6		0xA7
7	7		0xA8
8	8		0xA9
9	9		0xAA
!	!		0xAB
?	?		0xAC

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
.	.		0xAD
-	-		0xAE
|	|		0xAF
..	[..]		0xB0
“	["<]		0xB1
”	[>"]		0xB2
'	'		0xB3
'	'		0xB4
♂	[MALE]	[M]	0xB5
♀	[FEMALE]	[F]	0xB6
$	$		0xB7
,	,		0xB8
×	×	[B9]	0xB9
/	/		0xBA
A	A		0xBB
B	B		0xBC
C	C		0xBD
D	D		0xBE
E	E		0xBF
F	F		0xC0
G	G		0xC1
H	H		0xC2
I	I		0xC3
J	J		0xC4
K	K		0xC5
L	L		0xC6
M	M		0xC7
N	N		0xC8
O	O		0xC9
P	P		0xCA
Q	Q		0xCB
R	R		0xCC
S	S		0xCD

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
T	T		0xCE
U	U		0xCF
V	V		0xD0
W	W		0xD1
X	X		0xD2
Y	Y		0xD3
Z	Z		0xD4
a	a		0xD5
b	b		0xD6
c	c		0xD7
d	d		0xD8
e	e		0xD9
f	f		0xDA
g	g		0xDB
h	h		0xDC
i	i		0xDD
j	j		0xDE
k	k		0xDF
l	l		0xE0
m	m		0xE1
n	n		0xE2
o	o		0xE3
p	p		0xE4
q	q		0xE5
r	r		0xE6
s	s		0xE7
t	t		0xE8
u	u		0xE9
v	v		0xEA
w	w		0xEB
x	x		0xEC
y	y		0xED
z	z		0xEE

Character in the Game	What to Type	What to Type (Alternate)	Hex Value in ROM
(?)	[EF]		0xEF
:	:		0xF0
Ä	Ä		0xF1
Ö	Ö		0xF2
Ü	Ü		0xF3
ä	ä		0xF4
ö	ö		0xF5
ü	ü		0xF6
(?)	[F7]		0xF7
(?)	[F8]		0xF8
(?)	[F9]		0xF9
(?)	[FA]		0xFA
+	+		0xFB
(?)	[FC]		0xFC
(?)	[FD]		0xFD
\n (newline)	(Enter/Return)		0xFE
(Free Space/End)	[FF]		0xFF

Names used in the ini

```
[ALL]
offsetthatcontainssecondromid = 0xFFFFFE
# This is the offset that ini uses to find your ROM.

justusestandardini = False
# This disables the dynamic ini feature when set to "True".

checkfordevbuilds = False
# Would you like to get updates for prereleases?
# Or just the major releases?

checkforupdates = True
# I guess if you didn't want to be notified about updates at all,
# you could set this to False. But why???:p

autosavepokeswhenswitching = False
# If you want to autosave when switching Pokés, set this to True.


[GAME]
name = Pokemon FireRed (E)
# This is the name of your ROM.
```

```
# It will appear at the top of the window when the ROM is open.

gamecode = BPRE
# DO NOT change this.
# This is how the program tells what the base ROM for your hack was.

numberofpokes = 411
# Number of Pokémon (including ???) in your hack.

pokebasestats = 0x2547A0
# Offset of the base stats table for the Pokémon.

pokebasestatslength = 0x1C
# The length of each entry in the table above. (Most likely,
# you won't need to change this, or any other table length.)

pokenames = 0x245EEB
# The offset of the table of Pokémon names.

pokenameslength = 0xB
# The length of each Pokémon name.

typenames = 0x24F1A0
# The offset of the table of type names.

typenameslength = 0x7
# The length of each type name.

numberoftypes = 18
# The number if types.

items = 0x3DB028
# The offset of the table of items.

numberofitems = 0x177
# The number of items.

itemsdatalength = 0x2C
# Then length of each entry in the items table.

abilities = 0x24FC40
# The offset of the table of ability names.

numberofabilities = 0x4E
# Number of abilities.

abiltiesnamelength = 0xD
# The length of each ability name.

egggroups = Monster,Water1,Bug,Flying,Field,Fairy,Grass,Human-
Like,Water3,Mineral,Amorphous,Water2,Ditto,Dragon,Undiscovered
# This is a list of all of the egg groups in your hack,
# here so you change them to your liking or add new ones
# (if you manage that feat!:p).

leveluptypes = Erratic,Fast,Medium-Fast,Medium-Slow,Slow,Fluctuating
```

```
# A list of the different styles of experience gain.
# You change these to your liking.

learnedmoves = 0x25d7b8
# The offset of the learned moves table which contains
# the pointers to each 'mon's individual learned moves table.

learnedmoveslength = 2
# The length of each move entry.
# If you use Jambo's move table hack, this will be 3.

attacknames = 0x247094
# The offset of the table of all attack names.

numberofattacks = 0x163
# Number of attacks in your hack.

attacknamelength = 0xD
# The length of each attack name.

eggmovepointer1 = 0x045C50
# The first pointer to the egg move table.

eggmovepointer2 = 0x045CC8
# The second pointer to the egg move table.

tmhmcompatibility = 0x252BC8
# The offset of the TM/HM compatibility table.

tmhmcompatibilitylength = 8
# The length of each entry in the TM/HM comp. table.

tmlist = 0x45A80C
# The offset of the table that contains the move for each TM/HM.

tmlistentrylength = 2
# The length of each entry in the TM/HM table above.

numberoftms = 50
# Number of TMs.

numberofhms = 8
# Number of HMs.

evolutiontable = 0x259754
# The offset of the table of evolutions.

evolutionsperpoke = 5
# The number of evolutions each 'mon has.

lengthofoneentry = 8
# The length of one evolution in the evolution table.

evolutionmethods = None,Friendship #snipped
# The names of all of the different types of evolution.
```

```
evomethodsproperties = None,Level #snipped
# What is the argument for each evolution type above?
# Choices are:
# None – Should be obvious:P
# Level – Evolve at what level?
# Item – Evolve using (or holding) what item?
# Location/MapName – Evolve when at a map with what specific name?
# Map – Evolve when on what specific Bank & Map?
# Type – Evolve with what Type? (Fire, Dragon, etc)
# Pokemon – Evolve by what Pokémon?
# Move – Evolve by what move?

evolutiontablepointers = 0x42F6C,0x42FBC,0x43138,0x4599C,0xCE8C4
# All of the pointers to the evolution table.

offsetstochangetolslr0r60x1 =
0x42f9c,0x43182,0x43026,0x43008,0x43016,0x43050,0x4307A,0x430A8,0x43
0C8,0x430EC,0x430FC
# This, and the next 3 entries are used to edit the ASM routines
# that load the evolution table.
# There is a more in-depth explanation in the evolution section.

offsetstochangetonewminus1 = 0x43116,0x4319e,0x459A2       #Edit evo
ASM.

theshedinjafix = 0xCE766
# Edit evo ASM.

changetonewnumbertimes8 = 0x4598A,0x459C0,0x4598E,0x459C2
# Edit evo ASM.

pokedex = 0x44E850
# The offset of the table that has all of the Pokédex entries.

lengthofpokedexentry = 0x24
# The length of each 'dex entries.
# Emerald has one less pointer, so it is 0x20 instead of 0x24.

nationaldexorder = 0x251FEE
# The table that contains the National 'dex order.

dextype = FRLG
# The 'dex structure. Valid values are FRLG, RS, and E.
# (Case-sensitive)

jambo51learnedmovehack = False
# If you are using Jambo51's move table hack, set this to "True".

numofnondexpokesbetweencelebiandtreeko = 25
# This is the number of Pokémon between Celebi and Treeko
# that don't have 'dex entries.
# If you have given them 'dex entries, set this to 0.

numberofnondexpokesafterchimecho = 28
# This is the number of Pokemon without 'dex entries after Chimecho.
```

```
movetutorcomp = 0x459B7E
# The move tutor compatibility table.

movetutorcomplen = 2
# The length of each entry in the move tutor compatibility table.

movetutorattacks = 0x459B60
# The move tutor attacks table.

mtattackslen = 2
# Length of each entry in the move tutor attacks table.

mtattacksnum = 15
# Number of move tutor attacks.

frontspritetable = 0x2350AC
# The front sprite table.

backspritetable = 0x23654C
# The back sprite table.

frontpalettetable = 0x23730C
# The normal palette table.

shinypalettetable = 0x2380cc
# The shiny palette table.

enemyytable = 0x2349CC
# The enemy y position table.

playerytable = 0x235E6C
# The player y position table.

enemyaltitudetable = 0x23A004
# The enemy altitude position table;

iconspritetable = 0x3D37A0
# The icon table.

iconpalettetable = 0x3D3E80
# The icon palette table.
# A list of bytes that each tells what palette each icon uses.

iconpalettes = 0x3D3740
# The actual icon palettes.

numiconpalettes = 3
# The number of icon palettes.

itemanimationtable = 0x45FD54
# The item animation table. Used for expanding Pokémon.

itemanimationtableentlen = 5
# Length of each entry in the item animation table.

hoenncryauxtable = 0x2539D4
```

```
# The cry table for Hoenn Pokémon and up.

footprints = 0x43FAB0
# The footprints table.

habitats = 0x452c4c
# The habitat table.

habitatpointers = 0x106888,0x1068C8,0x106990,0x1069F8,0x106A98
# Pointers to the habitat table.

locationnames = 0x3F1CAC
# The table of location names.

locationstart = 0x58
# What is the index of the first name in the table

locationend = 0xC4
# What is the index of the last name in the table.

locationtblfmt = 1
# FR/LG have a different table format from RSE.
# 1 = FR/LG = [Pointer for name 1][Pointer for name 2][etc]
# 2 = RSE = [Pointer for name 1][4 bytes of numbers, counters?]
# [Pointer for name 2][4 bytes of numbers, counters?][etc]
```