Sword of Mana Hex Edit Guide

by curambar

Updated to v0.3 on Jul 10, 2009

```
                Sword of Mana Hex addresses 0.3
            ======================================
                          By Curambar



                       \-----------------/
                       |Table of Contents|
                       /-----------------\


                   \----------------------------------------/
                   | 1. Intro                               |
                   |----------------------------------------|
                   | 2. Updates                             |
                   |----------------------------------------|
                   | 3. The codes                           |
                   |                                        |
                   |    3.0. How To Use                     |
                   |                                        |
                   |    3.1. Common Codes                   |
                   |        3.1.1.  Event Items             |
                   |        3.1.2.  Recovery Items          |
                   |        3.1.3.  Mistery Items           |
                   |        3.1.4.  Trait Coins             |
                   |        3.1.5.  Spirit Icons            |
                   |        3.1.6.  Meat Items              |
                   |        3.1.7.  Seeds                   |
                   |        3.1.8.  Fruits                  |
                   |        3.1.9.  Veggies                 |
                   |        3.1.10. Weird Items             |
                   |        3.1.11. Accesories              |
                   |        3.1.12. Raw Materials           |
                   |        3.1.13. Summon Items            |
                   |        3.1.14. Lucre                   |
                   |        3.1.15. Monsters killed         |
                   |                                        |
                   |    3.2. Hero Specific Codes            |
                   |        3.2.1. Level & XP               |
                   |        3.2.2. HP & MP                  |
                   |        3.2.3. Weapon Levels            |
                   |        3.2.4. Spirit Levels            |
                   |        3.2.5. Class Levels             |
                   |        3.2.6. Stats                    |
                   |                                        |
                   |    3.3. Heroine Specific Codes         |
                   |        3.3.1. Level & XP               |
                   |        3.3.2. HP & MP                  |
                   |        3.3.3. Weapon Levels            |
                   |        3.3.4. Spirit Levels            |
                   |        3.3.5. Class Levels             |
                   |        3.3.6. Stats                    |
                   |----------------------------------------|
                   | 4. Appendixes                          |
```

==========

1. Intro

When I played this game, I had some trouble finding certain quest items, such as Bubu Worms or Moon Drops. I looked for these codes everywhere, but they seemed to be nowhere. Then I remembered what my dad said: "If you want something to be done, do it yourself". After that, I started looking for the codes myself, using the cheat utility included in VisualBoy Advance.

==========

2. Updates

VERSION 0.3:
- Weapon and Levels
- EXP
- Hex edit guide
- Codebreaker
- Major math mumbo-jumbo

vERSION 0.2:
- Added codes for the US version.
- Corrected some tpyos ^_^
- Added Magic Rope and Popoi's Notebook
- Added Weird Items (Thanks to Dan Sawchuk for the data)
- Added Level, Exp and stuff.
- Added Number of Killed Monsters
- Some esthetic changes

Version 0.1:
- Just started, all the item codes available.

==========

3. The Codes

==========

3.0. How To Use

==========

If you want to use this codes, you'll need an emulator that allows hex cheat codes, such as VisualBoy Advance. Usually, the codes have the format

XXXXXXXX:YY or XXXXXXXX:YYYY

where XXXXXXXX is the memory address, YY is the value of that address for 1 byte codes (from 0 to 255), and YYYY the value for 2 byte codes (from 0 to 65535), in hexadecimal format.

For example, if I need 80 Chocolumps, the code is 02020D6A:50, and if I need to have 999 Batmos killed, the code is 02020F72:03E7

If you need a value in hex, I advise you to use the calculator included with Windows, if you select the option Scientific, you'll be able to calculate hex values. Otherwise, I'll leave some values here:

DEC	HEX
01	01
09	09
10	0A
99	63
999	03E7

The codes will be listed in the following format:

Code 02020XXX:YY		
Replace XXX with:		
USA	EUR	Item
D38	D48	Cactus Essence
D3A	D4A	Mana Pendant
D3C	D4C	Rusty Sword
D3E	D4E	Moon Mirror
...		

The first part of the code is given (02020, in this case), and you have to replace the last 2 or 3 digits of the memory address for the ones in the chart, depending on the rom version you're using [QUICK HINT: when you start the game, your character will be having a bad dream. If you see an intro with a tree and some legend about the tree and the goddess and such, you're playing the USA version. If the intro is not there, you're playing the EUR version]. As for the YY, it represents the amount of such item. Say, if you need 99 of them, you will replace it for 63 (that's 99 in hex).

==========

3.1. Items

3.1.1. Event Items

Code 02020XXX:YY

Replace XXX with:

USA	EUR	Item
D38	D48	Cactus Essence
D3A	D4A	Mana Pendant
D3C	D4C	Rusty Sword
D3E	D4E	Moon Mirror
D40	D50	Control Room Key
D42	D52	Gold Key
D44	D54	Silver Key
D46	D56	Leaflet
D48	D58	Glittering Sword
D4A	D5A	Glittering Armour
D4C	D5C	Glittering Helm
D4E	D5E	Dudbear's Gold
D50	D60	Honey
D52	D62	.Keepsake Pendant
D54	D64	. Silver Knife
D56	D66.	Blood Pouch
ED0	EE0	Courtney's Letter
ED2	EE2	Kurt's Letter
ED4	EE4	Barbecued Newt
ED6	EE6	Barbecued Tail
ED8	EE8	Invoice
EDA	EEA	Black Mask
EDC	EEC	Moon Drop
EDE	EEE	Cancun Feather
EE0	EF0	BubU Worm
EE2	EF2	Light Geode
EE4	EF4	Dark Geode
EE6	EF6	Moon Geode
EE8	EF8	Fire Geode
EEA	EFA	Water Geode
EEC	EFC	Wood Geode
EEE	EFE	Wind Geode
EF0	F00	Earth Geode
EF2	F02	Sword of Mana

3.1.2. Recovery Items

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
58	68	Gumdrop
5A	6A	Chocolump
5C	6C	Honey Elixir
5E	6E	Magic Walnut
60	70	Prestoveggie
62	72	Stardust Herb
64	74	Angel Grail
66	76	GummiFrog

3.1.3. Mistery Items

Remember that Amigo Whistle only appears during battle.

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
68	78	Blink Weed
6A	7A	Potent Posy (Fragrant Flower)
6C	7C	Tone Stone
6E	7E	Dream Reed
70	80	Magic Rope
72	82	Poi Poi's Notebook (Not an actual item, more like a menu ^^)
74	84	Tiny Tapper
76	86	Amigo Whistle

3.1.4. Trait Coins

Remember that Trait Coins only appear during battle.

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
78	88	Light Coin
7A	8A	Dark Coin
7C	8C	Moon Coin
7E	8E	Fire Coin
80	90	Water Coin
82	92	Wood Coin
84	94	Wind Coin
86	96	Earth Coin

3.1.5. Spirit Icons

Remember that Spirit Icons only appear during battle.

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
88	98	Wisp Icon
8A	9A	Shade Icon
8C	9C	Luna Icon
8E	9E	Salamander Icon
90	A0	Undine Icon
92	A2	Dryad Icon
94	A4	Jinn Icon
96	A6	Gnome Icon

3.1.6. Meat Items

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
98	A8	Animal Meat
9A	AA	Thin Meat
9C	AC	Insect Meat
9E	AE	Lizard Meat
A0	B0	Bird Meat
A2	B2	Morph Meat
A4	B4	Fish Meat
A6	B6	Magical Meat
A8	B8	Tough Meat
AA	BA	Rotten Meat
AC	BC	Demon Meat
AE	BE	Dragon Meat
B0	C0	Odd Meat
B2	C2	Mixed Meat
B4	C4	Spicy Meat
B6	C6	Phantom Meat

3.1.7. Seeds

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
B8	C8	Round Seed
BA	CA	Small Seed

BC	CC	Oblong Seed
BE	CE	Long Seed
C0	D0	Crooked Seed
C2	D2	Flat Seed
C4	D4	Big Seed
C6	D6	Spiny Seed

3.1.8. Fruits

Code 02020DXX:YY

Replace XX with:

USA	EUR	Item
C8	D8	Bellgrapes
CA	DA	Diceberry
CC	DC	Peach Puppy
CE	DE	Applesocks
D0	E0	Orange'opus
D2	E2	Citrisquid
D4	E4	Springanana
D6	E6	Mangolephant
D8	E8	Rocket Papaya
DA	EA	Loquat-Shoes
DC	EC	Boarmelon
DE	EE	Pine O'Clock
E0	F0	Rhinolupe
E2	F2	Kittypie
E4	F4	Cherry Bombs
E6	F6	Fishy Fruit

3.1.9. Veggies

Code 02020XXX:YY

Replace XXX with:

USA	EUR	Item
DE8	DF8	Lilipods
DEA	DFA	Masked Potato
DEC	DFC	Spiny Carrot
DEE	DFE	Honey Onion
DF0	E00	Cornflower
DF2	E02	Dolphin Squash
DF4	E04	Cabbadillo
DF6	E06	Conchurnip
DF8	E08	Needlettuce
DFA	E0A	Whalamato
DFC	E0C	Orcaplant
DFE	E0E	Mush-in-a-Box

E00	E10	Bumpkin
E02	E12	Garlicrown
E04	E14	Heart Mint
E06	E16	Spade Basil

3.1.10. Weird Items (Thanks to Dan Sawchuk who made me note this weird thing)

I think these codes are useless, they are for some items called "Shields" and "Eggs", but they don't appear in the regular item screen, but in the sell screen in any shop. My guess is that they were supposed to be regular items in the game but they were dropped for some reason. The blank fifth row in the armor subscreen makes me wonder... but I disgress. If you want to see them, though, here they are:

Code 02020EXX:YY

Replace XX with:

USA	EUR	Item
08	18	Holy Shield
0A	1A	Dark Shield
0C	1C	Moonlight Shield
0E	1E	Burning Shield
10	20	SeaMist Shield
12	22	Arbor Shield
14	24	Anemoi Shield
16	26	Terra Shield
18	28	Diamond Shield
A4	B4	Fauna Egg
A6	B6	Flora Egg
A8	B8	Insect Egg
AA	BA	Reptile Egg
AC	BC	Fowl Egg
AE	BE	Amorph Egg
B0	C0	Fish Egg
B2	C2	Magicali Egg
B4	C4	Demihuman Egg
B6	C6	Undead Egg
B8	C8	Demon Egg
BA	CA	Dragon Egg

3.1.11. Accesories

Code 02020EXX:YY

Replace XX with:

USA	EUR	Item
20	30	BB Ring
22	32	Gem Ring

24	34	Cicada Earrings
26	36	Quartz Ring
28	38	Cobra Earring
2A	3A	WhiteLight Ring
2C	3C	Fiend Fang
2E	3E	Bandit Earrings
30	40	Red Moon Horn
32	42	D-Fence Ring
34	44	Mist Pendant
36	46	Knight Crest
38	48	Gjallar Horn
3A	4A	Dragon Choker
3C	4C	Sage Stone
3E	4E	Cardinal Eye
40	50	FlameFlicker
42	52	Draupnir
44	54	General Crest
46	56	Dragon Ring
48	58	Rune Earrings
4A	5A	Code Bead
4C	5C	Wishbone
4E	5E	Crystal Ring
A0	B0	Brownie Ring
A2	B2	Bath Set
CC	DC	Belle Bell
CE	DE	Chimpfish Iris

3.1.12. Raw Materials

Code 02020EXX:YY

Replace XX with:

USA	EUR	Item
50	60	Topple Cotton
52	62	Sultan Silk
54	64	Jadd Hemp
56	66	Altena Felt
58	68	Oak Wood
5A	6A	Holly Wood
5C	6C	Baobab Wood
5E	6E	Charcoal
60	70	Ash Wood
62	72	Dion wood
64	74	Mistletoe Wood
66	76	Fossil Wood
68	78	Animal Hide
6A	7A	Gator Skin
6C	7C	Centaur Hide
6E	7E	Pegasus Hide
70	80	Animal Bone

72	82	Elephant Tusk
74	84	Black Bone
76	86	Fossil
78	88	Menos Bronze
7A	8A	Forsena Iron
7C	8C	Granz Steel
7E	8E	Lorimar Iron
80	90	Altena Alloy
82	92	Maia Lead
84	94	Mythril Silver
86	96	Orichalcum
88	98	Fish Scale
8A	9A	Lizard Scale
8C	9C	Snake Scale
8E	9E	Dragon Scale
90	A0	Jake Aerolite
92	A2	Hal Aerolite
94	A4	Anhk Aerolite
96	A6	Vinek Aerolite
98	A8	Marble
9A	AA	Obsidian
9C	AC	Pedan Stone
9E	AE	Crystal

3.1.13. Summon Items

Remember that Summon Items only appear during battle.

Code 02020EXX:YY

Replace XX with:

USA	EUR	Item
BE	CE	Selva Card
C0	D0	Pokiehl Card
C2	D2	Tote Card
C4	D4	Rosiotti Card
C6	D6	Olbohn Card
C8	D8	Gaia Card
CA	DA	Matilda Card

3.1.14. Lucre

Lucre is set in 3 bytes. The addresses are

USA VERSION	EUR VERSION
02020D20:XXXX	02020D30:XXXX
02020D22:YY	02020D32:YY

The maximum lucre is 99999 (01869F in hex), so you'll need the codes

USA VERSION	EUR VERSION
02020D20:869F	02020D30:869F
02020D22:01	02020D32:01

3.1.15. Monsters Killed

This codes are really useful for getting the black monsters.
I'm NOT writing down the name of each monster, you have the
numbers in the Popoi's notebook :p

[For a quick alternative with Codebreaker, see the bottom of the chart]

Code 0202XXXX:YYYY

YYYY is 2 byte, thus, for 999 monsters killed, use 03E7,
or use 03E8 for 1000 (Black) monsters.

Replace XXX with:

USA	EUR	Monster #
0F38	0F48	001
0F3A	0F4A	002
0F3C	0F4C	003
0F3E	0F4E	004
0F40	0F50	005
0F42	0F52	006
0F44	0F54	007
0F46	0F56	008
0F48	0F58	009
0F4A	0F5A	010
0F4C	0F5C	011
0F4E	0F5E	012
0F50	0F60	013
0F52	0F62	014
0F54	0F64	015
0F56	0F66	016
0F58	0F68	017
0F5A	0F6A	018
0F5C	0F6C	019
0F5E	0F6E	020
0F60	0F70	021
0F62	0F72	022
0F64	0F74	023
0F66	0F76	024
0F68	0F78	025
0F6A	0F7A	026
0F6C	0F7C	027
0F6E	0F7E	028
0F70	0F80	029
0F72	0F82	030
0F74	0F84	031
0F76	0F86	032
0F78	0F88	033
0F7A	0F8A	034
0F7C	0F8C	035
0F7E	0F8E	036

```
0F80  0F90  037
0F82  0F92  038
0F84  0F94  039
0F86  0F96  040
0F88  0F98  041
0F8A  0F9A  042
0F8C  0F9C  043
0F8E  0F9E  044
0F90  0FA0  045
0F92  0FA2  046
0F94  0FA4  047
0F96  0FA6  048
0F98  0FA8  049
0F9A  0FAA  050
0F9C  0FAC  051
0F9E  0FAE  052
0FA0  0FB0  053
0FA2  0FB2  054
0FA4  0FB4  055
0FA6  0FB6  056
0FA8  0FB8  057
0FAA  0FBA  058
0FAC  0FBC  059
0FAE  0FBE  060
0FB0  0FC0  061
0FB2  0FC2  062
0FB4  0FC4  063
0FB6  0FC6  064
0FB8  0FC8  065
0FBA  0FCA  066
0FBC  0FCC  067
0FBE  0FCE  068
0FC0  0FD0  069
0FC2  0FD2  070
0FC4  0FD4  071
0FC6  0FD6  072
0FC8  0FD8  073
0FCA  0FDA  074
0FCC  0FDC  075
0FCE  0FDE  076
0FD0  0FE0  077
0FD2  0FE2  ---   \
0FD4  0FE4  ---    |--> Not listed, probably dropped.
0FD6  0FE6  ---   /
0FD8  0FE8  078
0FDA  0FEA  079
0FDC  0FEC  080
0FDE  0FEE  081
0FE0  0FF0  082
0FE2  0FF2  083
0FE4  0FF4  084
0FE6  0FF6  085
0FE8  0FF8  086
0FEA  0FFA  087
0FEC  0FFC  088
0FEE  0FFE  089
0FF0  1000  090
0FF2  1002  091
0FF4  1004  092
0FF6  1006  093
```

```
0FF8  1008  094
0FFA  100A  095
0FFC  100C  096
0FFE  100E  097
1000  1010  098
1002  1012  099
1004  1014  100
1006  1016  101
1008  1018  102
100A  101A  103
100C  101C  104
100E  101E  105
1010  1020  106
1012  1022  107
1014  1024  108
1016  1026  109
1018  1028  110
101A  102A  111
101C  102C  112
101E  102E  113
1020  1030  114
1022  1032  115
1024  1034  116
1026  1036  117
1028  1038  118
102A  103A  119
```

Ok, writing down all of these codes is plain nasty. But behold, oh VBA users, for we have the CODEBREAKER!!!
CODEBREAKER basically is the same as putting the codes manually, but it has a few pros and cons: It's more complicated, but much more confortable.

We will use a "slide code", that is, a code that repeats itself until we say so. In this case, we want it for our 119 monsters, rite? So, letIs select a CODEBREAKER code (top right button in the cheat menu) and put this code: (Replace YYYY with the number of your choice. 1000 is 03E8 in hex):

```
USA VERSION     EUR VERSION
-------------   -------------
42020F38 YYYY   42020F48 YYYY
0000004D 0002   0000004D 0002
42020FD8 YYYY   42020FE8 YYYY
0000002A 0002   0000002A 0002
```

That's it. Easy, right? For those of you that want to know what the heck are you doing here, keep reading the appendixes, specially the 4.3. Codebreaker.

3.2. Hero specific codes

3.2.1. Level & XP

Your player level is coded in hex, as usual. Remember that if you want lvl 99 you'll have to use hex code 63. Be warned, if you use this code, you'll bypass the lvl-up process, thus not acquiring the stat bonuses nor the class upgrades!

Actually, this code is kinda useless xD

USA VERSION	EUR VERSION
020203F2:YY	02020402:YY

The real way to do this is to mess with the gained XP. This address uses 2 bytes, meaning that if you need 5000 XP, you'll use YYYY = 1388.

USA VERSION	EUR VERSION
02020424:YYYY	02020434:YYYY

Once you use this code, the next enemy you kill will make the "lvl Up" pop up, and you'll have to go through the level up process until you reach the level you're supposed to have with that amount of XP (No idea how much XP for each lvl, sorry). Don't put a big number here (as FFFF) or else the game may freeze.

3.2.2. 999 HP & MP

For a full explanation on how to put different values of HP and MP, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

USA VERSION	EUR VERSION
020203F8:E7	02020408:E7
020203F9:9F	02020409:9F
020203FA:7F	0202040A:7F
020203FB:BE	0202040B:BE
0202040C:CE	0202041C:CE
0202040D:07	0202041D:07 (*)

(*) This code depends directly on the Rod Weapon Level. In this case, as we are playing as the Hero, we don't care. But, if you wanted to have a lvl 99 rod nevertheless, you'd have to put 0202040D:1F instead.

3.2.3. Weapon Levels

For a full explanation on how to put different values other than 99, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

The following list is for all weapons lvl 99.

USA VERSION	EUR VERSION
020203F3:63	02020403:63
020203F6:18	02020406:18
020203F7:03	02020407:03
0202040D:18	0202041D:18 (2)
0202040E:8F	0202041E:8F
0202040F:31	0202041F:31
02020410:1E	02020420:1E
02020411:8F	02020421:8F
02020412:C7	02020422:C7
02020413:18	02020423:18

02020414:03   02020424:03 (1)

   A lot of this lines messes some other parts of the game, so it's not a good idea to let this code activated. For inputting this, it's better to save the game if you wish, pause the game, input the code, go back to the game, and back again to the cheat screen in order to delete all of these codes.

(1) This line messes up the XP of your sword. By using this, you're setting it to 0. If you'd like to put a high value on the sword XP as a bonus, use 02020414:FF instead.

(2) This line depends on the Max HP. If you are using this along with the 999 max HP code, you'll have to use 0202040D:1F instead. Here's a full list of the actual value you have to input here, depending on your current Max HP (that is, if you don't want to accidentally change it)

Max HP	Value
0 to 127	18
128 to 255	19
256 to 383	1A
384 to 511	1B
512 to 639	1C
640 to 767	1D
768 to 895	1E
896 to 999	1F

3.2.4. Spirit Levels

For a full explanation on how to put different values other than 99, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

The following list is for all spirits lvl 99.

USA VERSION	EUR VERSION
02020421:30	02020431:30
02020422:1E	02020432:1E
02020423:8F	02020433:8F
02020424:61	02020434:61
02020425:3C	02020435:3C
02020426:1E	02020436:1E
02020427:8F	02020437:8F
02020428:C1	02020438:C1
02020429:18	02020439:18

   A lot of this lines may mess some other parts of the game, so it's not a good idea to let this code activated. For inputting this, it's better to save the game if you wish, pause the game, input the code, go back to the game, and back again to the cheat screen in order to delete all of these codes.

3.2.5. Class Levels

   There's a problem here. If you activate this codes, you'll automatically get a high level some classes, but you won't get the bonuses you get when you go up a level. Also, if I put, for example, all the classes in, let's say, lvl 50,

next time I get a level up, I'll have a hard time selecting which "job" I'll get, because your stats are likely to met full conditions for more than one job.

So, I'll put here the variables you need in order to get a specific setting, along with the formula and ranges for each one. For more information, please refer to section 4.5.

Each class has it's formula. For each formula, we'll have 2 or three numbers called variables, which we get by doing simple math.

```
Warrior level:
               Take your level and divide it by 8.
               The result is A1, the integer rest is A0

               Example: Level 29 --> 29:8 = 3
                                     The rest is 29 - 3*8 = 5
                                     So, A1 = 3, A0 = 5

Monk Level:
               Convert the level into hex. You'll have, in order, B1 and B0.

Magician Level:
               Divide the level by 32. The result is C2.
               Take the rest and divide it by 2. The result is C1.
               and the rest is C0.
               Example: Level 45 --> 45:32 = 1 ==> C2 = 1
                                     Rest = 45 - 1*32 = 13
                                     13:2 = 6 ==> C1 = 6
                                     Rest = 13 - 2*6 = 1 ==> C0 = 1

Sage Level:
               Divide the level by 64. The result is D2.
               Take the rest and divide it by 4. The result is D1.
               and the rest is D0.
               Example: Level 45 --> 45:64 = 0 ==> D2 = 0
                                     Rest = 45 - 0*64 = 45
                                     45:4 = 11 ==> D1 = B (11 in hex)
                                     Rest = 45 - 4*11 = 1 ==> D0 = 1

Thief Level:
               Take your level and divide it by 8.
               The result is E1, the integer rest is E0
               Same as in Warrior Level.

Random Level:
               Convert the level into hex. You'll have, in order, F1 and F0.
```

Once you have all of your desired variables, let's make the codes, shall we? For each code we need to calculate both the first and second words. The first X represents the first word and Y the second one. All of the addresses start with 02020DXX

USA	EUR	X (1st digit)	Y (2st digit)
2F	3F	X = 2*A0	Y = A1
30	40	X = B1 + 8*C0	Y = B0
31	41	X = C2 + 4*D0	Y = C1
32	42	X = D2 + 2*E0	Y = D1
33	43	X = F0	Y = E1
34	44	X = 0	Y = F1

Let's see an example. I'm already a Cleric (That is, I've 5 levels in Sage), and I want to be a Priest, for which I need 5 levels in Magician and 10 levels in Sage. So, I need A = 0, B = 0, C = 5 , D = 10, E = 0, F = 0. Let's do the maths, shall we?

A) A = 0 means result and rest equals 0, hence A0 = 0 and A1 = 0.

B) 0 in hex is 00, so B0 = 0, B1 = 0.

C) We divide 5 by 32. The result is 0, and the rest is 5. => C2 = 0
We take the rest (5), divide it by 2, and we have result 2 and rest 1, thus C1 = 2, C0 = 1.

D) We divide 10 by 32. The result is 0, and the rest is 10. => D2 = 0
We take the rest (10), divide it by 4, and we have result 2 and rest 2, thus D1 = 2, D0 = 2.

E) It's all 0's, so E1 = 0 and E0 = 0

F) Same as above, F1 = 0, F0 = 0.

Now let's make the codes.

```
Address
2F       X = 2*A0       = 2*0       = 0 , Y = A1 = 0 => 02020D2F:00
30       X = B1 + 8*C0  = 0 + 8*1   = 8 , Y = B0 = 0 => 02020D30:80
31       X = C2 + 4*D0  = 0 + 4*2   = 8 , Y = C1 = 2 => 02020D31:82
32       X = D2 + 2*E0  = 0 + 2*0   = 0 , Y = D1 = 2 => 02020D32:02
33       X = F0         = 0             , Y = E1 = 0 => 02020D33:00
34       X = 0          = 0             , Y = F1 = 0 => 02020D34:00
```

The only ones we really need to use are the ones that have not only 0's.
Please note you won't automatically win the Priest class, but you'll have to go up yet another level, whether normally or with the XP cheat code. Then, you go up a level in some class (related to your desired class) and just then you'll get your class lvl up.

3.2.6. Stats

Code 02025XXX:YYYY

YYYY is 2 byte, thus, for 999 in all stats, use 03E7

Replace XXX with:

USA	EUR	Stat
A10	6E0	POW
A12	6E2	DEF
A14	6E4	INT
A16	6E6	MND
A18	6E8	AGI

3.3. Heroine specific codes

Many of the heroine specific codes are just the same as the hero codes, with an hex 0058 added to the hero address.

3.2.1. Level & XP

Your player level is coded in hex, as usual. Remember that if you want lvl 99 you'll have to use hex code 63. Be warned, if you use this code, you'll bypass the lvl-up process, thus not acquiring the stat bonuses nor the class upgrades! Actually, this code is kinda useless xD

USA VERSION	EUR VERSION
0202044A:YY	0202045A:YY

The real way to do this is to mess with the gained XP. This address uses 2 bytes, meaning that if you need 5000 XP, you'll use YYYY = 1388.

USA VERSION	EUR VERSION
02020498:YYYY	020204A8:YYYY

Once you use this code, the next enemy you kill will make the "lvl Up" pop up, and you'll have to go through the level up process until you reach the level you're supposed to have with that amount of XP (No idea how much XP for each lvl, sorry). Don't put a big number here (as FFFF) or else the game may freeze.

3.2.2. 999 HP & MP

For a full explanation on how to put different values of HP and MP, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

USA VERSION	EUR VERSION
02020450:E7	02020460:E7
02020451:9F	02020461:9F
02020452:7F	02020462:7F
02020453:BE	02020463:BE
02020464:CE	02020474:CE
02020465:07	02020475:07 (*)

(*) This code depends directly on the Rod Weapon Level. If you use this code, you have to be sure your Rod level is below 32. If you wanted to have a lvl 99 Rod here, you'd have to put 02020465:1F instead.

3.2.3. Weapon Levels

For a full explanation on how to put different values other than 99, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

The following list is for all weapons lvl 99.

USA VERSION	EUR VERSION

```
0202044B:63   0202045B:63
0202034E:18   0202045E:18
0202034F:03   0202045F:03
02020465:18   02020475:18 (2)
02020466:8F   02020476:8F
02020467:31   02020477:31
02020468:1E   02020478:1E
02020469:8F   02020479:8F
0202046A:C7   0202047A:C7
0202046B:18   0202047B:18
0202046C:03   0202047C:03 (1)
```

A lot of this lines messes some other parts of the game, so it's not a good idea to let this code activated. For inputting this, it's better to save the game if you wish, pause the game, input the code, go back to the game, and back again to the cheat screen in order to delete all of these codes.

(1) This line messes up the XP of your sword. By using this, you're setting it to 0. As we are playing as the heroine, we don't care much. but, if you'd like to put a high value on the sword XP as a bonus, use 0202046C:FF instead.

(2) This line depends on the Max HP. If you are using this along with the 999 max HP code, you'll have to use 02020465:1F instead. Here's a full list of the actual value you have to input here, depending on your current Max HP (that is, if you don't want to accidentally change it)

Max HP	Value
0 to 127	18
128 to 255	19
256 to 383	1A
384 to 511	1B
512 to 639	1C
640 to 767	1D
768 to 895	1E
896 to 999	1F

3.2.4. Spirit Levels

For a full explanation on how to put different values other than 99, please refer to the appendixes, section 4.4. The codes in Sword of Mana.

The following list is for all spirits lvl 99.

```
USA VERSION   EUR VERSION
-----------   -----------
02020479:30   02020489:30
0202047A:1E   0202048A:1E
0202047B:8F   0202048B:8F
0202047C:61   0202048C:61
0202047D:3C   0202048D:3C
0202047E:1E   0202048E:1E
0202047F:8F   0202048F:8F
02020480:C1   02020490:C1
02020481:18   02020491:18
```

A lot of this lines may mess some other parts of the game, so it's not a good idea to let this code activated. For inputting this, it's better to save

the game if you wish, pause the game, input the code, go back to the game, and back again to the cheat screen in order to delete all of these codes.

3.2.5. Class Levels

See the Hero codes section for an explanation on how to make the codes.

All the codes start with 02020DXX

USA	EUR	X (1st digit)	Y (2st digit)
87	97	X = 2*A0	Y = A1
88	98	X = B1 + 8*C0	Y = B0
89	99	X = C2 + 4*D0	Y = C1
8A	9A	X = D2 + 2*E0	Y = D1
8B	9B	X = F0	Y = E1
8C	9C	X = 0	Y = F1

3.2.6. Stats

Code 02025XXX:YYYY

YYYY is 2 byte, thus, for 999 in all stats, use 03E7

Replace XXX with:

USA	EUR	Stat
A10	6E0	POW
A12	6E2	DEF
A14	6E4	INT
A16	6E6	MND
A18	6E8	AGI

---

4. Appendixes

4.1. Hexadecimal, Decimal and Binary

Computers don't count as we do. We are used to count by using the number 10, meaning that we have 10 separated symbols for expresing quantities (0 to 9). When we reach the number following 9, we are out of symbols, and we start using two symbols at once, starting from the next in the chart. Thus, we write 10.

But, what if we had only 8 symbols? What if we didn't know 8 and 9, for example? We could count 0, 1, 2, 3, 4, 5, 6, 7, and we are out of symbols. So, by following the previous reasoning, we should start using 2 symbols. Which ones? 1 and 0, of course. Then, the following number is 10.

By now, many of you are saying "What are you talking about? If we have the number 7, the next is 8!!" Of course you're right, I'm just stating WHAT IF we didn't have 8 and 9 as numbers. This is what we mathematicians (yes, I am) call

BASE-N numbers.

But why is this relevant to us? Let's return to the computers. AS we said, computers don't have 10 fingers, so it's not natural for them to count with 10 digits. The only natural thing for computers is electricity. If you have an electrical circuit, whether electricity is flowing or not (or, the current in the circuit is low or high). So, computers only know about 2 digits, we may call them ON and OFF, or 1 and 0, or Y and N, as you wish.

What we are about to see is what happens when we try to count with a number of digits other than 10.

4.1.1. Binary (Base-2)

Binary is the name of the BASE-2 numbers. That is, we have ONLY 2 DIGITS for write down numbers, just 0 and 1. So, how are we supposed to count to, let's say, 9?. As we said before, 0 is plain zero, and 1 is plain 1. So far, so good, but what about the next number? Of course, once we are out of symbols, we start stacking them. The next number will be 10. The next one is logical, is 11. Once here, we are again out of symbols. So we'll put yet another one. The next number is 100. Lets clarify this:

REGULAR NUMBERS BASE-10	BINARY NUMBERS BASE-2
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
...	...

It's not really that hard. Many scientific calculators include a base utility, allowing you to find out the binary notation of any number. There's even a geek joke, that says "There are only 10 kind of people. The ones that understand binary and the ones that don't" [10 is binary for 2 xD]

Well, back to computers. When you store a number, character, or symbol in a computer, it's stored as a code (for example, @ is 64, A is 65, and so on), but in binary mode. This means that for each capital A you see, what the computer is seeing is the number 01000001. In a hard drive or flash drive, each 1 or 0 represents a magnetic status. By setting this micro magnets up or down, we store or read data from a drive. Each one of this "states" (up or down; on or off) is called BIT. We use bits a lot in normal life. Internet speed is measured in bits per second (actually, more like thousands or millions of bits per second). So, if you have a connection of 100 Mbps, you are transfering to your computer roughly 100 million bits per second.

So, binary code is really something if you want to understand computers.

4.1.2. Hexadecimal (Base-16)

Let's face it. Binary code may be easy to use for a computer, but is quite annoying for us to deal with. Why is that? Because for a relatively small number, such as 200, we have in binary 11001000. Eight bloody numbers, instead of the regular three we need in our normal life. That's why binary code is usually packed in sets of four, called WORDS. How many "words" are there? Let's count them, shall we?

BINARY WORD	VALUE	HEX VALUE
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

We have 16 different words, representing the numbers from 0 to 15. But we had a name for when this things happen. This is BASE-16, or hex (short for hexadecimal). We have 16 "numbers" we can use to represent quantities.

Ok, we all know the first 10 numbers, from 0 to 9. But we have no special symbol for 10. In fact, as we don't have one, we write it down using TWO symbols (1 and 0). So, if we want to have 16 different symbols, we have to make some of them up. Mathematicians used the first letters of the alphabet, thus the symbols for hex are the ones in the rightmost column.

Ok, let's count. Until 15 there's no problem, we can still write the number down using just one digit. For the next number (our regular 16), we are out of symbols. What do we do? We use two symbols, and we have 10. (ALWAYS REMEMBER: number 10 in our regular system is different from 10 in another system). From there on, we keep counting. Our 17 is hex 11, our 18 is hex 12, and so on.

Computer code is almost always represented this way, and the utility that lets you peek into it is called a "hex editor". When scientists invented computers, they decided that 256 symbols were all you can possibly need for writing letters, numbers and stuff (and, by that time, it was good). So, they programmed the computers so you can store data using 256 different types of symbols per character (such as @, p, 8... 256 of them). Each one of these characters is called BYTE. As you can guess, when the computers began to spread worldwide, 256 symbols were not enough. But that's another story ^_^.

Let's see how many bits we need to express a byte, and the answer is 8 bits. So, each 8 bits we have a byte. 8 bits is also two "words", so, each byte can be written down using two hex symbols. For example, if a byte contains the symbol 200 (it's the plus symbol, by the way), it will appear in a hex editor as "C8", meaning that the binary representation for that number is:

```
1100 1000
 C    8
```

Transforming numbers from and to hex is not a nice task, as it needs many calculations. Your best chance is to have a scientific calculator. However, for those of you that want to know, we are going to learn how to do it.

4.1.3. Base transformation

First we are going to see the "easy" part. How to convert a hex number into a normal number (actually, the "normal" numbers are called DECIMAL NUMBERS, or BASE-10 NUMBERS).

First thing you have to remember is the value of each digit. Refer to the chart in 4.1.2. if you need to.
Ok, let's do it. In order to transform a hex number into decimal, we have to make some calculations. We have to multiply the value of each digit to a number that represents it's position. For the last digit we use 1, for the next, we use 16. For the next, 16*16 = 256, for the next 16*16*16 = 4096, and so on. Finally, we sum up all the results.
Let's use a 4-digit hex number as an example:

```
5  C  3  A
|  |  |  |
|  |  |  ¬-----> 10 *    1 =    10     (A means 10)
|  |  |
|  |  ¬--------> 3  *   16 =    48
|  |
|  ¬-----------> 12 *  256 =  3072     (C means 12)
|
¬--------------> 5  * 4096 = 20480
                             -----
                      Total = 23610
```

And now, the nasty part. The opposite way. Lets say we need to transform the number 12500 into hex. What we need to do is start dividing the number by 16, and write down the integer rest or modulus of the operation. The, you continue doing this with the result of the previous division, until you can't divide it any more.

```
12500 : 16 = 781, Rest = 4
781   : 16 =  48, Rest = 13   [This is D in hex]
48    : 16 =   3, Rest = 0
```

Once we are done, we start writing down the LAST division result, followed by the rests or moduli, written backwards:

12500 dec = 30D4 hex

Piece of cake, huh?

==========
4.2. How to make your own codes
==========

4.2.1. Basics

What a memory cheat code does, is to search for a specified memory address, and stick a given value into it. If you keep the cheat code activated, this value never changes, and so you can have infinite values (life, energy, money). Or, alternatively, you can activate the cheat code, save the game (by battery save or by state save), deactivate the code and continue playing. In this case,

you're just setting a value but after that it can go up or down, depending on your actions.

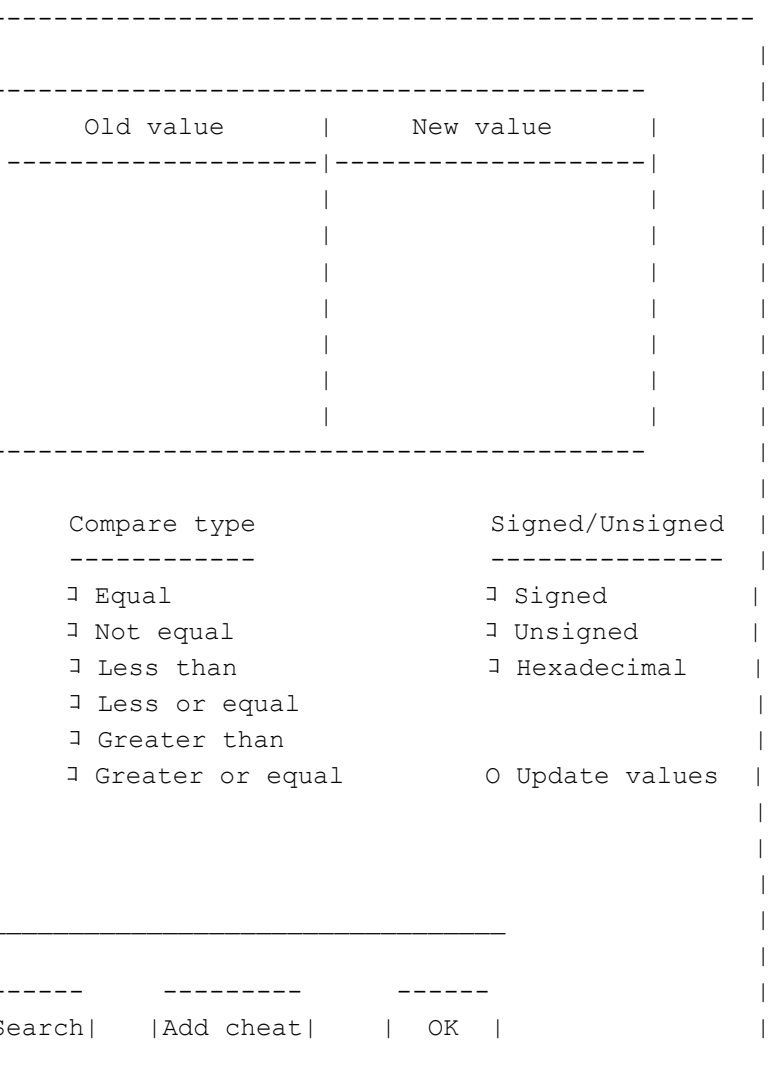
For example, I can have a code that sets a timer to 1 minute. If I keep the cheat code running, the timer will be ALWAYS at 1 minute. This could be a problem if, for example, you get points for the time remaining. Some games use the actual timer value and start decreasing it (we all remember Super Mario Bros. xD). If you froze up the value, you'll be stuck there, getting infinite points for your time, but as the timer never goes down, the process will loop to infinite. What's the moral of this? Be careful, use the codes for what they are supposed to, and be extra careful with the "infinite" codes.

Ok, so we want to make our own code. What do we need? Only two things. The memory address (or addresses) where the value you want to change is, and the new value you want to replace it with. The value is usually easy, i.e. you need 9 lives, you're using value 9. There are some cases (specially with newish games) where the values are encrypted, or coded not byte-wise but bit-wise (This game - Sword of Mana - is a good example of this), and you need to analyze the results.

First, we're learning how to look for a memory address using the cheat menu included with Visual Boy Advance. Then, we'll be seeing how to analyze and decypher encrypted data.

4.2.2. Search

VBA includes a neat cheat utility, that lets you look for specific values, or check values as they go up and down. Let's familiarize with this utility. Go to Cheats - Search for Cheats. The following windows appears:



| ----- ------ --------- ------ |

There are two types of search. Specific or related. We use specific value search when we know for sure the value we're looking for. Let's say, we have currently 5 lives. So, we go to the search window and select "Specific value".

Next, we have to select the size. This depends on the maximum allowed value of the data. 8 bits is for numbers between 0 and 255 (usually lives fall into this range). 16 bits is for values between 0 and 65535. Larger quantities, such as HP, MP fall here. Last, 32 bits, calls for huge numbers, anything less than 4294967295 is good here. Usually the money you carry and the XP fall here. The thing is, we can't know how the game is coded. Maybe they put the value of HP, that ranges between 0 and 999, in only 12 bits of data. You have to cross your fingers and select the one you THINK is right. Following the example for the 5 lives, we select "8 bits".

Then we select how to check the data. Are we looking exactly for a specific number? Or just anything below is good? Or maybe we need to search for the numbers greater than 1000. Select your choice here. In our example, we are setting this to "Equal"

The "Signed/Unsigned" box refers to the type of number we are dealing with. 99% of the time we'll be using "unsigned", that means that a 8-bit value, goes from 0 to 255, while a signed one goes from -128 to 127. Hexadecimal is for when you'd like to search in a more pro way, but if you know the decimal value, unsigned is fine (values coded in BCD -binary coded decimal- are a pain, they use hex values but they are written as decimal. This means that if you want to search for decimal 45, you have to search for HEX 45).

The checkbox "Update values" is used when you're searching for an elusive address. If you check this, every time you access the cheat window, the values in the "New value" and "Old value" column are updated.

Finally, the value dialog is where you enter the number you're looking for. In our example, we'll be writing down 5.

Once you're done with this, press Start, and next Search. The results will be displayed in the upper screen. Chances are, there are too many addresses with the value 5, so the system will tell you it cannot display that many numbers. If this is not the case, surely you'll be stuck with a bunch of memory addresses and you cannot know for sure which is the one you need. So, what do we do? Press OK and go back to the game. Continue playing and purposely lose a life. Now you have 4. Go back to the screen, and put 4 in the value dialog. This time DON'T PRESS START, as we will lose the previous search. Just press SEARCH and the system will search for the number 4 among the previous values. This will narrow our search a bit (quite a bit, hopefully). If you still have too many results, lose or gain another life and search again. Once we are done, we will have only one or two (maybe three) results. Now we save the game, just in case. Back to the cheat screen, we click one of the results and with that row selected, press Add cheat. A screen will pop up asking you for a new value. Most of the options are disabled, so just put the new value we want (let's say 9) and add a description if you want (if you are using multiple codes DO THIS, or you'll end with a bunch of codes and you won't know which is which). Press OK and go back to the game. If you selected the proper option, the life number should have changed by now. If not, you may have selected the wrong address, or maybe the game does not update the value on the screen, but it does change the value, internally.

This won't happen always this smoothly. sometimes you don't know the real value you're looking for. Let's say we're playing an adventure game, and the HP is displayed as a bar, not as a number. We don't know the actual HP value, so we go to the cheat screen and select "Old value". The value dialog is grayed out, so we just press START and leave this window by clicking OK. We go back to the game, lose some HP and go back to the cheat window. This time, select the

option "Lesser than", as the HP value surely has decreased. Click SEARCH, and the screen will display ALL the values that have decreased since you last pressed START or SEARCH. Do this a few more times, and hopefully you will get your value. For inputting the cheat code, do the same as before. This method is quite slow, but it does work, unless the data is encrypted...

4.2.3. Analisys

Ok, so far we know how to search for memory adresses, given a specific value or by trial and error. But in both cases, we have to trust the programmers have written the code in the same "language" we are using to read. This means, that the values are properly coded in hexadecimal, or at least in BCD. But sometimes they aren't. In this cases, all our efforts to search for an specific value will fail, because the numbers the search engine can look for are not the ones that are in the game. Let's give a simple example. If you read section 4.1.1., you'll see that every computer code is really written in binary, meaning it's nothing more than a bunch of 0s and 1s. They are USUALLY stacked into words of 4 bits and bytes of 8 bits, but not always. Let's say we are programming a game, and we have to put into memory a series of numbers that will go from 0 to 15. For example, weapon stats. Numbers between 0 and 15 fit in 1 word, so if we use 1 byte per number, we are making the code readable, but we are wasting 4 bits per byte. Some programmers would prefer to use only one word for this, thus saving lots of space. But it will be hell for us, because we can only search for full 8-bit packs, not 4-bit. In this example, let's say we have two stats, with values 5 and 9. If they are programmed word-wise instead of byte-wise, you will have to look for the hex value 59 or 95, or worse, because they may be entwined with weird code.

Our options? Be really patient and try. Once you found an address, the rest of them should be "around", relatively near. So, let's say we have a cheat code with the mem address 02020546. The rest of the game data, is more likely to be in the same zone, let's say the 02020XXX zone. So, we save a state in slot #1, try a code at random, for example, 02020500:30, and we see if something changes, and how. Then, we get the saved state back (this action erases all cheat codes not saved with the state), and try 02020500:01. This way we can see if each word (each digit in the value) changes something different. By wasting LOTS of time, we might find some useful/interesting. We'll be seeing examples of this in section 4.4.

4.3 Codebreaker

Codebreaker is a commercial cheat device that is luckily emulated by VBA. You can use encrypted or decrypted codes, and they are easy to manage. The good thing about it is the ability of making slide codes, for changing lots of memory addresses at once; conditional cheating, by changing a value only if a certain criteria is matched; or activated cheats, that remain dormant until you press a certain combination of buttons to activate them.

We'll see now a brief explanation of the codebreaker formats.

4.3.1. Codebreaker formats

As we said before, there are many options for this system. The main code is very similar to a normal hex code, as it consists mainly of an address and a value. but the rest of the things are the juicy part ^_^

This codes can be single or multiple lined. If you use multiple lined codes, you'll have to input all the lines at once.

Formats:

3XXXXXXX 00YY

This is the basic format. It's exactly the same as your regular 1-byte hex code. It sets the value YY into the mem address XXXXXXX.

4XXXXXXX YYYY
AAAABBBB CCCC

The first multilined code. This is for the slide codes, the ones we use for setting multiple cheats at once. The bad news here is that you can only use it with 2 byte or 16-bit codes.
The first line sets the first memory address along with the first value you want for that address, same as you did with the previous code.
The second line has iteration data. AAAA is the incremental step of the value. This is, if you want to set a different value for each address, here you put how many numbers are you rising on each step. Leave it as 0000 if you want all the values to be the same.
BBBB tells us how many iterations are we making. That is, if you need to change 30 consecutive adresses, you will need 001E here(remember to always use hex).
CCCC is the distance, in bytes, between two consecutive addresses. AS we are using 2 byte values, the difference should be at least 2.

6XXXXXXX YYYY

This kind of code is useless if you aren't familiar with binary operations. It does an AND operation between YYYY and the current value of the address. You won't be using this much or at all, don't fret.

7XXXXXXX YYYY

This is the conditional code. It is used in combination with another line below. If the address XXXXXXX has the value set in YYYY, it executes the line below. Otherwise, the next line is skipped. This kind of codes are useful as triggers, as they activate themselves when the criteria is met.

8XXXXXXX YYYY

The regular 2-byte code. It writes the value YYYY into the address XXXXXXX.

AXXXXXXX YYYY

This is the same as the conditional code, but opposite. The next line is only executed if the value of XXXXXXX is NOT equal to YYYY.

D0000020 YYYY

This is also a condicional code. The difference is that in this case, the condition is not a value, but a button that need to be pressed. If the criteria is met, this is, if the correct button or buttons are pressed, the next line is executed.

YYYY is the sum of all the buttons you want to be pressed.
each button has a hex code, you select the ones you'd like
to use, add their values (in hex!) and put this as YYYY.

Button	Code (hex)
A	0001
B	0002
Select	0004
Start	0008
Right	0010
Left	0020
Up	0040
Down	0080
R	0100
L	0200

This means that if we want our code to be activated by
pressing Select + Up, we have to add 0004 + 0040, and then
we use the result (0044) in the YYYY portion of the code.

4.4. The codes of Sword of Mana

As I stated before, many of the values in this game are coded in a weird way. This means that some stats, such as the weapon and spirit levels, the HP / MP, are not stored as regular hex numbers in the game code, but each value has been chopped into pieces and stored in different adresses. This makes the direct way of searching them almost useless, because in a given address, you could find, for example, a piece of the max HP, added to another piece of the Rod level. So unless you have a global vision of all of the codes, looking for them is hard.

I'll put some of my discoveries here. But be warned, it's heavy, boring stuff you should only read if you are really interested, or you are desperately need to set another values but 99 or 999 (if this is the case, mail me and I'll do the math for you ^_^).

For each adress listed, I'll explain what each word does (word is the name of each digit of the byte), along with the formulae to calculate the final results.

Another thing: I'll be using ONLY the USA version addresses here. If you use the EUR version, just add an hex 00000010 to all of the addresses (for example, USA address 02020405 becomes EUR address 02020415).

All of the following addresses start with 02020---, I'll just replace the last three digits here. Each address has 1 byte, coded as XY (for example, 5A).

In the columns X and Y you will find what variable or variables are stored in the first and second word (digit) of each byte.

At the end, an explanation of these variables and how to make the codes work.

ADDRESS	X	Y
3F0	Ch1	Ch0
3F2	L1	L0
3F3	Mc1 + ?	Mc0
3F4	N1	N0
3F6	Sw1	8*Sw0 + ?

```
|  3F7    |     ?      |  Sw2 + ?  |
|  3F8    |     h1     |     h0    |
|  3F9    |     M1     | 4*M0 + h2 |
|  3FA    |     m0     |     M2    |
|  3FB    |  m2  + ?   |     m1    |
|  40C    |     H1     | 2*H0 + ?  |
|  40D    |     R1     | 8*R0 + H2 |
|  40E    |     K1     | 4*K0 + R2 |
|  40F    |     F0     |   K2 + ?  |
|  410    |     S1     | 8*S0 + F1 |
|  411    |     B1     | 4*B0 + S2 |
|  412    |     L1     | 2*L0 + B2 |
|  413    |     A1     | 8*A0 + ?  |
|  414    |     ?      |   A2 + ?  |
.---------.------------.-----------.
```

Allowed values for each variable, separated by stats:

```
ǂ----ǂ------ǂ
|Var | Range|
ǂ----ǂ------ǂ
|Ch1 | 0-F  |
|Ch0 | 0-F  |
ǂ----ǂ------ǂ
|L1  | 0-F  |
|L0  | 0-F  |
ǂ----ǂ------ǂ
|N1  | 0-F  |
|N0  | 0-F  |
ǂ----ǂ------ǂ
|M2  | 0-F  |
|M1  | 0-7  |
|M0  | 0-F  |
ǂ----ǂ------ǂ
|m2  | 0-3  |
|m1  | 0-F  |
|m0  | 0-F  |
ǂ----ǂ------ǂ
|H2  | 0-3  |
|H1  | 0-F  |
|H0  | 0-7  |
ǂ----ǂ------ǂ
|h2  | 0-3  |
|h1  | 0-F  |
|h0  | 0-F  |
ǂ----ǂ------ǂ
```

```
ǂ----ǂ------ǂ
|Var | Range|
ǂ----ǂ------ǂ
|Sw2 | 0-3  |
|Sw1 | 0-F  |
|Sw0 | 0-1  |
ǂ----ǂ------ǂ
|R2  | 0-3  |
|R1  | 0-F  |
|R0  | 0-1  |
ǂ----ǂ------ǂ
|K2  | 0-1  |
|K1  | 0-F  |
|K0  | 0-3  |
ǂ----ǂ------ǂ
|F1  | 0-7  |
|F0  | 0-F  |
ǂ----ǂ------ǂ
|S2  | 0-3  |
|S1  | 0-F  |
|S0  | 0-1  |
ǂ----ǂ------ǂ
|B2  | 0-1  |
|B1  | 0-F  |
|B0  | 0-3  |
ǂ----ǂ------ǂ
|L1  | 0-F  |
|L0  | 0-7  |
ǂ----ǂ------ǂ
```

```
ǂ----ǂ------ǂ
|Var | Range|
ǂ----ǂ------ǂ
|A2  | 0-3  |
|A1  | 0-F  |
|A0  | 0-1  |
ǂ----ǂ------ǂ
|Mc1 | 0-7  |
|Mc2 | 0-F  |
ǂ----ǂ------ǂ
```

Ok, the only thing left is how to combine this numbers in order to make real stuff. Each stat (level, mp, etc...) has a specific formula, using all of the variables we stated. When I say something like X-Y, the hyphen is not a minus symbol, just a separator between words of a hex code.

Character: Ch1-Ch0 (hex)
This code manages the appearence of the main character. This means the aspect, portrait and weapon. BE WARNED, if you put a character here who does not use the SAME weapon as your original char, it won't be able to attack.

Level: L1-L0 (hex)

As I said in the codes section, this is useless, as the level by itself changes nothing.

Inner Level: N1-N0 (hex)
This number tells the game how many exp points you need in order to advance to the next level. This is, if you set this to 1, you will need the same exp points as if you actually were level 1.

Max MP: 64*M2 + 4*M1 + M0 (this is what I meant by encrypted values :p)

Current MP: m2-m1-m0 (hex)

Max HP: 128*H2 + 8*H1 + H0

Current HP: h2-h1-h0 (hex)

Sword lvl: 32*Sw2 + 2*Sw1 + Sw0

Rod lvl: 32*R2 + 2*R1 + R0

Knuckles lvl: 64*K2 + 4*K1 + K0

Flail lvl: F1-F0 (hex)

Scythe lvl: 32*S2 + 2*S1 + S0

Bow lvl: 64*B2 + 4*B1 + B0

Lance lvl: 8*L1 + L0

Axe lvl: 32*A2 + 2*A1 + A0

Mace lvl: 16 * Mc1 + Mc0

Ok. What the heck man? This mumbo jumbo means that in order to change the level of your rod, you have to know beforehand the third "digit" of your max HP or you'll be changing both things by inputting a cheat. So, this practice is reserved only for the bravest xD

I assume this is common practice in some games to avoid easy hacking.

I'll leave this here, because putting all of the codes will result in a real mess. Imagine that we have still left the spirit levels, the stats for every NPC in the game... No way. If you have been reading so far, you should be able to find them out by yourselves by now.

==========

5. Ending

5.1. Thanks

Thanks to Dan Sawchuk who made me note both the difference between the USA and the Euro versions, and the weird Shield and Egg codes.

Thanks to Nate (The Court Judges) for a nice Companion Code I'll be

```
investigating and uploading later ^_^

    Thanks to YOU for reading my guide, I hope it will be useful. For any
question regarding the use of the codes mail me to suarez.lea@gmail.com (don't
send questions about things not covered in the guide, they will be covered as I
find them)


================
5.2. Final words
================

    This guide was written by Leandro Suarez (Curambar in GameFAQS and
Neoseeker). It is not meant to be used on any other site besides GameFAQS,
Neoseeker and any other site I've deemed to have it hosted on. It's not to be
edited in ANY way for other use. (Unless I give permission to do so.)

Copyright ϕ 2009 by Curambar. All rights reserved.
==============================================================================
```