Complete FireRed Upgrade



Contents

[Set Up 3](#_Toc6347027)

[Necessary Modifications 3](#_Toc6347028)

[Configuration Options 4](#_Toc6347029)

[Other Necessary Features Setup 17](#_Toc6347030)

[Mega Evolution / Primal Reversion / Ultra Burst 17](#_Toc6347031)

[Z-Moves 25](#_Toc6347032)

[Trainer Sliding Messages 25](#_Toc6347033)

[Trainer Backsprites 25](#_Toc6347034)

[Battle Frontier 25](#_Toc6347035)

[Battle Music 25](#_Toc6347036)

[Battle Terrain 25](#_Toc6347037)

[Poke Balls 25](#_Toc6347038)

[Pickup 25](#_Toc6347039)

[Time of Day Based Wild Encounters 26](#_Toc6347040)

[Other Features Included 29](#_Toc6347041)

[Save Expansion 29](#_Toc6347042)

[Updated Flutes 29](#_Toc6347043)

[Trainer Face Fix 29](#_Toc6347044)

[Extra Pedometers 29](#_Toc6347045)

[Select from PC Hack 29](#_Toc6347046)

[Dex Nav 29](#_Toc6347047)

[Dynamic Overworld Palettes 30](#_Toc6347048)

[Script Specials 30](#_Toc6347049)

[Creating New Battle Mechanics 55](#_Toc6347050)

[Moves 55](#_Toc6347051)

[Abilities 55](#_Toc6347052)

[Poke Balls 55](#_Toc6347053)

[Credits 56](#_Toc6347054)

# Set Up

## Necessary Modifications

**The following modifications must be made before setting up the engine!**

**Pokemon Defines**:

1. Open the file **include/constants/species.h**.
2. Modify the Pokemon indices found in this file to match the ones in your hack.
3. **DO NOT DELETE ANY POKEMON NAMES**. If there is a Pokemon species you are not using, then set its index to **0xFEFE**.
4. If you have not added in any new Pokemon to Fire Red, you can leave the unused species indices as their default values.

**Item Defines**:

Part 1

1. Open the file **include/constants/items.h**.
2. Modify the item indices found in this file to match the ones in your hack.
3. **DO NOT DELETE ANY ITEM NAMES**. If there is an item you are not using, then set its index to **0xFEFE**.

Part 2

1. Open the file **include/constants/hold\_effects.h**.
2. Scroll down to where it says “//NEW ITEM EFFECTS”. This is the list of new hold item effects. When relevant, the item quality is included.
3. Set up your items in G3T using the example for the *Fire Gem* as a base:



As you can see, the item effect is set to 94 and the item quality is set to 10 (the move type for *Fire*).

**Configuration Options**

See below.

## Configuration Options

There are many configurable options in the file **src/config.h**. These options are meant to provide the user with as much versatility as possible. Below is a more detailed description of each option than show in the configuration file:

**Var Options**

|  |  |
| --- | --- |
| *Flag Definition* | *Description* |
| TOTEM\_VAR | This represents are series of vars using for initiating battles with [Totem Pokemon](https://bulbapedia.bulbagarden.net/wiki/Totem_Pok%C3%A9mon). There are four vars in total, each representing a specific Pokemon slot on the field. Adding the following values to the var will indicate which slot that var is for:  0: *Player Pokemon in Singles, Left Player Pokemon in Doubles*  1: *Enemy Pokemon in Singles, Right Enemy Pokemon in Doubles*  2: *Right Player Pokemon in Doubles*  3: *Right Enemy Pokemon in Doubles*  The vars must be set to the addition of two values. Choose one from each of the following sets:  **Stats**:  1: *Attack*  2: *Defense*  3: *Speed*  4: *Special Attack*  5: *Special Defense*  6: *Accuracy*  7: *Evasion*  **Amount**:  0x10: *Increase Stat by 1*  0x20: *Increase Stat by 2*  0x30: *Increase Stat by 3*  0x40: *Increase Stat by 4*  0x50: *Increase Stat by 5*  0x60: *Increase Stat by 6*  0x90: *Decrease Stat by 1*  0xA0: *Decrease Stat by 2*  0xB0: *Decrease Stat by 3*  0xC0: *Decrease Stat by 4*  0xD0: *Decrease Stat by 5*  0xE0: *Decrease Stat by 6*  So, for instance, having the enemy Pokemon start the battle with its *Attack* raised by *2*, you would set the var TOTEM\_VAR + 1 to the value of 0x21 (0x1 + 0x20). |
| NPC\_FOLLOWING\_VAR | If the *Follow Me* feature is used, this var must be set to the *NPC Id* (*Person Id* in *Advance Map*) of the following NPC. If no NPC is following the player, this var should be set to 0. |
| OW\_SPRITE\_SWITCH\_VAR | Setting this var to a value other than 0 will change the picture used for the player’s Overworld sprite. The value should correspond to the *Picture Id* of the NPC. |
| BACKSPRITE\_SWITCH\_VAR | Setting this var to a value other than 0 will change the default back sprite loaded for the player in battle.  See **src/table/Trainer\_Backsprite\_Table.s** for a list of available backsprites. |
| BATTLE\_BG\_VAR | If CUSTOM\_BATTLE\_BACKGROUNDS is defined, Setting this var to a value other than 0 will cause the regular battle background loaded to be replaced by a custom one.  See **/include/battle.h** for a list of options.  Search for BATTLE\_TERRAIN\_GRASS in the file to see them. |
| SWARM\_SPECIES\_VAR | A var that is automatically set by the engine. It contains the map name Id of the map where there currently is a [*swarm*](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_outbreak) in progress. |
| SWARM\_MAP\_NAME\_VAR | A var that is automatically set by the engine. It contains the species that is currently *swarming*. |
| DEXNAV\_VAR | Holds the species to search for via Select from the Overworld in the DexNav feature. Press *Select* in the DexNav GUI to save. |
| VAR\_PLAYER\_WALKRUN | Switch player walking/running frames. Upper byte is table Id, Eg. Setting to 0x0200 will load the walking/running frames from table 2, sprite 0 |
| VAR\_PLAYER\_BIKING | Switch player biking frames. Eg. 0x0201 loads from table 2 |
| VAR\_PLAYER\_SURFING | Switch player surfing frames. |
| VAR\_PLAYER\_VS\_SEEKER | Switch player VS Seeker frames. |
| VAR\_PLAYER\_FISHING | Switch player Fishing frames |
| VAR\_PLAYER\_VS\_SEEKER\_ON\_BIKE | Switch player Biking/Vs seeker frames |
| VAR\_TRAINERCARD\_MALE | Switch Trainer Card Male Player Front Sprite to a Trainer Id |
| VAR\_TRAINERCARD\_FEMALE | Switch Trainer Card Female Player Front Sprite to a Trainer Id |
| VAR\_RUNTIME\_CHANGEABLE | If a person event is given table Id 0xFF, it can be changed in a runtime setting by changing these variables to a sprite number. Eg. Setting an NPC Id to 0xFF08 will change all NPCs with 0xFF08 to the sprite Id held in VAR\_RUNTIME\_CHANGEABLE+8. Eg. If VAR\_RUNTIME\_CHANGEABLE = 0x4080, the NPCs would change to the sprite value held in Var 0x4088 |
| VAR\_HEALINGMAP | Ported from JPANs engine, set this variable to the map and bank for the player to respawn to after whiting out. Eg. If I want to respawn in the player’s room, I would set this variable to 0x0104. Configure in asm\_defines.s |
| VAR\_HEALING\_XPOS | The x-position the player will respawn at on the map in VAR\_HEALINGMAP. Configure in asm\_defines.s |
| VAR\_HEALING\_YPOS | The y-position the player will respawn at on the map in VAR\_HEALINGMAP. Configure in asm\_defines.s |

**Flag Options**

|  |  |
| --- | --- |
| *Flag Definition* | *Description* |
| INVERSE\_FLAG | Setting this flag will enable [Inverse Battles](https://bulbapedia.bulbagarden.net/wiki/Inverse_Battle).  This flag is automatically cleared at the end of each battle. |
| SKY\_BATTLE\_FLAG | Setting this flag will indicate to the engine that a [Sky Battle](https://bulbapedia.bulbagarden.net/wiki/Sky_Battle) is in progress.  This flag is automatically cleared at the end of each battle. |
| NO\_CATCHING\_FLAG | Setting this flag will cause enemy Pokemon to always dodge balls thrown at them.  This flag is automatically cleared at the end of each battle. |
| NO\_RUNNING\_FLAG | Setting this flag prevents the player from running away during wild battles.  This flag is automatically cleared at the end of each battle. |
| NO\_CATCHING\_AND\_RUNNING\_FLAG | This flag acts as a combination of the above two flags.  This flag is automatically cleared at the end of each battle. |
| CATCH\_TRAINERS\_POKEMON\_FLAG | If CATCH\_TRAINERS\_POKEMON is defined, then setting this flag allows the player to capture Pokemon belonging to the opposing trainer. Capturing a Pokemon in this way will automatically end the battle.  This flag is automatically cleared at the end of each battle. |
| EXP\_SHARE\_FLAG | If OLD\_EXP\_SHARE is not defined, then setting this flag activates the Gen 6+ Exp. Share. |
| DOUBLE\_BATTLE\_FLAG | If ACTIVATE\_DOUBLE\_BATTLE\_FROM\_FLAG is defined, then setting this flag will cause battles against trainers to be [Double Battles](https://bulbapedia.bulbagarden.net/wiki/Double_Battle), if possible. |
| TAG\_BATTLE\_FLAG | This flag is set by the engine when the scripting command *trainerbattle 0x10* is used in a script to activate a tag battle. If setting this flag manually, take care to also set PARTNER\_VAR and PARTNER\_BACKSPRITE\_VAR. This flag is automatically cleared at the end of each battle. |
| TWO\_OPPONENT\_FLAG | This flag is set by the engine when the scripting command *trainerbattle 0x11* is used in a script to activate a battle against two opponents. If setting this flag manually, take care to also set SECOND\_OPPONENT\_VAR. This flag is automatically cleared at the end of each battle. |
| ACTIVATE\_TUTORIAL\_FLAG | Setting this flag activates Professor Oak’s tutorial during the next trainer battle. This flag is automatically cleared at the end of each battle. |
| WILD\_CUSTOM\_MOVES\_FLAG | //TODO |
| SMART\_WILD\_FLAG | Setting this flag allows Wild Pokemon to use the basic AI checks used in trainer battles. This flag is automatically cleared at the end of each battle. |
| SCALE\_TRAINER\_LEVELS\_FLAG | Setting this flag causes all Trainer Pokemon to have levels that match the highest level in your party. |
| HIDDEN\_ABILITY\_FLAG | Setting this flag before a Wild battle causes Wild Pokemon to be generated with their hidden abilities. It is cleared at the end of each battle. |
| DOUBLE\_WILD\_BATTLE\_FLAG | Setting this flag causes all wild battles to be against two wild Pokemon in a Double battle format. It is cleared at the end of each battle. |
| NO\_RANDOM\_WILD\_ENCOUNTERS\_FLAG | Setting this flag will stop Pokemon from appearing while walking through grass or caves, of while surfing on water. Pokemon can still appear if the player chooses to fish or smash rocks |
| FLAG\_LONG\_PEDOMETER  FLAG\_MED\_PEDOMETER FLAG\_SMALL\_PEDOMETER\_1 FLAG\_SMALL\_PEDOMETER\_2 | 4 byte pedometer (max value 0xFFFFFFFF or 4,294,967,295)  2 byte pedometer (max value 0xFFFF or 65,535)  1 byte pedometer (max value 0xFF or 255)  1 byte pedometer (max value 0xFF or 255) |

**Battle Tower Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| BATTLE\_TOWER\_FLAG | Setting this flag indicates to the engine that the Player is in the [Battle Tower](https://bulbapedia.bulbagarden.net/wiki/Battle_Tower). This means that:   * Trainer Pokemon will be generated within the restrictions of the tier set in the var defined in BATTLE\_TOWER\_TIER. The amount of Pokemon generated will match the number set in the var defined in BATTLE\_TOWER\_POKE\_LEVEL. * Trainer Pokemon will have Pokemon with the level contained in the var defined in BATTLE\_TOWER\_POKE\_LEVEL. * The battle format will be loaded from the var defined in BATTLE\_TOWER\_BATTLE\_TYPE. * The music in-battle will be played based on what is contained in the var defined in BATTLE\_TOWER\_SONG\_OVERRIDE. |
| BATTLE\_TOWER\_POKE\_NUM | Setting this var to a value between 1 & 6 dictates the quantity of Pokemon the player and trainers can use in the Battle Tower.  Setting it to 0 will default in 1.  Setting it to a number greater than 6 will default in 6. |
| BATTLE\_TOWER\_POKE\_LEVEL | Setting this var to a value between 1 and what is defined in MAX\_LEVEL will set all Pokemon in the Battle Tower to that level.  Setting it to 0 will default in 1.  Setting to a number greater than MAX\_LEVEL will default in MAX\_LEVEL. |
| BATTLE\_TOWER\_BATTLE\_TYPE | Setting this var to one of the below values set the battle format in the Battle Tower:  0: Single Battle  1: Double Battle  2: Multi Battle  3: Link Multi Battle  Any other value defaults in Single Battle. |
| BATTLE\_TOWER\_TIER | Setting this var to one of the below values indicates to the engine which ruleset should be following for battles in the Battle Tower:  0: Regular Battle Tower Rules  1: No Restrictions  2: [Smogon OU](https://www.smogon.com/dex/sm/formats/ou/)  3: [Smogon Uber](https://www.smogon.com/dex/sm/formats/uber/)  4: [Smogon Little Cup](https://www.smogon.com/dex/sm/formats/lc/)  5: Skeli’s Middle Cup |
| BATTLE\_TOWER\_TID | If battles in the Battle Tower are against a trainer with this trainer Id, a random team will automatically be generated for the trainer. |
| BATTLE\_TOWER\_TRAINER\_NAME | This var is automatically set by the engine to hold the index of the random name for the Battle Tower trainer. It is set to 0xFFFF after every battle. Do not set it manually. |
| BATTLE\_TOWER\_SONG\_OVERRIDE | Setting this var to a song Id will cause that song to be played in Battle Tower battles and Link Battles. |
| TOWER\_TRAINER\_ID\_VAR | This var is set by special 0xXX to indicate which trainer class and details is being spawned as the first opponent. |
| TOWER\_TRAINER\_ID\_2\_VAR | This var is set by special 0xXX to indicate which trainer class and details is being spawned as the second opponent in multi battles. |
| TOWER\_TRAINER\_ID\_PARTNER\_VAR | This var is set by special 0xXX to indicate which trainer class and details is being spawned as the player’s partner in multi battles if the partner is chosen to be randomized. |

**Times of Day**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| TIME\_MORNING\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that morning starts. |
| TIME\_DAY\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that day starts. |
| TIME\_EVENING\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that evening starts. |
| TIME\_NIGHT\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that night starts. |

**Other Number Definitions**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| KANTO\_DEX\_COUNT | Number of Pokemon in the regional Pokedex. |
| NATIONAL\_DEX\_COUNT | Number of Pokemon in the national Pokedex. |
| MAX\_NUM\_POKEMON | Set to the highest Pokemon index + 1. |
| MAX\_LEVEL | The highest possible level for a Pokemon. If you change this value, make sure the also modify the equivalent value found in “[special\_inserts.asm](https://github.com/Skeli789/Complete-Fire-Red-Upgrade/blob/master/special_inserts.asm)”. |
| NUM\_TRAINER\_CLASSES | The number of trainer classes. Vanilla FR has 107. |
| EVOS\_PER\_MON | If you’ve changed the number of evolutions per Pokemon, update this number. |
| DUSK\_BALL\_MULTIPLIER | The catch rate for [Dusk Balls](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9_Ball#Dusk_Ball). |
| STANDARD\_IV | The number of IVs for each stat that standard Trainer’s Pokemon are generated with. |
| SWARM\_CHANCE | The chance in percent that a [swarm](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_outbreak) Pokemon will be generated on a route if there is currently a swarm in progress on that route. |
| WILD\_DOUBLE\_RANDOM\_CHANCE | The chance that a wild double will be initiated if the player is walking in grass with a background byte with its 4th bit set. For instance, grass with a background byte of 0x5 will have a chance of starting a wild double battle, and grass with a background byte of 0x25 will have a chance of starting a wild double battle and be covered by the player (water is similarly 0x6 and 0x26). |

**Badge Obedience**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| BASE\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have no badges. |
| BADGE\_1\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 1 badge. |
| BADGE\_2\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 2 badges. |
| BADGE\_3\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 3 badges. |
| BADGE\_4\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 4 badges. |
| BADGE\_5\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 5 badges. |
| BADGE\_6\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 6 badges. |
| BADGE\_7\_OBEDIENCE\_LEVEL | The highest level that a traded Pokemon will obey the player at if they have 7 badges. |

**Memory Locations**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| SEEN\_DEX\_FLAGS | The memory location of the *Seen* Pokedex flags. If you have not expanded the Pokedex, this should be set to:  gSaveBlock2Ptr->pokedex.seen |
| CAUGHT\_DEX\_FLAGS | The memory location of the *Caught* Pokedex flags. If you have not expanded the Pokedex, this should be set to:  gSaveBlock2Ptr->pokedex.owned |

Putting a // before any of the below features will remove them from the engine.

**Misc Features**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| INVERSE\_BATTLES | Enables the possibility of having an [Inverse Battle](https://bulbapedia.bulbagarden.net/wiki/Inverse_Battle) if the INVERSE\_FLAG is set. |
| TIME\_ENABLED | Various features are updated to account for the time of day. Certain palettes are also dynamically faded depending on the time of day. Removing this feature will always result in *Daytime*. |
| DNS\_IN\_BATTLE | If TIME\_ENABLED is defined, then certain background palettes will be faded dynamically in battle depending on the time of day. These values can be set in **src/DNS.c**. |
| OVERWRITE\_RIVAL | Loads the buffered rival’s name for trainer classes 0x51, 0x59, and 0x5A. |
| TRAINER\_CLASS\_POKE\_BALLS | Creates trainer Pokemon with custom Poke Balls based on trainer class determined by what is set in **src/Tables/** **Class\_Based\_Poke\_Ball\_Table.c**. |
| TRAINERS\_WITH\_EVS | Trainer Pokemon with a custom moveset, custom item, an AI value greater than 1, and an IV value (labeled EVs in most trainer editors) greater than 0 will have custom stats loaded from the spread number indicated by the IV value in **src/Tables/Trainers\_With\_EVs\_Table.c**. For example, setting the IV value to 1 will loaded the spread labeled “1” in the *TrainersWithEvsTable*.  Modifying the required conditions to activate this feature can be done by searching for the line *#ifdef TRAINERS\_WITH\_EVS* in **src/build\_pokemon.c**. |
| CONTINUE\_LOST\_BATTLES | If TUTORIAL\_BATTLES is defined, then if Var 0x8000 is set to 0xFEFE before a battle begins, *trainerbattle 0x9* can be used to continue a lost battle. The result of the battle will then be stored in Var 0x800D (LastResult).  If TUTORIAL\_BATTLES is not defined, then *trainerbattle 0x9* will always allow a lost battle to be continued. |
| REALLY\_SMART\_AI | The AI knows everything about the Pokemon on the player’s side (such as which moves it has, its ability, item, etc.) without having seen it first. This does not allow the AI to know what move the player will use before they use it. |
| DISPLAY\_REAL\_MOVE\_TYPE\_ON\_MENU | When choosing a move, the move type the move will become will be displayed (such as Hidden Power, Weather Ball in Weather, etc.). |
| DISPLAY\_REAL\_ACCURACY\_ON\_MENU | When pressing the *L*-button while choosing a move, the "true" move accuracy will be displayed. For example, the move *Psychic* used by a Pokemon with *Compound Eyes* will have its accuracy appear as 130. |
| DISPLAY\_REAL\_POWER\_ON\_MENU | When pressing the *L*-button while choosing a move, the "true" move power will be displayed. For example, moves *like Fury Cutter* will show their correct power. |
| CUSTOM\_BATTLE\_BACKGROUNDS | Setting BATTLE\_BG\_VAR will allow custom backgrounds to be loaded in battles. |
| OVERWRITE\_BG\_FOR\_LEADER\_CHAMPION | Special Battle Background palettes will be loaded in for battles against Gym Leaders and the Champion, using the regular indoor background graphics as a base. |
| BRIDGE\_FIX | The water battle background will only be loaded in battle if the player's surfing sprite is shown. This means that if the player is walking on water, the battle background loaded will be incorrect. |
| MEGA\_EVOLUTION\_FEATURE | Mega Evolutions can be used. |
| TUTORIAL\_BATTLES | Professor Oak’s tutorial will be activated for *trainerbattle 0x9*. |
| TANOBY\_RUINS\_ENABLED | Causes [Unown](https://bulbapedia.bulbagarden.net/wiki/Unown_(Pok%C3%A9mon)) to be spawned in maps using the [Tanoby Ruins](https://bulbapedia.bulbagarden.net/wiki/Tanoby_Chambers) map names according to the current chamber. |
| ALTERING\_CAVE\_ENABLED | If the current map is the [Altering Cave](https://bulbapedia.bulbagarden.net/wiki/Altering_Cave) and Var 0x4024 is set, Wild Pokemon will spawn. |
| SWEET\_SCENT\_ONLY\_IN\_CLEAR\_WEATHER | In certain generations, [Sweet Scent](https://bulbapedia.bulbagarden.net/wiki/Sweet_Scent_(move)) only spawns wild Pokemon in the Overworld if the weather is clear. |
| OBEDIENCE\_BY\_BADGE\_AMOUNT | Pokemon obedience is determined by the number of badges the Player has rather than by which badges the player has. The other badge defines in this case act as “number of badges acquired” instead of “acquired badge X”. |
| *SELECT\_FROM\_PC* | *If uncommented, allow the player to select and manipulate data of pokemon from the PC storage boxes. See* [*PC Selection*](#_Select_from_PC) *for more details.* |
| EXISTING\_OW\_ADDRESS | Uncomment and define the address of your existing overworld sprite tables, if it exists. For reference, the table is at 0x81a2000 in JPANs engine. If uncommented, a new table will be generated, the offset of which will be defined in *offsets.ini* |
| SET\_HEALING\_PLACE\_HACK | If uncommented, the whiteout hack from JPANs engine is implemented, allowing *VAR\_HEALINGMAP*, *VAR\_HEALING\_XPOS*, & *VAR\_HEALING\_YPOS* to be utilized. |
| FOSSIL\_IMAGE\_HACK/ EXISTING\_FOSSIL\_IMAGE\_TABLE\_ADDRESS | If *FOSSIL\_IMAGE\_HACK* is uncommented, a table is generated allowing *Var 0x8004* to load an image from said table. *EXISTING\_FOSSIL\_IMAGE\_TABLE\_ADDRESS* then allows you to define a table address, if you’ve already inserted JPANs engine. The table in his engine was 0x81a4600, for reference. See [Special 0x18B](#_Other_Specials) for table details |

**Misc Battle Effect Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_BURN\_DAMAGE | [Burn](https://bulbapedia.bulbagarden.net/wiki/Burn_(status_condition)) damage takes 1/8 of max health instead of 1/16. |
| OLD\_PARALYSIS\_SPD\_DROP | [Paralysis](https://bulbapedia.bulbagarden.net/wiki/Paralysis_(status_condition)) lower Speeds down to 1/4 instead of ½. |
| OLD\_CONFUSION\_CHANCE | [Confusion](https://bulbapedia.bulbagarden.net/wiki/Status_condition#Confusion) stops attacks 50% of the time instead of 33%. |
| INFINITE\_WEATHER | [Weather abilities](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_with_effects_on_weather_conditions) make weather last for infinite turns. |
| INFINITE\_TERRAIN | [Terrain abilities](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_with_effects_on_terrain) make terrain last for infinite turns. |
| NO\_SHEER\_COLD\_NERF | Remove all Gen 7 [Sheer Cold](https://bulbapedia.bulbagarden.net/wiki/Sheer_Cold_(move)) nerfs. |
| OLD\_MOVE\_SPLIT | The [Physical/Special](https://bulbapedia.bulbagarden.net/wiki/Damage_category#Physical.2FSpecial_split) split is based on move types.  Status moves are still set with the split byte, however. |

**Ability Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_GALE\_WINGS | [Gale Wings](https://bulbapedia.bulbagarden.net/wiki/Gale_Wings_(Ability)) activates regardless of the user's HP. |
| OLD\_PRANKSTER | [Prankster](https://bulbapedia.bulbagarden.net/wiki/Prankster_(Ability)) won’t fail against Dark-Types. |

**Damage Calculation Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_CRIT\_DAMAGE | [Critical hits](https://bulbapedia.bulbagarden.net/wiki/Critical_hit) to do 2x damage; 3x with [Sniper](https://bulbapedia.bulbagarden.net/wiki/Sniper_(Ability)). |
| CRIT\_CHANCE\_GEN\_6 | Uses the Gen 6 crit chance. |
| CRIT\_CHANCE\_GEN\_2\_TO\_5 | Uses the Gen 2-5 crit chance. |
| BADGE\_BOOSTS | Having badges gives the Player’s Pokemon [stat boosts](https://bulbapedia.bulbagarden.net/wiki/Badge#Stat_boost). |
| OLD\_ATE\_BOOST | “[Ate](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_that_can_modify_move_types)” abilities give a 1.3x boost instead of 1.2x. |
| OLD\_GEM\_BOOST | [Gems](https://bulbapedia.bulbagarden.net/wiki/Gem) give a 1.5x boost instead of 1.2x. |
| OLD\_EXPLOSION\_BOOST | [Exploding](https://bulbapedia.bulbagarden.net/wiki/Explosion_(move)) moves halve the target’s Defense. |
| OLD\_HIDDEN\_POWER\_BP | [Hidden Power](https://bulbapedia.bulbagarden.net/wiki/Hidden_Power_(move)) has its Base Power calculated from the attacker’s IVs. |
| PORTAL\_POWER | Enables Hoopa-Unbound’s signature ability in Pokemon Unbound, Portal Power. This reduces the power of non-contact moves by 25%. |
| OLD\_SOUL\_DEW\_EFFECT | [Soul Dew](https://bulbapedia.bulbagarden.net/wiki/Soul_Dew) doubles Latios & Latias' Sp. Atk & Sp. Def. |
| OLD\_PARENTAL\_BOND\_DAMAGE | The second hit of [Parental Bond](https://bulbapedia.bulbagarden.net/wiki/Parental_Bond_(Ability)#Generation_VI) does 50% of the original damage instead of 25%. |

**Capturing Pokemon Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| CATCH\_TRAINERS\_POKEMON | Allows the possibility to capture trainer Pokemon by setting the CATCH\_TRAINERS\_POKEMON\_FLAG. |
| NO\_HARDER\_WILD\_DOUBLES | In Gen 5, Pokemon encountered in wild double battles were [harder to catch](https://bulbapedia.bulbagarden.net/wiki/Catch_rate#Capture_method_.28Generation_V.29) (based on how many species are owned). This feature implements that catch rate decrement. |
| CRITICAL\_CAPTURE | Allows for [Critical Capture](https://bulbapedia.bulbagarden.net/wiki/Catch_rate#Critical_capture) to occur. The odds at which this occurs can be found in the function  *bool8 CriticalCapture(u32 odds)* found in **src/catching.c**. |

**Exp. Gain Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_EXP\_SHARE | The [Exp. Share](https://bulbapedia.bulbagarden.net/wiki/Exp._Share) acts like it did before Gen 6. |
| TRAINER\_EXP\_BOOST | Gives an Exp boost for defeating a trainer’s Pokemon. (Pre Gen 7) |
| OLD\_EXP\_SPLIT | Exp. is split amongst all participating Pokemon. (Pre Gen 6) |
| FLAT\_EXP\_FORMULA | Use a Flat Exp. calculation formula. (Gens 2- 4, 6) |
| GEN\_7\_BASE\_EXP\_YIELD | Base Exp is retrieved from the table *gBaseExpBySpecies* found in the file **src/Tables/Experience\_Tables.c**, instead of being loaded from the Pokemon’s base stats. This is done to account for larger Exp. values that started in Gen 5. The table is pre-set to match [Gen 7 Exp. values](https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_effort_value_yield). |

**Other Options**

|  |  |
| --- | --- |
| *Definition* | *Description* |
| ACTIVATE\_DOUBLE\_BATTLE\_FROM\_FLAG | Allow DOUBLE\_BATTLE\_FLAG to be set to active double battles for trainers if possible. |
| NO\_GHOST\_BATTLES | Disables the [Ghost battle](https://bulbapedia.bulbagarden.net/wiki/Ghost_(literal)) feature from Pokemon Tower in Lavender town. |
| GEN\_4\_PLAYER\_RUNNING\_FIX | Increase the lag between frames as the player OW runs, to simulate a more accurate Gen 4 running effect. |
| GEN4\_PLUS\_SELECTION\_SCREEN | This does not give the [Gen 4+ selection screen](https://www.pokecommunity.com/showthread.php?t=414022), it only adds features that supports it. |
| OBEDIENCE\_CHECK\_FOR\_PLAYER\_ORIGINAL\_POKEMON | Opens up the possibility that the Player's Pokemon can [disobey](https://bulbapedia.bulbagarden.net/wiki/Obedience) them, as opposed to just traded Pokemon. |
| WILD\_ALWAYS\_SMART | All wild Pokemon use AI features meant for trainers. |
| HAIL\_IN\_BATTLE | Enables the [Hail](https://bulbapedia.bulbagarden.net/wiki/Hail_(weather_condition)) weather effect in battle when the OW weather is set to WEATHER\_STEADY\_SNOW (0x7). |
| FOG\_IN\_BATTLE | Enables the [Fog](https://bulbapedia.bulbagarden.net/wiki/Fog) weather effect in battle. Do not enable this feature without first enabling one of the fog features below! |
| FOG\_IN\_BATTLE\_1 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_1 (0x6). |
| FOG\_IN\_BATTLE\_2 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_2 (0x9). |
| FOG\_IN\_BATTLE\_3 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_3 (0xA). |
| HIDE\_HEALTHBOXES\_DURING\_ANIMS | Hides the healthboxes (battle bars, etc.) during move animations, and some special animations (like Mega Evolution). This is done in Gen 4+ |
| DONT\_HIDE\_HEALTHBOXES\_ATTACKER\_STATUS\_MOVES | If HIDE\_HEALTHBOXES\_DURING\_ANIMS is defined, when the attacker is using a move that only targets itself, the healthboxes will not be hidden. |
| ENCOUNTER\_MUSIC\_BY\_CLASS | The music played when a trainer spots the player in the overworld is determined by the trainer class, rather than the music Id set in the trainer data. The song options are lists in **src/Tables/Music\_Tables.c** and can be modified by changing the values in *gClassBasedTrainerEncounterBGM*. Any class not defined in the array will be automatically set to BGM\_EYE\_BOY. |

## Other Necessary Features Setup

### Mega Evolution / Primal Reversion / Ultra Burst

Before setting up Mega Evolution, two things must be done. First, make sure EVOS\_PER\_MON in the config file is set to the correct number (it should be the same as the number + 1 at 0x43116 in your rom). Second, your Pokemon Editor of choice will need to be modified:

**G3T**:

In your Gen3Tools folder, open up **Customisation/** **Pokemon Editor.ini**, and add the line **FE=Mega Evolution** to the bottom of the file.



**D&D**:

Has Mega Evolution pre-installed, however it cannot set up Wish-based Mega Evolution correctly.

**G3HS**:

1. Open up the file **PokeRoms.ini**, find your rom code.
2. Modify “evolutionmethods” such that the 254th evolution method is set to *Mega Evolution*:



1. Modify “evomethodsproperties” such that the 254th method is set to *Item*.

[Mega Evolution](https://bulbapedia.bulbagarden.net/wiki/Mega_Evolution)

Mega Evolution set up is similar to the how the [previous](https://www.pokecommunity.com/showthread.php?t=348182) Mega Evolution system by Touched was set up. If a Pokemon is able to Mega Evolve, Mega Evolution can be trigged by pressing start on the move menu once the mega trigger appears.

1. Create a Mega Stone in G3T with the following format:



1. Create a Mega Ring key item. Its format is the same as any other key item.
2. Update the definition for the Mega Ring item in **include/constants/items.h**.
3. Open the file **src/mega.c** and search for KeystoneTable. Add your mega ring item to this table:



\*\*NOTE: The item does not need to be called *Mega Ring*. Any item added to this table can act as a mega ring.

1. If DEBUG\_MEGA is defined, then Mega Evolution can be used from the start of the game without the requirement of having the Mega Ring in the bag. Otherwise, the player must have a Mega Ring item in the bag to use Mega Evolution (Mega Evolution can always be used in Link Battles or the Battle Tower. If you want a trainer to be able to use Mega Evolution, add a mega ring item to their battle items:



Don’t forget to give the trainer’s Pokemon the relevant Mega Stone to hold!

1. Find the Pokemon in G3T that you wish to bestow the ability of Mega Evolution upon. Set up the evolution data with following template:



In the above image:

* “Nothing required” is set to the mega stone item index created earlier.
* “Evolve to” is set to a new species representing the mega form (you’ll need to expand Pokemon or replace an existing one). Make sure this mega form has the same gender rate and Exp rate as the non-mega form or things will get messed up.

*Wish Mega Evolution* (for Rayquaza) should be set up using the following template:

In the above image:

* “Nothing required” should be set to the move id for *Dragon Ascent* (0x22F / 559). G3T has issues with setting numbers past 0xFF, so you’ll need to hex edit or use a different Pokemon editor to set the proper value for *Dragon Ascent*.
* “Evolve to” is set to a new species representing Mega Rayquaza (you’ll need to expand Pokemon or replace an existing one). Make sure Mega Rayquaza has the same gender rate and Exp rate as Rayquaza or things will get messed up.
* The first of “Unknown Bytes” should be set to 2.

1. Set the Mega form’s evolution data like the following template:



Notice that the item is left blank. Also notice that *Mega Rayquaza* still has its first unknown byte set to 2. Do not forget this!

1. Now just give the player a mega ring item and a Pokemon with the appropriate mega stone and they will be able to Mega Evolve!

Notes about Mega Evolution:

* Only a single Mega Evolution may be used by each side during any given battle.
* In multi battles, both trainers can Mega Evolve a single Pokemon, effectively allowing for two Mega Evolutions per side during any given battle.
* Mega Evolution is not prevented if any Pokemon on the side Ultra Bursted or underwent Primal Reversion.
* If Rayquaza uses a Z-Move, it’ll be unable to Mega Evolve.

[Primal Reversion](https://bulbapedia.bulbagarden.net/wiki/Primal_Reversion)

Primal Reversion does not rely on a mega ring to activate and will thus always activate if Kyogre or Groudon hold the appropriate item. Much less setup is required for Primal Reversion.

1. Find the Red Orb and Blue Orb in G3T’s item editor and modify them to match the following:



The item effect should be set as 93 and the item quality should be set as 0 for the Red orb and 1 for the Blue Orb. These numbers determine whether the Primal Reversion is *Red* (0) or *Blue* (1).

1. Modify Kyogre and Groudon’s evolution data in G3T to match the following:



In the above images:

* “Nothing required” is set to the *Blue Orb* item index for Kyogre and to the *Red Orb* item index for Groudon. Note again that G3T does not represent these item indices correctly so you may need to use another editor.
* “Evolve to” is set to a new species representing the primal form (you’ll need to expand Pokemon or replace an existing one). Make sure this primal form has the same gender rate and Exp rate as the non-primal form or things will get messed up.
* The first of the “Unknown Bytes” is set to 1 to represent Primal Reversion.

1. Set up the primal forms’ evolution data in G3T to match the following:



Notice that the items are left blank. Also notice that both Primal Pokemon still have their first unknown bytes set to 1. Do not forget this!

Note about Primal Reversion:

* The alpha and omega symbols on the health bar are generated based on which species is in its Primal form. By default, the alpha symbol is set to appear if the species is Primal Kyogre, and the omega symbol is set to appear if the species is Primal Groudon. To change this requirement, modify the following functions in **src/mega.c**:



These can easily be modified by adding species to compare to. So if I wanted the omega symbol to appear for Primal Dialga, I would make the following modification:



Don’t forget to also define PKMN\_PRIMAL\_DIALGA in **include/constants/species.h**.

[Ultra Burst](https://bulbapedia.bulbagarden.net/wiki/Ultra_Burst)

Primal Reversion does not rely on a mega ring to activate and will thus always activate if Necrozma holds the appropriate item. Much less setup is required for Ultra Burst.

1. Create an *Ultranecrozium Z* item in G3T with the following format:



1. Modify **both** Necrozma forms’ evolution data in G3T to match the following:



In the above images:

* “Nothing required” is set to the *Ultranecrozium Z* item index. Note again that G3T does not represent these item indices correctly so you may need to use another editor.
* “Evolve to” is set to a new species representing *Ultra Necrozma* (you’ll need to expand Pokemon or replace an existing one). Make sure all *Necrozma* forms have the same gender rate and Exp rate or things will get messed up.
* The first of the “Unknown Bytes” is set to 3 to represent Ultra Burst.

1. Set up the *Ultra Necrozma’s* evolution data in G3T to match the following:



In the above image:

* The items are left blank.
* The first unknown byte is set to 1. Do not forget this!

Note about Ultra Burst:

* Contrary to what is shown in the above image, *Ultra Necrozma* does **not** need reversion data from both *Necrozma* fusion forms. *Ultra Necrozma* will always revert to the form is *Ultra Bursted* from at the end of the battle, regardless of which species is written in its evolution data. If the *Ultra Necrozma* was encountered in the wild, it will revert to the first species in its evolution list by default.
* *Ultra Burst* is trigged the same way as *Mega Evolution* on the move menu.
* As *Ultra Burst* is not considered *Mega Evolution*, *Ultra Necrozma* can still use Z-Moves if it knows the appropriate base move (*Photon Geyser* by default).

### Z-Moves

//TODO

### Trainer Sliding Messages

//TODO

### Trainer Backsprites

//TODO

### Battle Frontier

//TODO

### Battle Music

//TODO

### Battle Terrain

//TODO

### Poke Balls

//TODO

### Pickup

//TODO

### Time of Day Based Wild Encounters

In order to use this feature, TIME\_ENABLED must be defined. Once it is open the file **src/Tables/Wild\_Encounter\_Tables.c**. Find where it says //Modify this section. This is the data that will be modified.

**Example: Modifying The Night Data**

For this example, HOOTHOOT will be added onto ROUTE 1 in place of PIDGEY.

First, the wild data needs to be created. If you open up Advance Map to the wild data for ROUTE 1, you should see this:



Now that you know what the data looks like, convert it into a C structure (see image below) and paste it at the top file, directly under the line “#ifndef UNBOUND //Modify this section”, or under previously created wild data structures. No matter what, it must be placed above the line

“const struct WildPokemonHeader gWildMonMorningHeaders[] =”:



^ This line is very important and must be added in as well. Make sure the label correctly matches the name for your newly created wild data structure. The *21* is the encounter rate.

Now that the wild data has been copied from Advance Map, it’s time to make modifications.

Change all the “PKMN\_PIDGEY” to “PKMN\_HOOTHOOT”:



The data for Route 1 at night is now complete. As this is data for night time, we need to add it to our night table, “*gWildMonNightHeaders*”. Make sure you leave the pre-existing entry at the bottom of the table:



The only thing left to do now is define “*MAP\_ROUTE\_1*”. Go back to Advance Map and find the map bank and map number for ROUTE 1 (the map bank is 3, and the map number is 19). At the top of the file, add a line formatted like #define MAP\_NAME ((MAP\_BANK << 8) | MAP\_NUM)



**Make sure the map name matches what’s in the brackets for “.mapGroup” and “.mapNum” (ie. ROUTE\_1 became MAP\_ROUTE\_1).**

Now wild night data has successfully been added for Route 1. Morning and Evening data follow the same pattern. Any route that doesn’t have morning or night data defined will load the standard day data set in Advance Map.

For water, fishing, or Rock Smash data, follow the same steps, but look [here](https://github.com/pret/pokeemerald/blob/c74bfbe11017e4f937dc6466a93a6d3dc72cdac7/src/data/wild_encounters.h) to see how to structure those kinds of wild datasets.

If you followed everything correctly, here is what the file should look like now:  




# Other Features Included

### Save Expansion

//TODO

### Updated Flutes

The [Black Flute](https://bulbapedia.bulbagarden.net/wiki/Black_Flute) and [White Flute](https://bulbapedia.bulbagarden.net/wiki/White_Flute) have been updated to the standards from ORAS.

### Trainer Face Fix

The player will face trainers before battle.

### Extra Pedometers

From JPANs engine, this includes extra pedometers to allow for various step-related scripts and actions. There is a 32bit (4 byte) pedometer that is always active, as well as 1 32bit, 1 16bit, and 2 8bit pedometers that can be activated/deactivated with flags. These flags can be viewed above starting with *FLAG\_LONG\_PEDOMETER*.

### Select from PC Hack

If SELECT\_FROM\_PC is defined, you can select pokemon directly from the PC to modify certain data. To initiate, set *pcSelect\_StateTracker* (defined in *asm\_defines.s*) to 1 before using Special 0x3C. The hack will store the box number and slot to *Var 0x8000* and *0x8001*, respectively. All data retrieval/manipulation specials will be able to access the selected PC pokemon by setting *Var 0x8003* to 1. Here’s an example of nicknaming a pokemon in the PC:

writebytetooffset 0x1 pcSelect\_StateTracker

special 0x3c

waitstate

writebytetooffset 0x0 pcSelect\_StateTracker

compare LASTRESULT 0x7F

if 0x1 goto @nothingSelected // user cancelled out of the PC menu

bufferpokemon 0x0 0x8002

setvar 0x8003 0x1 // data source is in the PC Box

special 0x7d

compare LASTRESULT 0x1

if 0x1 goto @traded

special 0x9e

waitstate

### Dex Nav

A simplified [Dex Nav](https://bulbapedia.bulbagarden.net/wiki/DexNav) system for Fire Red. It replaces the *PokeDex* option in the Start Menu with *TOOLS*, which contains *PokeDex* and *DexNav*. Selecting the DexNav opens the graphical user interface (GUI) to allow the player to view what pokemon they have caught/seen on the current map, and either press *Select* or *A* on a given pokemon to search for it on the map. *Select* will save the pokemon information to VAR\_DEXNAV and allow the player to search for that pokemon via *Select* from the overworld. [Credit to FBI for the source code.](https://github.com/EternalCode/Dexnav)

### Dynamic Overworld Palettes

[Navenatox’s dynamic overworld palettes](https://github.com/Navenatox/DynamicOverworldPalettes) has been included in this engine for convenience and wonderful integration with the character customization and freedom of overworld sprite palettes.

### Script Specials

Several new scripting specials have been added to the engine. Many have been ported from JPAN’s hacked engine, and thus will work similar to how they worked there.

If a special is shown to have a var (such as *Var 0x8000*) as an input, set that var to the required data. If SELECT\_FROM\_PC is enabled, remember to keep track the pokemon source from *Var 0x8003* before calling specials that manipulate pokemon attributes.

If a special is shown to have a return value, it must be called with the **special2** scripting command.

#### Pokemon Specials

The following specials check or change pokemon attributes. If SELECT\_FROM\_PC is enabled, *Var 0x8003* will allow you to check/change data from PC boxed pokemon if it is set to 1. Otherwise it will check/change from a party pokemon.

Special 0x7 – EV/Contest Stat Checker

**Details**: Checks a party/boxed Pokemon’s EVs or Contest stats

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Stat to check

|  |  |
| --- | --- |
| HP EV | 0x0 |
| Attack EV | 0x1 |
| Defense EV | 0x2 |
| Speed EV | 0x3 |
| Special Attack EV | 0x4 |
| Special Defense EV | 0x5 |
| Coolness | 0x6 |
| Beauty | 0x7 |
| Cuteness | 0x8 |
| Smartness | 0x9 |
| Toughness | 0xA |
| Luster | 0xB |

**Returns**: Stat value to given variable

**Example Script:**

Setvar 0x8003 0x0 // select from party

Setvar 0x8004 0x3 // 4th pokemon in party

Special2 0x800D 0x7

Buffernumber 0x0 0x800D // buffer EV stat into [buffer1]

Special 0x8 – Pokemon IV Checker

**Details**: Checks a party/boxed Pokemon’s IVs

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* IV stat to check

|  |  |
| --- | --- |
| HP IV | 0x0 |
| Attack IV | 0x1 |
| Defense IV | 0x2 |
| Speed IV | 0x3 |
| Special Attack IV | 0x4 |
| Special Defense IV | 0x5 |

**Returns**: IV stat value to given variable

**Example Script (with PC Selection Hack):**

Writebytetooffset 0x1 0x0203b7ac // or whatever pcSelect\_StateTracker is set to

Special 0x3C // select boxed mon, box stored to var8000, slot to var8001

Waitstate

Compare 0x800D 0x7F // player exited without selecting

If 0x1 goto @didNotSelect

Setvar 0x8003 0x1 // from boxed mon

Setvar 0x8005 0x1 // check attack IV

Special2 0x800D 0x8

Buffernumber 0x0 0x800D // buffer attack IV to [buffer1]

Special 0x9 – Pokemon Ribbon Checker

**Details**: Checks a pokemon’s ribbons. The first 5 ribbons have values between 1-4 for Normal, Super, Hyper, and Master rank, so be sure to reference the correct bit value(s). See [this page](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_data_substructures_in_Generation_III#Ribbons_and_Obedience) for more info.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Ribbon bit to check. Here are the possible bits and known ribbon values

|  |  |  |
| --- | --- | --- |
| **Ribbon** | **Bit(s)/Rank (Var8005 Val)** | **Hex Value** |
| Cool Ribbons | Normal (1) = bit 1 (0001)  Super (2) = bit 2 (0010)  Hyper (3) = bits 1,2 (0011)  Master (4) = bit 3 (0100) | 0x1  0x2  0x3  0x4 |
| Beauty Ribbons | Normal (1) = bit 4  Super (2) = bit 5  Hyper (3) = bit 4,5  Master (4) = bit 6 | 0x8  0x10  0x18  0x20 |
| Cute Ribbons | Normal (1) = bit 7  Super (2) = bit 8  Hyper (3) = bit 7,8  Master (4) = bit 9 | 0x40  0x80  0xC0  0x100 |
| Smart Ribbons | Normal (1) = bit 10  Super (2) = bit 11  Hyper (3) = bit 10,11  Master (4) = bit 12 | 0x200  0x400  0x600  0x800 |
| Tough Ribbons | Normal (1) = bit 13  Super (2) = bit 14  Hyper (3) = bit 13,14  Master (4) = bit 15 | 0x1000  0x2000  0x3000  0x4000 |
| Champion | Bit 16 | 0x8000 |
| Winning | Bit 17 | 0x10000 |
| Victory | Bit 18 | 0x20000 |
| Artist | Bit 19 | 0x40000 |
| Effort | Bit 20 | 0x80000 |
| Special 1 | Bit 21 | 0x100000 |
| Special 2 | Bit 22 | 0x200000 |
| Special 3 | Bit 23 | 0x400000 |
| Special 4 | Bit 24 | 0x800000 |
| ?? | Bit 25 | 0x1000000 |
| Special 5 | Bit 26 | 0x2000000 |
| Special 6 | Bit 27 | 0x4000000 |
| ?? | Bits 28-30 | 0x8000000 to 0x40000000 |
| Obedience (Mew/Deoxys) | Bit 31 | 0x80000000 |

NOTE: the “hyper” status is a bit more challenging to check for, as the input is a bit number and these ribbon statuses require checking two bits. If this is something you are interested in implementing, you would need to remove the Normal bit upon receiving Super status, and then check for both bits with two separate special calls to determine the Hyper status.

**Returns:** 1 if the ribbon flag is set, 0 if not

**Example Script:**

Setvar 0x8003 0x0 // from party

Setvar 0x8004 0x2 //3rd mon in party

Setvar 0x8005 16 // check hall of fame flag

Special2 0x800D 0x9

Buffernumber 0x0 0x800D

Compare 0x800D 0x1

If 0x1 goto @isAChampion

// else, pokemon was not in the hall of fame

Special 0xA – Pokerus Timer Checker

**Details**: Checks the pokerus virus timer on a pokemon

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Pokerus time left to given variable

**Example Script:**

Setvar 0x8003 0x0 // from party

Setvar 0x8004 0x0 // first party pokemon

Special2 0x800D 0xA

Buffernumber 0x0 0x800D // pokerus timer into [buffer1]

Bufferpartypokemon 0x1 0x0 // buffer first poke name into [buffer2]

Msgbox @timeLeft 0x6

#org @timeLeft

= [buffer2] is sick for [buffer1] more cycles!

Special 0xB – Pokeball Checker

**Details**: Check the Pokeball ID of a pokemon. Note that the ball ID matches the item ID only for the in-game pokeballs. If you’ve expanded the pokeballs, the ball ID will not match and you’ll need more logic.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Pokeball ID to given variable

**Example Script:**

Writebytetooffset 0x1 0x0203b7ac // select from PC hack

Special 0x3C //store box/slot into vars 0x8000, 0x8001, respectively

Waitstate

Compare 0x800D 0x7F

If 0x1 goto @didNotSelect // player cancelled without selection

Setvar 0x8003 0x1 //from box

Special2 0x800D 0xB

//call @idCheck // needed to match ball ID to item ID if you’ve expanded pokeballs

Buffernumber 0x1 0x800D // buffer item number to [buffer2]

Special 0x7C //buffer boxed mon nickname to [buffer1]

Msgbox @ball 0x6

#org @ball

= [buffer1] is inside a [buffer2]! How fortunate!

Special 0xC – Check Capture Location

**Details**:

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Capture Location ID to given variable.

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x5 // last party mon

Special2 0x800D 0xC

Buffernumber 0x0 0x800D // buffer capture location to [buffer1]

Special 0xD – Happiness Checker

**Details**: Check the number of happiness points for a pokemon.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Happiness value (0-255) to given variable.

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x2 //3rd pokemon

Special2 0x800D 0xD

Buffernumber 0x0 0x800D //buffer happiness to [buffer1]

Compare 0x800D 255

If 0x1 goto @maxedHappiness

//etc

Special 0xE – Hold Item Checker

**Details**: Check Hold Item value of a Pokemon

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Hold Item ID

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x3 //4th pokemon

Special2 0x800D 0xE

Buffernumber 0x0 0x800D //buffer to [buffer1]

Special 0xF – Add/Subtract to EVs

**Details**: Add or subtract values to pokemon EVs (between 0 and 252)

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* stat for math (see Special 0x7 for indices)

*Var 0x8006:* Value to add. 0x01YY to subtract YY, 0x00ZZ to add ZZ

**Returns:** nothing

**Example Script:**

Setvar 0x8003 0x0 // from party

Setvar 0x8004 0x1 //2nd party pokemon

Setvar 0x8005 0x3 // Speed EV

Setvar 0x8006 0x0150 // subtracting 0x50, or 80 speed EVs

Special 0xF

Special 0x10 – Set IVs

**Details**: Set IV values for a pokemon. No math here, just setting to a specific value.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* IV stat to change (see Special 0x8 for indices), between 0x0 and 0x1F

*Var 0x8006:* IV value to set

**Returns:** nothing

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x0 //first pokemon

Setvar 0x8005 0x0 //hp IV

Setvar 0x8006 0x1F //value to set

Special 0x10 // maximize first party pokemon’s HP IV

Special 0x11 – Set Ribbons

**Details**: Set or clear a pokemon’s ribbon flag

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* ribbon flag to set (see Special 0x9 for indices/values)

0x00XX will set a ribbon, 0x01YY will clear a ribbon

**Returns:** nothing

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x2 //3rd pokemon

Setvar 0x8005 0x3 //3rd ribbon bit = Master Rank on Cool Ribbon

Special 0x11 // set the ribbon

Special 0x12 – Set Pokerus

**Details**: Set a pokerus timer of a Pokemon.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Number of cycles, 0x0 to 0xF, 0x10 to “cure”

**Returns:** nothing

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x0 //first mon

Setvar 0x8005 0x10 //cure pokemon’s pokerus

Special 0x12

Special 0x13 – Change Happiness

**Details**: Add or subtract to a pokemon’s happiness

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Amount to add/subtract.

0x01YY will subtract YY from happiness, 0x00XX will add XX to happiness

**Returns:** nothing

Special 0x14 – Change Pokeball

**Details**: Set the ball ID of a pokemon.

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Ball ID to set

**Returns:** nothing

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x0 //first pokemon

Setvar 0x8005 0x1 //set to master ball

Special 0x14

Special 0x15 – Change Hold Item

**Details**:

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Item ID to set

Note that if the pokemon is already holding an item, this will not change the item, unless *Var 0x8005* is set to 0 to remove the item. Then you can call it again to set a new hold item.

**Returns:** 0 to Var800D if successful item change, 1 if not

**Example Script:**

Setvar 0x8003 0x0 //from Party

Setvar 0x8004 0x5 //last Mon

Setvar 0x8005 0x0 //remove a hold item first

Special 0x15

Setvar 0x8005 ITEM\_SILKSCARF //item to give

Special 0x15 // give silk scarf

Compare 0x800D 1

If 0x1 goto @failed

Special 0x16 – Change Species

**Details**: Change the species of a pokemon

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* Species to change to

**Returns:** nothing

Special 0x17 – Change Attacks

**Details**: Set or remove a move for a pokemon. This one does not work with the PC Selection Hack.

**Input**:

*Var 0x8004:* Pokemon Slot (0-5)

*Var 0x8005:* Move Slot (0-3 for moves 1-4, respectively)

*Var 0x8006:* Move ID (0 to clear move slot)

**Returns:** nothing

**Example Script:**

Special 0x9F //select a pokemon from the menu, store slot to Var 0x8004

Waitstate

Compare 0x800D 0x6

If 0x4 goto @cancelled

Setvar 0x8005 0 //first move

Setvar 0x8006 MOVE\_HYPERBEAM //teach hyper beam in slot 0

Special 0x17

Special 0x18 – Check Species

**Details**: Check the species of a pokemon

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** Species ID to given variable

**Example Script:**

Setvar 0x8003 0x0 //from party

Setvar 0x8004 0x0 //check first pokemon

Special2 0x800D 0x18

Compare 0x800D PKMN\_RATTATA //check if first pokemon if rattata

If 0x0 goto @notCorrect

Special 0x19 – Check Attack PP

**Details**: Check a Pokemon move’s PP

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

*Var 0x8005:* move slot (0-3)

**Returns:** PP left to given variable

**Example Script:**

Setvar 0x8003 0x0 //from Party

Setvar 0x8004 0x0 //first pokemon

Setvar 0x8005 0x0 //first move

Special2 0x800D 0x19

Buffernumber 0x0 0x800D

Compare 0x800D 0x0

If 0x1 goto @noPPLeft

#### Party Specials

Special 0x62 – Erase Pokemon

**Details**: Erase a pokemon from your party, or the entire party

**Input**:

*Var 0x8004:* Slot to erase (0xF for entire party)

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0xF

Special 0x62 //erase entire party

Special 0x63 – Status Checker

**Details**: Check the primary status of a pokemon

**Input**:

*Var 0x8004:* Pokemon Slot

**Returns:** status inflicted to given variable

|  |  |  |
| --- | --- | --- |
| **Status** | **Bits** | **Hex Value** |
| Sleep | 1  2  3 | 0x1  0x2  0x4 |
| Poison | 4 | 0x8 |
| Burn | 5 | 0x10 |
| Frozen | 6 | 0x20 |
| Paralyzed | 7 | 0x40 |
| Badly Poisoned | 8 | 0x80 |

**Example Script:**

Setvar 0x8004 0x0 //first party pokemon

Special2 0x8004 0x63 // get statuses to var 0x8004

Setvar 0x8005 0x80 //badly poisoned

Special2 0x800D 0x42 // Var 0x8004 & Var 0x8005

Compare 0x800D 0x1 // & will return 1 if the pokemon has this status

If 0x1 goto @badlyPoisoned

Special 0x64 – Status Inducer

**Details**: Inflict a primary status on a party pokemon

**Input**:

*Var 0x8004:* pokemon slot, or 0xF for entire party

*Var 0x8005:* status(es) to induce (see Special 0x63 for values)

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0xF //entire party

Setvar 0x8005 0x20 //freeze entire party

Special 0x64

Special 0x65 – Check Pokemon HP

**Details**: Check the amount of HP remaining for a party pokemon

**Input**:

*Var 0x8004:* pokemon slot

**Returns:** HP to given variable

**Example Script:**

Setvar 0x8004 0x0 //first pokemon

Special2 0x800D 0x65

Compare 0x800D 0x0

If 0x1 goto @deadPoke

Special 0x66 – Inflict Party Damage

**Details**: Inflict damage on/heal a pokemon, or entire party

**Input**:

*Var 0x8004:* pokemon slot, 0xF for entire party

*Var 0x8005:* damage to inflict/heal

*Var 0x8006:* 1 to heal, otherwise inflict damage

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0xF //entire party

Setvar 0x8005 20 //20 damage to party

Setvar 0x8006 0x0 //damage party

Special 0x66

#### Key Specials

Special 0x2B – Check AB Buttons

**Details**: Check if A or B has been pressed

**Input**: nothing

**Returns:** To the given variable:

0x0 if neither pressed

0x1 if A pressed

0x2 if B pressed

0x3 if both pressed

**Example Script:**

#org @loop

Special2 0x800D 0x2B

Compare 0x800D 0x1 // check for A

If 0x0 goto @loop // player cannot continue onwards until they press A

// continue

Special 0x2C – Check D Pad

**Details**: Check D pad presses

**Input**: nothing

**Returns:** to given variable

0x0 if no direction is pressed

0x1 if up is pressed

0x2 if left is pressed

0x3 if down is pressed

0x4 if right is pressed

0x5 if up-left is pressed

0x6 if up-right is pressed

0x7 if down-left is pressed

0x8 if down-right is pressed

**Example Script:**

#org @start

Special2 0x800D 0x2C

Compare 0x800D 0x3

If 0x1 goto @pressedDown

//etc

Special 0x2D – Check Start/Select

**Details**: Check if start/select are pressed

**Input**: nothing

**Returns:** to given variable:

0x0 if nothing pressed

0x1 if select pressed

0x2 if start pressed

0x3 if both pressed

**Example Script:**

Special2 0x800D 0x2d

Compare 0x800D 0x2

If 0x1 goto @pressedStart

Special 0x2E – Check L/R

**Details**: Check if L/R pressed

**Input**: nothing

**Returns:** to given variable

0x0 if nothing pressed

0x1 if R pressed

0x2 if L pressed

0x3 if both pressed

**Example Script:**

Special2 0x800D 0x2e

Compare 0x800D 0x1

If 0x1 goto @pressedR

Special 0x2F – Dump Keys

**Details**: Dump any and all keys that have been pressed

**Input**: nothing

**Returns:** Key presses to given variable

|  |  |  |
| --- | --- | --- |
| **Key** | **Bit** | **Hex** |
| A | 1 = 0001 | 0x1 |
| B | 2 = 0010 | 0x2 |
| Select | 3 = 0100 | 0x4 |
| Start | 4 = 1000 | 0x8 |
| Right | 5 = 0001 0000 | 0x10 |
| Left | 6 = 0010 0000 | 0x20 |
| Up | 7 = 0100 0000 | 0x40 |
| Down | 8 = 1000 0000 | 0x80 |
| R | 9 = 0001 0000 0000 | 0x100 |
| L | 10 = 0010 0000 0000 | 0x200 |

**Example Script:**

Special2 0x800D 0x2f

Compare 0x800D 0x1

If 0x1 goto @pressedA

//etc

Special 0xC9 – Force Key Input

**Details**: Force a key input from the user

Honestly, kind of useless, can just use applymovement and others

**Input**:

*Var 0x8004:* Key(s) to force (bitfield)

*Var 0x8005:* Number of times to press it

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x1 //A

Setvar 0x8005 0x2 // press twice

Special 0xC9 //force player to press A twice

Special 0xCA – Prevent Key Press

**Details**: Prevent player from being able to press button(s)

**Input**:

*Var 0x8004:* Key(s) to prevent (bitfield). 0 to allow all keys

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x104 //prevent *R* and *Select* from doing anything

Special 0xCA

Special 0xCB – Assign Key Script

**Details**: assign a specific script to a key

**Input**:

*Var 0x8004:* Key to assign script to (0 to remove)

*Loadpointer 0x0:* Script pointer

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x100 //assign to R

Loadpointer 0x0 @doSomething

Special 0xCB //now, when in the overworld, pressing R will launch @doSomething

#### Variable Math Specials

Special 0x3E – Add Variables

**Details**: Add the values of two variables together.

**Input**:

*Var 0x8004:* first value or variable holding value

*Var 0x8005:* second value or variable holding value

**Returns:**

*Var 0x80**04 or variable inside:* sum of two values

*Given Var:* 1 if sum overflows 0xFFFF, 0 otherwise

**Example Script:**

Setvar 0x8004 50

Setvar 0x8005 100

Special2 0x800D 0x3E

Buffernumber 0x0 0x8004 // addition result in [buffer1]

Compare 0x800D 0x1

If 0x1 goto @overflow

Special 0x3F – Subtract Variables

**Details**: Subtract the values inside two variables

**Input**:

*Var 0x8004:* first value or variable holding value

*Var 0x8005:* second value or variable holding value

**Returns:**

*Var 0x8004 or variable inside:* difference of Var 0x8004 – Var 0x8005

*Given Var:* 1 if difference overflows 0x0, 0 otherwise

**Example Script:**

Setvar 0x8004 0x4059

Setvar 0x8005 0x4050

Special2 0x800D 0x3F // [var 0x4059] = [var 0x4059] – [var 0x4050]

Compare 0x800D 0x1

If 0x1 goto @overflowed // [var 0x4059] > [var 0x4050]

Special 0x40 – Multiple Variables

**Details**: multiply the values of two variables together

**Input**:

*Var 0x8004:* first value or variable holding value

*Var 0x8005:* second value or variable holding value

**Returns:**

*Var 0x8004 or variable inside:* Var 0x8004 Var 0x8005

*Given Var:* 1 if product overflows 0xFFFF, 0 otherwise

**Example Script:**

Setvar 0x8004 400

Setvar 0x8005 200

Special2 0x800D 0x40 // 400\*200 = 80000 = 0x13880 = (0xFFFF) + 0x3881

Buffernumber 0x0 0x8004 // value will be 0xFFFF, 0x800D will be equal to 1

Special 0x41 – Divide Variables

**Details**: Divide the values in two variables

**Input**:

*Var 0x8004:* numerator

*Var 0x8005:* denominator

**Returns:**

*Var 0x8004:* integer result of Var8004/Var8005

*Given Var:* remainder of division

**Example Script:**

Setvar 0x8004 50

Setvar 0x8005 6

Special2 0x8006 0x41 // Var8004 = 50/6 = 8

Buffernumber 0x0 0x8006 // Remainder = 2

Special 0x42 – AND Variables

**Details**: bitwise AND two variables

**Input**:

*Var 0x8004:* first value

*Var 0x8005:* second value

**Returns:** AND result of two variables to given variable

**Example Script:**

Setvar 0x8004 0xCB // 1100 1011

Setvar 0x8005 0xAA // 1010 1010

Special2 0x8004 0x42 // [1100 1011] & [1010 1010] = 1000 1010 = 0x8A

Special 0x43 – OR Variables

**Details**: bitwise OR two variables

**Input**:

*Var 0x8004:* first value

*Var 0x8005:* second value

**Returns:** OR result of two variables to given variable

**Example Script:**

Setvar 0x8004 0x4 // 0000 0100

Setvar 0x8005 0x10 // 0001 0000

Special2 0x8004 0x43 // 0x4 | 0x10 = 0x14 = 0001 0100

Special 0x44 – XOR Variables

**Details**: bitwise XOR of two variables

**Input**:

*Var 0x8004:* first value

*Var 0x8005:* second value

**Returns:** XOR result of two variables to given variable

**Example Script:**

Setvar 0x8004 0x12 // 0001 0010

Setvar 0x8005 0x18 // 0001 1000

Special2 0x8007 0x44 // Var8007 = 0xA = 0000 1010

#### Battle Specials

Special 0x51 – Wild Shiny Battle

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x52 – Temporary Status Inducer

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x53 – Temporary Status Canceller

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x54 – Permanent Status Inducer

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x55 – Permanent Status Canceller

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x56 – Change Roaming Pokemon

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x57 – Cancel Roaming Pokemon

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x58 – Wild Data Switch

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x59 – Cancel Wild Data Switch

**Details**:

**Input**:

**Returns:**

**Example Script:**

Special 0x97 – Random Grass Battle

**Details**: Initiate a grass battle with a random pokemon from the map wild data

**Input**: none

**Returns:** nothing

**Example Script:** see Special 0x98

Special 0x98 – Random Sea Battle

**Details**: Initiates a random water battle from the map wild data

**Input**: none

**Returns:** nothing

**Example Script:**

Special 0x8f // the trainer position special\*/

Special2 0x800d 0x7f // returns 1 if ground battles, 2 if water battle

Compare 0x800d 0x1

If 0x1 goto @grass

Compare 0x800d 0x2

If 0x1 goto @water

Release

End

#org @grass

Special 0x97 //generates a random grass battle

Release

End

#org @water

Special 0x98 //generates a random water battle

Release

End

Special 0x156 – Ghost Battle

**Details**: Initiate a ghost battle with a given pokemon, level, and held item

**Input**:

*Var 0x8004:* Ghost Species (default marowak)

*Var 0x8005:* Ghost Level

*Var 0x8006:* Ghost Hold Item

**Returns:** nothing

**Example Script:**

Setvar 0x8004 PKMN\_GARCHOMP

Setvar 0x8005 100

Setvar 0x8006 ITEM\_LIFEORB

Special 0x156

#### Timer Specials

Another feature from JPANs engine, which allows the player to utilize the game timer for timed events.

Special 0x46 – Start Timer

**Details**: Start the timer. If called after it started running, it resets the timer.

**Input**: none

**Returns:** nothing

Special 0x47 – Pause Timer

**Details**: Pauses the already-started timer.

**Input**: none

**Returns:** nothing

Special 0x48 – Resume Timer

**Details**: Resume a paused timer.

**Input**: none

**Returns:** nothing

Special 0x49 – Stop Timer

**Details**: Stops the timer and returns the value. Timer needs to be started anew, resuming the timer will cause the value to be inaccurate.

**Input**: none

**Returns:** The timer value to the given variable.

**Example Script:**

Special2 0x800D 0x49

Special 0x4A – Get Timer Value

**Details**: Just return the time on the timer.

**Input**: none

**Returns:** The timer value to the given variable.

**Example Script:**

Special2 0x800D 0x4A

Special 0x4B – Stop and Update Playtime

**Details**: Stop the timer and update playtime value.

**Input**: none

**Returns:** nothing

Special 0x4C – Update Playtime

**Details**: Update the playtime. Meant for functions that take a while to process that cause delay in playtime.

**Input**: none

**Returns:** nothing

Special 0x4D – Check Timer Value

**Details**: Check if timer has reached a value stored in *Var 0x8010*.

**Input**:

*Var 0x8010:* Value to check against

**Returns:** *1* if timer is greater or equal, *0* otherwise.

**Example Script:**

Setvar 0x8010 100

Special2 0x800D 0x4d

Compare 0x800D 0x1

If 0x1 goto @timeReached

Special 0x4E – Save Timer Value

**Details**: Store the timer value to a free RAM address to allow you to later reset it to this value.

**Input**: none

**Returns:** nothing

Special 0x4F – Start Timer at a Time

**Details**: Restart the timer at the value stored with Special 0x4E

**Input**: none

**Returns:** nothing

Special 0x50 – Store Timer Value to Variable

**Details**: Store the timer value stored from Special 0x4E

**Input**: none

**Returns:**

*Var 0x8006:* Timer value

Special 0x61 – Load Timer Value from Variable

**Details**: Set the timer value RAM from a variable

**Input**:

*Var 0x8006:* Timer value to set

**Returns:** nothing

#### Safari Specials

Special 0x86 – Get Safari Balls

**Details**: Check Safari Ball quantity

**Input**: nothing

**Returns:**

*Var 0x8004:* Normal Safari ball number.

*Var 0x8005:* Extra Ball slot number

*Given Var:* Combined number

**Example Script:**

Special2 0x800D 0x86

Buffernumber 0x0 0x8004 // normal number

Buffernumber 0x1 0x8005 // extra balls

Buffernumber 0x2 0x800D // combined safari ball number

Special 0x87 – Change Safari Balls

**Details**: Increase or decrease the safari ball count, maximum 100 balls

**Input**:

*Var 0x8004:* Number to increase/decrease by.

0x1XX decreases by XX, 0x1YY increases by YY

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x0130 //remove 30 safari balls

Special 0x87

Special 0x88 - Get Safari Pedometer

**Details**: Get the value of the safari pedometer

**Input**: None

**Returns:** Safari pedometer value to given variable

**Example Script:**

Special2 0x800D 0x88

Compare 0x800D 50 // check if taken at least 50 steps

If 0x4 goto @overFifty

Special 0x89 – Set Safari Pedometer

**Details**: Set a specific number of steps in the safari

**Input**:

*Var 0x8004:* Pedometer Value to set

**Returns:** nothing

**Example Script:**

Special2 0x800D 0x88 // get current pedometer

Compare 0x800D 50

If 0x3 goto @continue

Setvar 0x8004 50 // pedometer can only reach 50 steps

Special 0x89

#### Walking Specials

Special 0x7E – Get Tile Number

**Details**: Get the tile number at a specified location on the current map

**Input**:

*Var 0x8004:* Tile x-coordinate

*Var 0x8005:* Tile y-coordinate

**Returns:** Tile number to the given variable

**Example Script:**

Getplayerpos 0x8004 0x8005 // player’s current position

Addvar 0x8004 0x2 // check tile 2 steps to the right of the player

Special2 0x800D 0x7E

Compare 0x800D SOME\_TILE\_NUMBER

If 0x1 goto @correctTile

Special 0x7F – Get Tile Behaviour

**Details**: Get a specific tile set of attributes

**Input**:

*Var 0x8004:* Tile x-coordinate

*Var 0x8005:* Tile y-coordinate

**Returns:**

*Var 0x8004:* Tile background byte

*Var 0x8005:* Tile behaviour bytes

*Given Var:* Tile background byte

**Example Script:**

Getplayerpos 0x8004 0x8005

Special2 0x800D 0x7F // get tile attributes on player’s current tile

//etc..

Special 0x81 – Set Walking Script

**Details**: Load a walking script to run each step. Setting to zero removes any walking script.

**Input**:

*Loadpointer 0x0:* Script pointer

**Returns:** nothing

**Example Script:**

Loadpointer 0x0 @walkingMsg

Special 0x81

Release

End

#org @walkingMsg

Msgbox @msg 0x6

End

#org @msg

= This msgbox will play every single step.

Special 0x8A – Read Pedometer Value

**Details**: This special is not in JPANs original engine. It reads the value of one of the extra pedometers included in the engine, which are set with flags (see customization)

**Input**:

*Var 0x8004:* Pedometer to read

0: always active pedometer (32bit)

1: large valued-pedometer (32bit)

2: medium valued-pedometer (16bit)

3: first small pedometer (8bit)

4: second small pedometer (8bit)

**Returns:** Pedometer value to the given variable

**Example Script:**

Setvar 0x8004 0x0 // pedometer that’s always on

Special2 0x800D 0x8A // get number of steps player has walked

Buffernumber 0x0 0x800D

#### PC Selection Specials

A couple new specials are added, as well as a few existing specials changed to allow data manipulation of boxed pokemon.

Special 0x1A – Store/Return Party Pokemon Data

**Details**: Save or Return party/boxed Pokemon Data

**Input**:

*Var 0x8002*:

0 for store to free ram

1 for return to party from free ram

2 for store from free ram to box

3 for store from box to free ram

*Var 0x8005:* Party slot number (for special 0xFE inputs)

**Returns:** 0 or 1 to Var800D for success/failure, respectively

**Script Example:**

Setvar 0x8002 0 //store

Setvar 0x8005 0 // save first party mon data

Special 0x1A //first party pokemon data now in Enemy data slot 5

Special 0x1B – Swap Party/Boxed Pokemon Data

**Details**: Swap party and box data

**Input**:

*Var 0x8000:* Box Number

*Var 0x8001:* Box Position

*Var 0x8002*: 0 for withdraw from box, 1 for store to box

*Var 0x8005:* Party slot number (for special 0xFE inputs)

**Returns:** 0 or 1 to Var800D for success/failure, respectively

**Script Example – Swap Party and Boxed Mon**

Msgbox @ask 0x6 // select party mon to deposit

Special 0x9F

Waitstate

Copyvar 0x8005 0x8004

Setvar 0x8002 0 // from party to free ram

Special 0x1A // store from party to free ram

msgbox @ask2 0x6 // select boxed mon to withdraw

writebytetooffset 0x1 0x0203b7ac

special 0x3C

waitstate

setvar 0x8002 0x0 //withdraw

special 0x1B // selected boxed mon to selected party slot (var8005)

setvar 0x8002 0x2 // free ram to box

special 0x1A // free ram (eg. Original selected party mon) to same box slot

Note: Rather than use a single special for this, the dynamic inputs of these specials allow for swapping party/boxed mon, trading, and more.

Special 0x7C – Buffer nickname

**Details**: Buffer a pokemon’s nickname to [buffer1]

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** nothing

**Example Script:** see Special 0x9E

Special 0x7D – Check Traded Pokemon

**Details**: Check if Pokemon is Traded

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** 0 if traded, 1 if not

**Example Script:** see Special 0x9E

Special 0x9E – Nickname Pokemon

**Details**: Nickname a Pokemon

**Input**:

*Var 0x8003:* From party (0), or box (1)

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively

From Party: *Var 0x8004* holds the party slot number

**Returns:** 0 if traded, 1 if not

**Example Script – Nickname a Boxed Pokemon**

Writebytetooffset 0x1 0x0203b7ac

Special 0x3C

Waitstate

Setvar 0x8003 0x1

Special 0x7C

Msgbox @askNickname 0x5 // nickname [buffer1]?

Compare 0x800D 0x1

If 0x0 goto @nope

Special 0x7D

Compare 0x800D 0

If 0x1 goto @traded

Special 0x9E //nickname boxed mon

Waitstate

#### Other Specials

Special 0x24 – Add Multichoice Text By Variable

**Details**: Add a dynamic multichoice option by variables

**Input**:

*Var 0x8004:* Upper halfword of pointer

*Var 0x8005:* Lower halfword of pointer

*Var 0x8006:* Multichoice Index

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x0890

Setvar 0x8005 0x5040

Setvar 0x8006 0x0

Special 0x24 //multichoice index 0 is string pointer 0x08905040

**NOTE:** Special 0x25 is much easier to use, you don’t have to worry about upper/lower sections of a word

Special 0x25 – Add Multichoice Text by Pointer

**Details**: Add a dynamic multichoice option by a pointer

**Input**:

*Var 0x8006:* Multichoice Index

*Loadpointer 0x0:* Pointer to String

**Returns:** nothing

**Example Script:**

Setvar 0x8006 0x0

Loadpointer 0x0 @option1

Special 0x25

Setvar 0x8006 0x1

Loadpointer 0x0 @option2

Special 0x25

Preparemsg @msg

Waitmsg

Multichoice 0x0 0x0 0x20 0x0

Compare 0x800D 0x0

If 0x1 goto @selectedOption1

Compare 0x800D 0x1

If 0x1 goto @selectedOption2

**NOTE:** multichoice 0xX 0xY 0xWW 0xZ

0xX: X position of box

0xY: Y position of box

0xWW: multichoice box index

0x20: 2 options (min)

…

0x25: 7 options (max)

0xZ: 0x0 if ‘B’ can cancel box, 0x1 if not

Special 0x75 – Buffer Species

**Details**: Buffer a pokemon’s species to [buffer3] and size to [buffer1]

**Input**:

*Var 0x8005:* holds the variable that stores measurements

*Var 0x8006:* species to evaluate

**Returns:** nothing

**Example Script:** See below

Special 0x76 – Measure Pokemon

**Details**: Play the measure pokemon game

**Input**:

*Var 0x8004:* party slot of pokemon

*Var 0x8005:* holds the variable that stores measurements

*Var 0x8006:* species to evaluate

**Returns:** To the given variable:

*1* if pokemon is not of selected type

*2* if pokemon is smaller

*3* if bigger, also stores biggest value in variable in *Var 0x8005*

*4* if equal

Special 0x9C – Old Man Battle

**Details**: Initiate an old man battle with a specific Pokemon and level

**Input**:

*Var 0x8004:* species

*Var 0x8005:* level

**Returns:** nothing

**Example Script:**

Setvar 0x8004 PKMN\_WEAVILE

Setvar 0x8005 50

Special 0x9C

waitstate

Special 0x18B – Show Fossil Image

**Details**: Load a custom image into the fossil image window

|  |  |  |
| --- | --- | --- |
| Pointer Table – defined in config.h | | |
| Name | Bytes | Description |
| Fossil Pointer | 4 | Pointer to a section of data that has the needed information for the function to work |
| Palette Pointer | 4 | A simple uncompressed palette must be on the other end |
| Fossil Data (Pointed to from “Fossil Pointer”) | | |
| Image Pointer | 4 | Pointer to the actual image. Uncompressed, 64x64 pixel, so each should take 4kb worth of space |
| Constant bytes | 4 | Don’t know what they do, but when changed they mess up the whole picture. Must be 0008581b |
| Null bytes | 8 | 8 zeros. Changing them will result in the same as messing with the above bytes. |

**Input**:

*Var 0x8004:* Image number

*Var 0x8005:* X coordinate on screen

*Var 0x8006:* Y coordinate on screen

**Returns:** nothing

**Example Script:**

Setvar 0x8004 0x2 //show third image in table

Setvar 0x8005 0x0

Setvar 0x8006 0x0 //at (0x0)

Special 0x18b

Special 0xA0 - Check And/Or Set Daily Event

**Details**: Checks if a daily event has been run. It can also simultaneously set a daily event to “done”.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

*Var 0x8001*: Set to 0 if you just want to check if the event has been done. Any other value sets the daily event as “done”.

**Returns**: 0 if the event has already been completed. 1 otherwise.

Special 0xA1- Update Time in Vars

**Details**: Updates the time stored in a pair of vars to the current time.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

Special 0xA2 - Get Time Difference

**Details**: Gets the time difference between the data stored in a pair of vars and the current time. These vars should be set by *special 0xA1* or *special 0xA2*.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

*Var 0x8001*: Set to one of the following values:

* *0* - Get the minute difference.
* *1* - Get the hour difference.
* *2* - Get the day difference.
* *3* - Get the month difference.
* *4* - Get the year difference.

**Example Script**:

#define SP\_DAILY\_EVENT 0xA0

#define SP\_UPDATE\_TIME\_IN\_VARS 0xA1

#define SP\_GET\_TIME\_DIFFERENCE 0xA2

#define DAILY\_EVENT\_VAR 0x50D2 'Also uses 0x50D3

#dynamic 0x740000

#org @start

setvar 0x8000 DAILY\_EVENT\_VAR

setvar 0x8001 0x0 'Don’t set daily event var to done

special2 LASTRESULT SP\_DAILY\_EVENT

compare LASTRESULT 0x0

if == goto @AlreadyDid

setvar 0x8000 DAILY\_EVENT\_VAR

special SP\_UPDATE\_TIME\_IN\_VARS

msgbox @havenot ‘“I have not done it.”

callstd MSG\_FACEPLAYER

end

#org @AlreadyDid

setvar 0x8000 DAILY\_EVENT\_VAR

setvar 0x8001 0x0 'Minute difference

special2 LASTRESULT SP\_GET\_TIME\_DIFFERENCE

buffernumber 0x0 LASTRESULT

msgbox @already ‘“\v\h02 minutes ago I did it.”

callstd MSG\_FACEPLAYER

end

# Creating New Battle Mechanics

## Moves

## Abilities

## Poke Balls

# Credits

**Graphics**:

*Golche* - Attack Particles, Battle Backgrounds, Other Graphics

*Bela* - Poke Balls

*Solo993* - Backsprites

*canstockphoto.ca* - Battle Backgrounds

**Code**:

*Skeli* - A Bunch of Stuff

*Ghoulslash* - Attack Animations, Various Code

*Lixdel* - Attack Animations

*Pret* - PokeRuby, PokeFireRed, PokeEmerald

*Sagiri* - Trainer Class Poke Balls, Pickup Update

*DizzyEgg* - Emerald Battle Engine Upgrade V1 & V2, Dizzy's Emerald Hacked Engine

*FBI* - Expanded Saveblock, Dexnav

Navenatox – Dynamic Overworld Palettes