Complete FireRed Upgrade

Contents

Set Up

Necessary Modifications

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

Pokémon Defines:
1. Open the files **include/constants/species.h** and **include/constants/pokedex.h**.
2. Modify the Pokémon indices found in this file to match the ones in your hack.
3. **DO NOT DELETE ANY POKÉMON NAMES**. If there is a Pokémon species you are not using, then set its index to **0xFEFE**.
4. If you have not added in any new Pokémon 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:
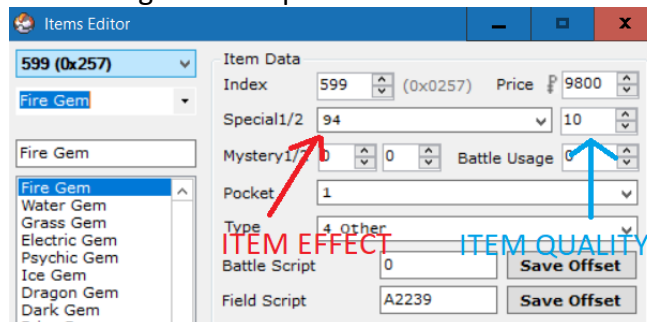
As you can see, the item effect is set to 94 and the item quality is set to 10 (the move type for *Fire*).

Configuration Options
See below.

Configurable 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
TERRAIN_VAR	Setting this var to one of the following values before initiating a battle will load the battlefield with the corresponding terrain: 1: Electric Terrain 2. Grassy Terrain 3. Misty Terrain 4. Psychic Terrain
TOTEM_VAR	This represents are series of vars using for initiating battles with Totem Pokémon. There are four vars in total, each representing a specific Pokémon slot on the field. Adding the following values to the var will indicate which slot that var is for: 0: *Player Pokémon in Singles, Left Player Pokémon in Doubles* 1: *Enemy Pokémon in Singles, Right Enemy Pokémon in Doubles* 2: *Right Player Pokémon in Doubles* 3: *Left Enemy Pokémon 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, in a single battle, having the enemy Pokémon 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 **assembly/data/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 species that is currently swarming. Swarms are set to change on a daily basis. If TIME_ENABLED is commented out, then the code will need to be modified to find an alternative method to enable swarms. It can be found in **src/wild_encounter.c**.
SWARM_MAP_NAME_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* in progress.
DEXNAV_VAR	A var that holds the species to search for in the Overworld via the *DexNav* feature. Press *Select* in the DexNav GUI to save.
SECOND_OPPONENT_VAR	A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xB*) to represent the trainer id of the second trainer in battles against two opponents. If set manually in conjunction with TWO_OPPONENT_FLAG, a battle against two opponents will be started the next time a trainer battle is initiated.
PARTNER_VAR	A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xC*) to represent the trainer id of the partner trainer in multi battles. If set manually in conjunction with TAG_BATTLE_FLAG, a battle with a partner against a single trainer will be started the next time a trainer battle is initiated. If

	setting manually, take care to set PARTNER_BACKSPRITE_VAR as well.
PARTNER_BACKSPRITE_VAR	A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xC*) to represent the backsprite id of the partner trainer in multi battles. If setting manually, take care to also set PARTNER_VAR and TAG_BATTLE_FLAG.

Flag Options

NOTE: Many of the following flags are cleared at the end of battle. To remove this, open the file **src/end_battle.c** and remove the flag from gEndBattleFlagClearTable.

Flag Definition	*Description*
INVERSE_FLAG	Setting this flag will enable Inverse Battles. 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 is in progress. This flag is automatically cleared at the end of each battle.
NO_CATCHING_FLAG	Setting this flag will cause enemy Pokémon 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_POKÉMON_FLAG	If CATCH_TRAINERS_POKÉMON is defined, then setting this flag allows the player to capture Pokémon belonging to the opposing trainer. Capturing a Pokémon 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 commented out, 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, if possible.
TAG_BATTLE_FLAG	This flag is set by the engine when the scripting

	command *trainerbattle 0xA* or *trainerbattle 0xC* 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 0xA* or *trainerbattle 0xB* 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	If TUTORIAL_BATTLES is defined, 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	Setting the flag before a wild battle starts will create the wild Pokémon with the moves given in the input vars. This works with both regular wild battles and scripted wild battles (if scripted, set the input vars before using the *wildbattle* scripting command). Setting any value to 0xFFFF will cause the default move to be loaded in that slot. Setting any value to 0x0 will load a blank move in that slot. Note that there are additional inputs for wild double battles. The input is as follows: Var 0x8000: **Move 1** - Pokémon **1** Var 0x8001: Move 2 - Pokémon **1** Var 0x8002: Move 3 - Pokémon **1** Var 0x8003: Move 4 - Pokémon **1** Var 0x8004: **Move 1** - Pokémon **2** (Wild Double) Var 0x8005: Move 2 - Pokémon **2** (Wild Double) Var 0x8006: Move 3 - Pokémon **2** (Wild Double) Var 0x8007: Move 4 - Pokémon **2** (Wild Double)
SMART_WILD_FLAG	Setting this flag allows wild Pokémon 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 Pokémon to have levels that match the highest level in your party.
HIDDEN_ABILITY_FLAG	Setting this flag before a Wild battle causes Wild Pokémon to be generated with their hidden abilities. It is cleared at the end of each battle. It also lets the *Givepokemon* scripting command give

	Pokémon with their hidden abilities.
DOUBLE_WILD_BATTLE_FLAG	Setting this flag causes all wild battles to be against two wild Pokémon in a Double battle format. It is cleared at the end of each battle.
NO_RANDOM_WILD_ENCOUNTERS_FLAG	Setting this flag will stop Pokémon from appearing while walking through grass or caves, or while surfing on water. Pokémon can still appear if the player chooses to fish, smash rocks, or use *Sweet Scent*.
FLAG_REMOVE_EVO_ITEM	A flag set by the engine to help with certain item-based evolutions.

Pedometer Flags

Setting any of these flags will initiate a pedometer of the corresponding size. The pedometer value can be read using *special 0x8A*.

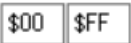| *Definition*	*Description*
FLAG_LONG_PEDOMETER	4 byte pedometer (max value 0xFFFFFFFF or 4 294 967 295)
FLAG_MED_PEDOMETER	2 byte pedometer (max value 0xFFFF or 65 535)
FLAG_SMALL_PEDOMETER_1	1 byte pedometer (max value 0xFF or 255)
FLAG_SMALL_PEDOMETER_2	1 byte pedometer (max value 0xFF or 255)

Battle Tower Options

Definition	*Description*
BATTLE_TOWER_FLAG	Setting this flag indicates to the engine that the Player is in the Battle Tower. This means that: • Trainer Pokémon will be generated within the restrictions of the tier set in the var defined in BATTLE_TOWER_TIER. The amount of Pokémon generated will match the number set in the var defined in BATTLE_TOWER_POKE_LEVEL. • Trainer Pokémon will have Pokémon 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 Pokémon 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 Pokémon 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 3: Smogon Uber 4: Smogon Little Cup 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. |

Character Customization Vars

Definition	*Description*
VAR_PLAYER_WALKRUN	Set this var to change the player's walking/running overworld sprite frames. The upper byte is used as the table Id, For example, setting to 0x0200 will load the walking/running frames from table 2, sprite 0.
VAR_PLAYER_BIKING	Switch player biking frames (same rules as above).
VAR_PLAYER_SURFING	Switch player surfing frames (same rules as above).
VAR_PLAYER_VS_SEEKER	Switch player VS Seeker frames (same rules as above).
VAR_PLAYER_FISHING	Switch player Fishing frames (same rules as above).
VAR_PLAYER_VS_SEEKER_ON_BIKE	Switch player Biking/Vs seeker frames (same rules as above).
VAR_TRAINERCARD_MALE	Set this var to the trainer sprite id of the male player front sprite that appears on the trainer card.
VAR_TRAINERCARD_FEMALE	Set this var to the trainer sprite id of the female player front sprite that appears on the trainer card.
VAR_RUNTIME_CHANGEABLE	If a person event has a given overworld table id 0xFF, it can be changed at runtime by changing these variables to a sprite number. For example, setting VAR_RUNTIME_CHANGEABLE+2 to 16, will cause all NPCs with ids 0xFF02 to appear with the little boy overworld sprite (in vanilla FR). Person event no: 2 Picture no: 0 Unknown: $00 $FF #define VAR_RUNTIME_CHANGEABLE 0x4080 #org 0x800000 setvar VAR_RUNTIME_CHANGEABLE+2 16 'Var 0x4082

Healing Place Hack

The following vars relate to JPAN's healing place hack. If SET_HEALING_PLACE_HACK is not defined, ignore these vars.

VAR_HEALINGMAP	Set this var to the map and bank for the player to respawn to after whiting out. For example, if it is set to 0x0104, the player will respawn in their room (vanilla FR). Configure in asm_defines.s
VAR_HEALING_XPOS	Set this var to the x-position the player will respawn at on the map in VAR_HEALINGMAP.
VAR_HEALING_YPOS	Set this var to the y-position the player will respawn at on

	the map in VAR_HEALINGMAP.

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 Pokémon in the regional Pokedex.
NATIONAL_DEX_COUNT	Number of Pokémon in the national Pokedex.
MAX_NUM_POKÉMON	Set to the highest Pokémon index + 1.
MAX_LEVEL	The highest possible level for a Pokémon. If you change this value, make sure the also modify the equivalent value found in "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 Pokémon, update this number.
EV_CAP	The most EVs a Pokémon can accrue for a given stat.
DUSK_BALL_MULTIPLIER	The catch rate for Dusk Balls.
STANDARD_IV	The number of IVs for each stat that standard Trainer's Pokémon are generated with.
SWARM_CHANCE	The chance in percent that a swarm Pokémon will appear 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 Pokémon will obey the

	player at if they have no badges.
BADGE_1_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 1 badge.
BADGE_2_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 2 badges.
BADGE_3_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 3 badges.
BADGE_4_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 4 badges.
BADGE_5_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 5 badges.
BADGE_6_OBEDIENCE_LEVEL	The highest level that a traded Pokémon will obey the player at if they have 6 badges.
BADGE_7_OBEDIENCE_LEVEL	The highest level that a traded Pokémon 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
EXISTING_FOSSIL_IMAGE_TABLE_ADDRESS	If FOSSIL_IMAGE HACK is defined, and you already have a fossil image table inserted somewhere in your hack that you wish to use, uncomment this line and replace the given pointer with your pointer. See Special 0x18B for table details.
EXISTING_OW_TABLE_ADDRESS	If you have used JPAN's hacked engine to add new overworld tables in, and you would like to use the table already inserted in your hack to keep track of these tables, uncomment this line and replace the given pointer with your pointer.

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 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 **include/new/DNS.h**.
OVERWRITE_RIVAL	Loads the buffered rival's name for trainer classes 0x51, 0x59, and 0x5A.
TRAINER_CLASS_POKE_BALLS	Creates trainer Pokémon 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 Pokémon 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.h**. For example, setting the IV value to 1 will loaded the spread labeled "1" in gTrainersWithEvsSpreads. Modifying the required conditions to activate this feature can be done by searching for the line *#ifdef TRAINERS_WITH_EVS* in **src/build_Pokémon.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 LAST_RESULT (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 Pokémon 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 or viewing a Pokémon's moves on the summary screen, the move type the move will become when used 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 Pokémon 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* and *Return* 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 to be spawned in maps using the Tanoby Ruins map names according to the current chamber. Error prevention has been added to also allow random Unown to be generated outside of the Tanoby Ruins maps.
ALTERING_CAVE_ENABLED	If the current map is the Altering Cave and Var 0x4024 is set, Wild Pokémon will spawn based on the contents of the var.
SWEET_SCENT_ONLY_IN_CLEAR_WEATHER	In certain generations, Sweet Scent only spawns wild Pokémon in the Overworld if the weather is clear.
OBEDIENCE_BY_BADGE_AMOUNT	Pokémon 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".
SAVE_BLOCK_EXPANSION	Expands the amount of memory that is saved when the player saves the game. This feature breaks compatibility with the FR *Mystery Gift* and *Trainer Tower* features. Uncommenting this line also requires removal of all related hooks. Search for *Save Expansion Hooks* in **hooks**.
SELECT_FROM_PC	If uncommented, allow the player to select and manipulate data of Pokémon from the PC storage boxes. See PC Selection for more details.
SET_HEALING_PLACE_HACK	If uncommented, the whiteout hack from JPAN's FR engine is implemented, allowing *VAR_HEALINGMAP*, *VAR_HEALING_XPOS*, and *VAR_HEALING_YPOS* to be utilized to overwrite the default respawn point.
FOSSIL_IMAGE_HACK	Grants the ability to load custom images from a table using Special 0x18B. If *EXISTING_FOSSIL_IMAGE_TABLE_ADDRESS* is commented out, then the table of images can be found by searching for gFossilImageTable in **src/script_specials.c**. Otherwise the table is loaded from *EXISTING_FOSSIL_IMAGE_TABLE_ADDRESS*.
EVO_HOLD_ITEM_REMOVAL	Evolving a Pokémon by having it hold an item upon level up or trading removes the item after evolution (like normal). Commenting this out means Pokémon will retain their items after evolution.
EXPAND_MOVESETS	Adds level up moves for each Pokémon which can be found in **src/Tables/Learnsets.c**. Comment this line if you would rather use the learnsets created in the *Dynamic Pokémon Expansion*. Commenting this line out without properly having expanded the level up moves in some way will cause Pokémon to learn garbage moves.
FATHER_PASSES_TMS	During breeding, any TMs the father knows will be passed down to the baby if it can learn that TM. This feature was removed from main series Pokémon games in Gen 6.

GIVEPOKEMON_BALL_HACK	The last byte of the *Givepokemon* scripting command allows you to pass in a Poké Ball type to assign to the pokemon. Ball types can be found in **include/new/catching.h**.

Misc Battle Effect Options

Definition	Description
OLD_BURN_DAMAGE	Burn damage does 1/8 of max health instead of 1/16.
OLD_PARALYSIS_SPD_DROP	Paralysis lower Speed down to 1/4 instead of ½.
OLD_CONFUSION_CHANCE	Confusion stops attacks 50% of the time instead of 33%.
INFINITE_WEATHER	Weather abilities make weather last for infinite turns.
INFINITE_TERRAIN	Terrain abilities make terrain last for infinite turns.
NO_SHEER_COLD_NERF	Remove all Gen 7 Sheer Cold nerfs.
OLD_MOVE_SPLIT	The Physical/Special 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 activates regardless of the user's HP.
OLD_PRANKSTER	Prankster won't fail against Dark-Types.

Damage Calculation Options

Definition	Description
OLD_CRIT_DAMAGE	Critical hits to do 2x damage; 3x with Sniper.
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 Pokémon stat boosts.
OLD_ATE_BOOST	"Ate" abilities give a 1.3x boost instead of 1.2x.
OLD_GEM_BOOST	Gems give a 1.5x boost instead of 1.2x.
OLD_EXPLOSION_BOOST	Exploding moves halve the target's Defense.
OLD_HIDDEN_POWER_BP	Hidden Power has its Base Power calculated from the attacker's IVs.
PORTAL_POWER	Enables Hoopa-Unbound's signature ability in Pokémon Unbound, *Portal Power*. This reduces the power of non-contact moves by 25%.
OLD_SOUL_DEW_EFFECT	Soul Dew doubles Latios & Latias' Sp. Atk & Sp. Def.
OLD_PARENTAL_BOND_DAMAGE	The second hit of Parental Bond does 50% of the original damage instead of 25%.

Capturing Pokémon Options

Definition	Description
CATCH_TRAINERS_POKÉMON	Allows the possibility to capture trainer Pokémon by setting the CATCH_TRAINERS_POKÉMON_FLAG.

NO_HARDER_WILD_DOUBLES	In Gen 5, Pokémon encountered in wild double battles were harder to catch (based on how many species are owned). This feature implements that catch rate decrement.
CRITICAL_CAPTURE	Allows for Critical Capture to occur. The odds at which this occurs can be found in the function: *static bool8 CriticalCapture(u32 odds)* found in **src/catching.c**.

Exp. Gain Options

Definition	*Description*
OLD_EXP_SHARE	The Exp. Share acts like it did before Gen 6.
TRAINER_EXP_BOOST	Gives an Exp boost for defeating a trainer's Pokémon. (Pre Gen 7)
OLD_EXP_SPLIT	Exp. is split amongst all participating Pokémon. (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 Pokémon'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.
CAPTURE_EXPERIENCE	When a Pokémon is caught, experience will be rewarded as if the caught Pokémon fainted.

Other Options

Definition	*Description*
ACTIVATE_DOUBLE_BATTLE_FROM_FLAG	Allow DOUBLE_BATTLE_FLAG to be set to activate double battles for trainers if possible.
NO_GHOST_BATTLES	Disables the Ghost battle feature from Pokémon 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, it only adds features that supports it.
OBEDIENCE_CHECK_FOR_PLAYER_ORIGINAL_ POKÉMON	Opens up the possibility that the Player's Pokémon can disobey them, as opposed to just traded Pokémon.
WILD_ALWAYS_SMART	All wild Pokémon use AI features meant for trainers.
HAIL_IN_BATTLE	Enables the Hail weather effect in battle when the OW weather is set to WEATHER_STEADY_SNOW (0x7).
FOG_IN_BATTLE	Enables the 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 listed 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.

Engine 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, if you're not using the *Dynamic Pokémon Expansion*, your Pokémon Editor of choice will need to be modified:

G3T:
In your Gen3Tools folder, open up **Customisation/Pokémon Editor.ini**, and add the line **FE=Mega Evolution** to the bottom of the file.

```
0D=Allow PKMN creation
0E=Create extra PKMN
0F=Beauty
FE=Mega Evolution
```

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*:

```
evolutionmethods = Breeding Only,Friendship,Friendship (Day),Friendship (N
one,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,None,None,None,None,None,None,None,None,None,None,None,
None,None,None,Mega Evolution
```

3. Modify "evomethodsproperties" such that the 254th method is set to *Item*.

[Mega Evolution](#)

Mega Evolution set up is similar to the how the [previous](#) Mega Evolution system by Touched was set up. If a Pokémon is able to Mega Evolve, Mega Evolution can be trigged by pressing start on the move menu once the mega trigger appears.

NOT Using Dynamic Pokémon Expansion:
1. Create a Mega Stone in G3T with the following format:
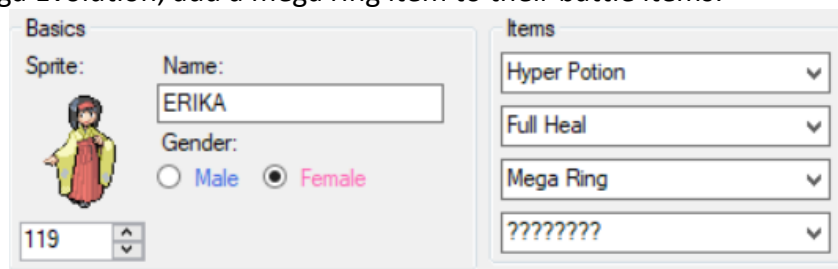
2. Create a Mega Ring key item. Its format is the same as any other key item.
3. Update the definition for the Mega Ring item in **include/constants/items.h**.
4. Open the file **src/mega.c** and search for KeystoneTable. Add your mega ring item to this table:

```
item_t KeystoneTable[] =
{
    ITEM_MEGA_RING,
};
```

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

5. 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 Pokémon the relevant Mega Stone to hold!

6. Find the Pokémon 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 Pokémon 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 Pokémon 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 Pokémon 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.

7. 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!

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

Using Dynamic Pokémon Expansion:
Add evolution methods in **src/Evolution_Table.c** with the following format:

Regular Mega Evolution:

```
[SPECIES_VENUSAUR] = {{EVO_MEGA, ITEM_VENUSAURITE, SPECIES_VENUSAUR_MEGA, MEGA_VARIANT_STANDARD}},
[SPECIES_VENUSAUR_MEGA] = {{EVO_MEGA, ITEM_NONE, SPECIES_VENUSAUR, MEGA_VARIANT_STANDARD}},
```

Wish Mega Evolution:

```
[SPECIES_RAYQUAZA] = {{EVO_MEGA, MOVE_DRAGONASCENT, SPECIES_RAYQUAZA_MEGA, MEGA_VARIANT_WISH}},
[SPECIES_RAYQUAZA_MEGA] = {{EVO_MEGA, MOVE_NONE, SPECIES_RAYQUAZA, MEGA_VARIANT_WISH}},
```

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 Pokémon, effectively allowing for two Mega Evolutions per side during any given battle.
- Mega Evolution is not prevented if any Pokémon on the side Ultra Bursted or underwent Primal Reversion.
- If Rayquaza uses a Z-Move, it'll be unable to Mega Evolve.

[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.

NOT Using Dynamic Pokémon Expansion:

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).

2. 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 Pokémon 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.

3. 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 Pokémon 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**:

```
bool8 IsBluePrimal(u8 bank)
{
    u16 species = GetBankPartyData(bank)->species;
    return species == PKMN_KYOGRE_PRIMAL;
}

bool8 IsRedPrimal(u8 bank)
{
    u16 species = GetBankPartyData(bank)->species;
    return species == PKMN_GROUDON_PRIMAL;
}
```

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

```
bool8 IsRedPrimal(u8 bank)
{
    u16 species = GetBankPartyData(bank)->species;
    return species == PKMN_GROUDON_PRIMAL || species == PKMN_DIALGA_PRIMAL;
}
```

Don't forget to also define PKMN_PRIMAL_DIALGA in **include/constants/species.h**!

Using Dynamic Pokémon Expansion:
Add evolution methods in **src/Evolution_Table.c** with the following format:

```
[SPECIES_GROUDON] = {{EVO_MEGA, ITEM_RED_ORB, SPECIES_GROUDON_PRIMAL, MEGA_VARIANT_PRIMAL}},
[SPECIES_GROUDON_PRIMAL] = {{EVO_MEGA, ITEM_NONE, SPECIES_GROUDON, MEGA_VARIANT_PRIMAL}},
```

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.

NOT Using Dynamic Pokémon Expansion:
1. Create an *Ultranecrozium Z* item in G3T with the following format:

2. 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 Pokémon 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.

3. 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 3. 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 (it needs for at least one of them). *Ultra Necrozma* will always revert to the form it *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).

Using Dynamic Pokémon Expansion:
Add evolution methods in **src/Evolution_Table.c** with the following format:

```
[SPECIES_NECROZMA_MANE]  =  {{EVO_MEGA, ITEM_ULTRA_NECROZIUM_Z, SPECIES_NECROZMA_ULTRA, MEGA_VARIANT_ULTRA_BURST}},
[SPECIES_NECROZMA_WINGS] = {{EVO_MEGA, ITEM_ULTRA_NECROZIUM_Z, SPECIES_NECROZMA_ULTRA, MEGA_VARIANT_ULTRA_BURST}},
[SPECIES_NECROZMA_ULTRA] = {{EVO_MEGA, ITEM_NONE, SPECIES_NECROZMA_MANE, MEGA_VARIANT_ULTRA_BURST},
                            {EVO_MEGA, ITEM_NONE, SPECIES_NECROZMA_WINGS, MEGA_VARIANT_ULTRA_BURST}},
```

Z-Moves

Z-Moves work akin to how they work in real Pokémon games.

If a Pokémon holds a *Z-Crystal* corresponding to a specific type, any move of that type can be turned into a Z-Move by pressing the *Start*-button on the move menu, and then the *A*-button to confirm the selection. If a move cannot be turned into a Z-Move, the *Start*-button will do nothing.

If a certain Pokémon holds its signature *Z-Crystal*, then its signature move can be turned into its signature *Z-Move* (also with the *Start*-button). A list of these signature Z-Moves can be found under *gSpecialZMoveTable* in **src/include/new/battle_start_turn_start.h**.

The only setup required for Z-Moves involves the creation of Z-Crystals. Each Z-Crystal should be created in G3T with the following format:

The item effect should be set as 130 and the item quality should be set to the move type the Z-Crystal works for (in the above image it is set to 2 [TYPE_FLYING]). The *Mystery 1* byte can also be set to 1 to remove the item quantity for Z-Crystals (acts as if the player only has the single, unique Z-Crystal).

Special Z-Crystals should be set up similarly, the key difference being that item quality should be set to 255.

Once the Z-Crystals are created, have a Pokémon hold one, give it the appropriate move, and then watch the magic happen!

Trainer Sliding Messages[1]

In generations after Gen 3, Trainers could interrupt the battle with a message. This engine supports three kinds of those messages:

1. After the opponent's first Pokémon faints.
2. After the opponent's last Pokémon is sent in.
3. When the opponent's last Pokémon is low on health.

Trainers can have any combination (or none at all) of the above messages. To set these up, do the following:

1. Navigate to **src/trainer_sliding.c** and search for *sTrainerSlides*. This is the table used to define the sliding messages.
2. Add an entry with the following format:

```
{0x59, sText_BenFirstMonDown, sText_BenLastSwitchIn, sText_BenLastLowHP},
```

Where 0x59 is the trainer Id (Youngster Ben in this case), *sText_BenFirstMonDown* is the message displayed when the opponent's first Pokémon has fainted, *sText_BenLastSwitchIn* is the message displayed when the opponent switches in their last Pokémon, and *sText_BenLastLowHP* is the message displayed when the opponent's last Pokémon is on low health. If you would not like the trainer to say anything at one of these stages, simply replace the entry with *NULL*:

```
{0x59, sText_BenFirstMonDown, NULL, sText_BenLastLowHP},
```

In this case, the trainer will say something after the first Pokémon is defeated and when their last Pokémon is low on HP, but not after they send in their last Pokémon.

3. At the top of the file (under the #include's), add declarations for the strings you've defined. For example, for the first message struct declared above, the top of the file should look like this:

```
#include "defines.h"
#include "defines_battle.h"
#include "../include/new/helper_functions.h"

extern u8 sText_VegaLastSwitchIn[];
extern u8 sText_VegaLastLowHP[];
extern u8 sText_VegaFirstMonDown[];
```

Each string title is preceded by "extern u8 " and ends with "[];".

---

[1] Credits to the Emerald Battle Engine Upgrade V2.0 for the original source code.

4. Open the file **strings/trainer_sliding_strings.string** and add entries for the strings you've defined. For example, for the first message struct declared above:

```
#org @sText_BenFirstMonDown
You hurt my friend!\p

#org @sText_BenLastSwitchIn
It's all or nothing, now!\p

#org @sText_BenLastLowHP
Oh, no[.]\p
```

The format for the strings follows similarly to *XSE*, the key difference being that each line does **not** start with "= ". An equal's sign at the beginning of the line will be treated as such and be seen in the game. Also note that each line ends with "\p" in order to wait for the player's key press.

Follows these steps and continuously add new entries to the table to add flavour to battles!

Multi Battles

There are 4 different types of Multi Battles supported by this engine:

1. Player Vs. Two Trainer Opponents.
   - ○ Can be set up in a script or by being spotted by two different trainers.
2. Player & Partner Vs. Two Trainer Opponents.
3. Player & Partner Vs. One Trainer Opponent.
4. Player & Partner Vs. Two Wild Pokémon.

There are two different ways of setting these up:

XSE Friendly Method
This method allows you to set up multi battles through scripted events.

- To set up a battle against two opponents:
  - Set the flag TWO_OPPONENT_FLAG.
  - Set the var SECOND_OPPONENT_VAR to the trainer id of the second trainer.
  - Use the *loadpointer* scripting command in conjunction with special **0xXX** to load the second trainer's defeat text.
  - Then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle.
- To set up a battle with a partner:
  - Set the flag TAG_BATTLE_FLAG.
  - Set the var PARTNER_VAR to the trainer id of the partner.
  - Set the var PARTNER_BACKSPRITE_VAR to the backsprite Id of the partner.
  - Then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle.
  - **This will initiate a battle with a partner against a single opponent.**
- To battle with a partner and two opponents, set all flags and vars listed in the previous two steps, and then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle. A sample script to do this looks as follows:

```
#define TWO_OPPONENT_FLAG 0x909
#define TAG_BATTLE_FLAG 0x908
#define SECOND_OPPONENT_VAR 0x5010
#define PARTNER_VAR 0x5011
#define PARTNER_BACKSPRITE_VAR 0x5012
#define SPECIAL_LOAD_SECOND_DEFEAT_TEXT 0xAC

#org @start
setflag TWO_OPPONENT_FLAG 'Setup battle against two opponents
setflag TAG_BATTLE_FLAG 'Setup battle with partner
setvar SECOND_OPPONENT_VAR 0x59 'Youngster Ben
setvar PARTNER_VAR 0x5B 'Team up with Youngster Josh
setvar PARTNER_BACKSPRITE_VAR 0x2 'Brendan's Backsprite
loadpointer 0x0 @SecondTrainerDefeatText
special SPECIAL_LOAD_SECOND_DEFEAT_TEXT
trainerbattle 0x3 0x5A 0x0 @FirstTrainerDefeatText 'Youngster Calvin
end
```

Non-XSE Friendly Method

This method for setting up multi battles is not possible to code in XSE, but it is significantly easier to code and allows more versatility with random trainer scripts. It is recommended to use the Thumb assembler in conjunction the xse defines provided in this engine to compile these custom scripts (which can then be called after inserting the hex data).

- To set up a battle against two opponents, use the following scripting command:

  trainerbattle 0xB **FOE_1_ID** *FOE_2_ID* **FOE_1_NPC_ID** *FOE_2_NPC_ID* 0x0 **INTRO_TEXT_A** *INTRO_TEXT_B* **DEFEAT_TEXT_A** *DEFEAT_TEXT_B* **CANNOT_BATTLE_TEXT**

  Where:
    - FOE_1_ID: The trainer Id of the first opponent.
    - FOE_2_ID: The trainer Id of the second opponent.
    - FOE_1_NPC_ID: The local Id (person Id in Advance Map) of the first opponent.
    - FOE_2_NPC_ID: The local Id (person Id in Advance Map) of the second opponent.
    - INTRO_TEXT_A: The intro battle text said by the first opponent.
    - INTRO_TEXT_B: The intro battle text said by the second opponent.
    - DEFEAT_TEXT_A: The defeat text said by the first opponent.
    - DEFEAT_TEXT_B: The defeat text said by the second opponent.
    - CANNOT_BATTLE_TEXT: The text said by either opponent when the player doesn't have enough viable Pokémon to fight with.

  This trainerbattle 0xB command is special such that you can assign it to random NPCs to effectively make better random double battles than with a *Twins* class, for example. When using this on two random NPCs, make sure they stand next to each other! Otherwise it'll look off when they walk towards the player together.

- To set up a battle with a partner, use the following scripting command:

  trainerbattle 0xC **FOE_ID** *PARTNER_ID* **PARTNER_BACKSPRITE_ID** 0x0 *DEFEAT_TEXT*

  Where:
    - FOE_ID: The trainer Id of the opponent.
    - PARTNER_ID: The trainer Id of the player's partner.
    - PARTNER_BACKSPRITE_ID: The backsprite Id of the player's partner.
    - DEFEAT_TEXT: The text said when the opponent loses the battle.

- To battle with a partner and two opponents, use the following scripting command:

  trainerbattle 0xA **FOE_1_ID** *FOE_2_ID* **PARTNER_ID** *PARTNER_BACKSPRITE_ID* 0x0 **DEFEAT_TEXT_A** *DEFEAT_TEXT_B*

  Where each of the title definitions is the same as listed *trainerbattle 0xB* and *trainerbattle 0xC*. Note that both *trainerbattle 0xB* and *trainerbattle 0xC* cannot be used on random NPCs. They must be used from within and event script.

And with that, you can set up amazing multi battles!

Wild Double Battles

Encountering two wild Pokémon at once was introduced in Gen 4 with it occurring when the player was teamed up with another player. Then, in Gen 5, it became possible to encounter two Pokémon at once in special grass. This engine supports both of those features.

Wild Double Battles With Partner

If there is ever a situation where you'd like all wild battles in a given area to be with a partner, add the following as an *On entering map/on menu close [5]* level script in Advance Map:

#define DOUBLE_WILD_BATTLE_FLAG 0x9F9

#org @start
setflag DOUBLE_WILD_BATTLE_FLAG
setflag TAG_BATTLE_FLAG 'Setup battle with partner (same as above)
setvar PARTNER_BACKSPRITE_VAR 0x2 'Brendan's Backsprite (same as above)
end

This will cause all battles against trainers on the map to be fought with a partner, and all wild battles fought on the map to be against two Pokémon. Conditions can also be added to the script (such as a checkflag to only execute the script if some flag is set). Don't forget to clear these flags once the player no longer needs a partner!

Wild Double Grass

Special grass tiles that can initiate wild battles can be created as well. 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. Wild double water tiles are similarly done using 0x6 and 0x26. The chance a wild double battle will be started when walking in this grass is determined by the value set (in percent) in WILD_DOUBLE_RANDOM_CHANCE.
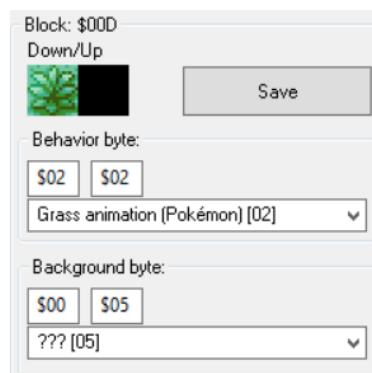
Scripted Wild Double Battles

Here is a sample script:

wildbattle 0xFFFF 0x0 0x0 'Double indicator
wildbattle PKMN_CLEFAIRY 20 ITEM_NONE
wildbattle PKMN_PIKACHU 20 ITEM_NONE
special 0x138 'Starts the battle
waitstate
Sample double wild grass in Advance Map:

I'm not entirely sure if this script will compile properly in XSE, but if you use it in PKSV by replacing *wildbattle* with *battle* and *waitstate* with *waitspecial* the battle will begin properly.

Trainer Backsprites

Adding a backsprite into the game has never been easier.

1. Go to **graphics/Backsprites** and add a minimum of **4** backsprites in for your new trainer. Make sure they're each a size of 64 x 64 and indexed to 16 colours! Sample *Brendan* and *May* backsprites come bundled with the engine.

Brendan_Back_0.   Brendan_Back_1.   Brendan_Back_2.   Brendan_Back_3.
bmp   bmp   bmp   bmp

2. Open **assembly/data/Trainer_Backsprite_Table.s** to allow the game to read your new backsprites.
3. Search for *gTrainerBackPicPaletteTable* and add your new entry after the old man. The format should be [FILE_NAME_NO_EXTENSION]0Pal. So, for instance, if I was adding the above Brendan sprites to the table:

```
.word 0x8E76F0C, 0x4 @Primo
.word 0x8E76F34, 0x5 @Old Man
.word Brendan_Back_0Pal, 0x6
```

Notice that each entry has an associated number. Numbers should go up by 1 for each entry. This is the **backsprite Id** referenced several other times in the documentation.

4. Search for *gTrainerBackAnimsPtrTable* and add your new entry after the old man. So, since in our example we're adding in another backsprite with 4 frames, copy the frame data from *May*:

```
.word 0x8239F6C @May
.word 0x8239F54 @Primo
.word 0x8239F5C @Old Man
.word 0x8239F6C @Brendan New
```

5. Search for *gTrainerBackPicCoords* and add your new entry after the old man's. For simplicity, just copy the old man's data:

```
.byte 0x8, 0x4, 0x0, 0x0 @Primo
.byte 0x8, 0x4, 0x0, 0x0 @Old Man
.byte 0x8, 0x4, 0x0, 0x0 @Brendan New
```

6. Search for *TrainerBackspritesTable* and add your new entry after the old man's. To add a new entry in this table, copy the old man's data and just change wherever it says "OldMan" to your new name. So in our example:

```
@5 - Old Man
.hword 0xFFFF
.hword 0x0
.word BackspriteOAM, 0x0
.word BackspriteImageTable_OldMan
.word BackspriteRotscaleTable
.word BackspriteAnimCmd

@6 - Brendan_New
.hword 0xFFFF
.hword 0x0
.word BackspriteOAM, 0x0
.word BackspriteImageTable_Brendan
.word BackspriteRotscaleTable
.word BackspriteAnimCmd
```

7. Scroll down to the end of the file and add a backsprite image table in. Copy the backsprite image table name from the entry just added to *TrainerBackspritesTable*, and then add the data like so:

```
BackspriteImageTable_Brendan:
.word Brendan_Back_1Tiles
.hword 0x800, 0x0

.word Brendan_Back_2Tiles
.hword 0x800, 0x0

.word Brendan_Back_3Tiles
.hword 0x800, 0x0

.word Brendan_Back_0Tiles
.hword 0x800, 0x0
```

Notice that the order of the images goes frame 1, frame 2, frame 3, and then frame 0. This is very important!

Unlike what was shown in the tutorial, it is also possible to add in data for a backsprite with 5 frames. In that case, just copy the data from one of the backsprite with 5 frames (like the player) instead.

Battle Terrain

If you have inserted any new battle backgrounds using the tutorial here, then certain modifications will need to be made to make the engine compatible with those backgrounds.

1. Open up the file **include/battle.h**. Search for BATTLE_TERRAIN_CHAMPION in the file, and add a new definition there. So, for instance, if your new background was a snow field:

```
#define BATTLE_TERRAIN_LANCE         0x12
#define BATTLE_TERRAIN_CHAMPION      0x13
#define BATTLE_TERRAIN_SNOW_FIELD    0x14
```

2. Open up the file **src/Tables/Terrain_Table.c**. Search for BATTLE_TERRAIN_CHAMPION in the file, and add a new entry to the table. So continuing on with the snow example:

```
[BATTLE_TERRAIN_CHAMPION + 4] =
    {
        .camouflageType = TYPE_NORMAL,
        .secretPowerEffect = MOVE_EFFECT_PARALYSIS,
        .secretPowerAnim = MOVE_BODYSLAM,
        .naturePowerMove = MOVE_TRIATTACK,
        .burmyForm = SPECIES_BURMY_TRASH,
    },

[BATTLE_TERRAIN_SNOW_FIELD + 4] =
    {
        .camouflageType = TYPE_ICE,
        .secretPowerEffect = MOVE_EFFECT_FREEZE,
        .secretPowerAnim = MOVE_AVALANCHE,
        .naturePowerMove = MOVE_FROSTBREATH,
        .burmyForm = SPECIES_NONE,
    },
```

Notice the entry name is the same as the definition from earlier **+ 4**.
The table has entries for:
- camouflageType: The type the move Camouflage transforms into.
- secretPowerEffect: The secondary effect of the move Secret Power.
- secretPowerAnim: The animation of the move *Secret Power*.
- naturePowerMove: The attack the move Nature Power becomes.
- burmyForm: The form Burmy transforms into after this battle. If you would not like *Burmy* to change form after this battle, leave it as SPECIES_NONE (as shown above).

Battle Music

There are three different music tables that can be set up. Each table can be found in **src/Tables/Music_Tables.c**. Any custom song definitions used in these tables should be added to **include/constants/songs.h**. Trainer classes can be found and added into the file **include/constants/trainer_classes.h**.

Class Based Encounter Music

If ENCOUNTER_MUSIC_BY_CLASS is defined, then the table, *gClassBasedTrainerEncounterBGM*, can be modified to determine which music plays in the background when each trainer class is encountered in the overworld. For example, this table causes all *Youngsters* to have the same encounter music, without having to the set the byte for each of them in their trainer data. If you wanted to change which encounter music Youngsters have, all you have to do is make the following change:

<div align="center">

`[CLASS_YOUNGSTER] = BGM_EYE_BOY,` → `[CLASS_YOUNGSTER] = BGM_EYE_GIRL,`

</div>

Now, all *Youngsters* will play the girl encounter when they spot the player.
If you do not want to use this feature, the switch statement in *SetUpTrainerEncounterMusic* found in **src/overworld.c** will need to be modified to add new encounter song Ids in.

Class Based Battle Music

The table, *gClassBasedBattleBGM*, can be modified to determine the song that plays during trainer battles against certain classes. For example, if I wanted to make all *Team Rocket* battles play the *Gym Leader* theme, I would add the following onto the end of the table:

<div align="center">

```
        [CLASS_ELITE_4] = BGM_BATTLE_GYM_LEADER,
        [CLASS_TEAM_ROCKET] = BGM_BATTLE_GYM_LEADER,
};
```

</div>

In a *Multi Battle*, if either trainer has custom battle music, their theme will play. If both trainers have custom battle music, then the theme for the trainer on the right (first opponent) will play.

Wild Species Based Battle Music

The table, *gClassBasedTrainerEncounterBGM*, can be modified to determine the song that plays during wild battles against certain species of Pokémon. For example, if I wanted to make all *Rattata* battles play the *Deoxys* theme, I would add the following onto the end of the table:

<div align="center">

```
        [SPECIES_DEOXYS] = BGM_BATTLE_DEOXYS,
        [SPECIES_RATTATA] = BGM_BATTLE_DEOXYS,
};
```

</div>

Poke Balls

Several new Poké Balls have been added to the engine, in addition to the Trainer Class Based Poké Ball hack.

Adding Support for Added Balls

Although catching data has been added in for the new balls, item data has not. This means that if you want to give the player a certain ball, you'll need to add in item data for it. Adding a new Poké Ball follows the following format in G3T:

Regarding the *Type* field, this related to the *Ball Type* - 1 of the given ball. So looking in **include/new/catching.h**, we see that the *Dusk Ball's* type is 13, so subtracting 1 is 12.

Class Based Poke Balls

Loosely based on the hack created by Sagiri, if TRAINER_CLASS_POKE_BALLS is defined, this implements the feature from Gen 7 where certain Trainer classes always send out Pokémon in a specific type of Poké Ball.

To modify the trainer class balls, open **src/Tables/Class_Based_Poke_Ball_Table.c**. All the trainer classes have been preloaded into the table, but if you would like to change a trainer class name to your own custom name, do so in **include/constants/trainer_classes.h** and then update the table accordingly. The ball type defines that can be used can be found in **include/new/catching.h**. So, for example, if you wanted to change the *Youngster*'s ball to a *Great Ball*, you'd make the following change:

And then the next time the player battles a *Youngster*, all their Pokémon would be sent out in *Great Balls*!

Trainers With EVs

Loosely based on [DoesntKnowHowToPlay's hack](), if TRAINERS_WITH_EVS is defined, this feature allows trainer Pokémon to have certain EV spreads. Preloaded spreads can be found in **src/Tables/Trainers_With_EVs_Table.h**. New spreads should also be added in here as well.

Note that since this is a **.h** file, if you want your changes to compile, you will need to make a change to the file **src/build_pokemon.c** so the compiler recompiles the Trainer's With EVs file. The change can be as simple as adding a whitespace character and then removing it. **Don't use the undo button to remove the change!** The change needs to be removed manually by using the backspace character so the editor saves the file with an updated timestamp.

Assigning a Spread

In order for a Pokémon to use a spread, the following must be done:

- The Pokémon must have a custom moveset.
- The Pokémon must have a custom item (this item can still be set to 0).
- The Pokémon must have an IV (labeled "EVs" in most Trainer editors) value greater than 1 (the 0th spread is left empty for this reason).

For example, looking in the file, you can see that spread 5 is predefined as a "Bulky Physical Attacker". If I wanted to assign this to a Pokémon, the layout would look like this (in [Hopeless Trainer Editor]()):
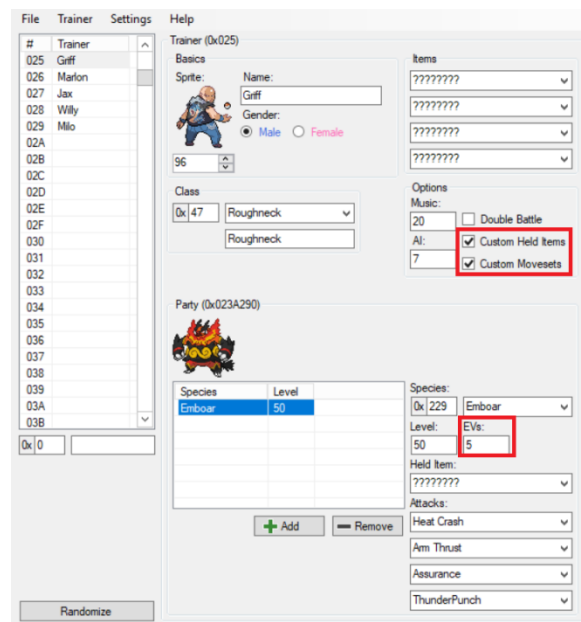
As can be seen in the image, both *Custom Movesets* and *Custom Hold Items* have been selected, and the *EVs* value has been set to 5. Note that the hold item has been set to "*????????*", meaning that *Emboar* is not actually holding an item (this is still fine).

Creating a New Spread

If you want to create a new spread, add it to the end of the table and start counting at 31. The table can at most hold up to the 255th spread. Spreads have the following members:

- .nature: The nature of the Pokémon. Natures can be found in **include/pokemon.h**.
- .ivs: The IVs for the Pokémon. All stats are set to this value (meaning custom *Hidden Powers* are not possible).
- .hpEv: The number of *HP* EVs the Pokémon has.
- .atkEv: The number of *Attack* EVs the Pokémon has.
- .defEv: The number of *Defense* EVs the Pokémon has.
- .spAtkEv: The number of *Special Attack* EVs the Pokémon has.
- .spDefEv: The number of *Special Defense* EVs the Pokémon has.
- .spdEv: The number of *Speed* EVs the Pokémon has.
- .ball: If set to TRAINER_EV_CLASS_BALL and TRAINER_CLASS_POKE_BALLS is defined, then the ball loaded will be the one for the trainer class. Otherwise look in **include/new/catching.h** for a list of available Poké Ball types.
- .ability: The ability number of the Pokémon. Can be set to:
    - Ability_1: The Pokémon will have its first ability.
    - Ability_2: The Pokémon will have its second ability.
    - Ability_Hidden: The Pokémon will have its hidden ability.
    - Ability_Random_1_2: The Pokémon will have one of its primary abilities.
    - Ability_RandomAll: The Pokémon will have one of its possible abilities.

Example:

```
[31] =
{
    .nature = NATURE_DOCILE,
    .ivs = 31,
    .hpEv = 128,
    .atkEv = 252,
    .spdEv = 128,
    .ball = BALL_TYPE_CHERISH_BALL,
    .ability = Ability_RandomAll,
},
```

In the above example, the nature for the spread is set to *Docile*, each stat has an IV of 31, the *HP* stat has 128 EVs, the *Attack* stat has 252 EVs, and the *Speed* stat has 128 EVs. All EV stat not referenced in this spread will be set to a default value of 0. Any Pokémon using this spread will also be released in a *Cherish Ball* and have any one of it possible abilities (including its hidden ability).

Battle Frontier
//TO WRITE

Pickup[2]

The items found by the ability Pickup can be modified in **src/Tables/Pickup_Items.c**. Modify *sPickupCommonItems* and *sPickupRareItems* to change the items that appear. **DO NOT** add any new items to the tables; only change the pre-existing items. The default values were chosen based on the following table (common is highlighted in blue, rare is highlighted in red):

Select from PC Hack

If SELECT_FROM_PC is defined, the player can select Pokémon 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 Pokémon by setting *Var 0x8003* to 1. For example, nicknaming a Pokémon in the PC:

```
#define pcSelect_StateTracker 0x203B7AC

#org @start
writebytetooffset 0x1 pcSelect_StateTracker
special 0x3C 'Select boxed mon, box stored to Var8000, slot to Var8001
waitstate
writebytetooffset 0x0 pcSelect_StateTracker
compare LASTRESULT 0x7F
if 0x1 goto @NothingSelected  'User cancelled out of the PC menu
bufferPokémon 0x0 0x8002
setvar 0x8003 0x1   'Data source is in the PC Box
special 0x7d
compare LASTRESULT 0x1
if 0x1 goto @traded
special 0x9e
waitstate
end
```

---

[2] Credits to Sagiri for the original code.

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 WildPokémonHeader gWildMonMorningHeaders[] =":

```
const struct WildPokemon gRoute1_LandMonsNight[] =
{
    {3, 3, PKMN_PIDGEY},
    {3, 3, PKMN_RATTATA},
    {3, 3, PKMN_PIDGEY},
    {3, 3, PKMN_RATTATA},
    {2, 2, PKMN_PIDGEY},
    {2, 2, PKMN_RATTATA},
    {3, 3, PKMN_PIDGEY},
    {3, 3, PKMN_RATTATA},
    {4, 4, PKMN_PIDGEY},
    {4, 4, PKMN_RATTATA},
    {5, 5, PKMN_PIDGEY},
    {4, 4, PKMN_RATTATA},
};

const struct WildPokemonInfo gRoute1_LandMonsInfoNight = {21, gRoute1_LandMonsNight};
```

^ 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":

```
const struct WildPokemon gRoute1_LandMonsNight[] =
{
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {2, 2, PKMN_HOOTHOOT},
    {2, 2, PKMN_RATTATA},
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {4, 4, PKMN_HOOTHOOT},
    {4, 4, PKMN_RATTATA},
    {5, 5, PKMN_HOOTHOOT},
    {4, 4, PKMN_RATTATA},
};

const struct WildPokemonInfo gRoute1_LandMonsInfoNight = {21, gRoute1_LandMonsNight};
```

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:

```
const struct WildPokemonHeader gWildMonNightHeaders[] =
{
    {
        .mapGroup = MAP_GROUP(ROUTE_1),
        .mapNum = MAP_NUM(ROUTE_1),
        .landMonsInfo = &gRoute1_LandMonsInfoNight,
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    },
    {

        .mapGroup = 0xFF,
        .mapNum = 0xFF,
        .landMonsInfo = NULL
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    }
};
```

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)

```
#define MAP_ROUTE_1 ((3 << 8) | 19)
```

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 to see how to structure those kinds of wild datasets.

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

```c
#include "..\\defines.h"

#ifndef UNBOUND //Modify this section

#define MAP_ROUTE_1 ((3 << 8) | 19)

const struct WildPokemon gRoute1_LandMonsNight[] =
{
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {2, 2, PKMN_HOOTHOOT},
    {2, 2, PKMN_RATTATA},
    {3, 3, PKMN_HOOTHOOT},
    {3, 3, PKMN_RATTATA},
    {4, 4, PKMN_HOOTHOOT},
    {4, 4, PKMN_RATTATA},
    {5, 5, PKMN_HOOTHOOT},
    {4, 4, PKMN_RATTATA},
};

const struct WildPokemonInfo gRoute1_LandMonsInfoNight = {21, gRoute1_LandMonsNight};

const struct WildPokemonHeader gWildMonMorningHeaders[] =
{
    {
        .mapGroup = 0xFF,
        .mapNum = 0xFF,
        .landMonsInfo = NULL,
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    }
};

const struct WildPokemonHeader gWildMonEveningHeaders[] =
{
    {
        .mapGroup = 0xFF,
        .mapNum = 0xFF,
        .landMonsInfo = NULL,
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    }
};

const struct WildPokemonHeader gWildMonNightHeaders[] =
{
    {
        .mapGroup = MAP_GROUP(ROUTE_1),
        .mapNum = MAP_NUM(ROUTE_1),
        .landMonsInfo = &gRoute1_LandMonsInfoNight,
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    },
    {
        .mapGroup = 0xFF,
        .mapNum = 0xFF,
        .landMonsInfo = NULL
        .waterMonsInfo = NULL,
        .rockSmashMonsInfo = NULL,
        .fishingMonsInfo = NULL,
    }
};

const struct SwarmData gSwarmTable[] =
{
    {
        .mapName = 0xFF,
        .species = 0xFFFF,
    },
};
```

Swarms

Day & Night System

A brand new DNS has been included in the engine. It features dynamic palette fading throughout the day, as well as options to allow windows to light up at night. Both of these options can be customized by editing **include/new/DNS.h**. Note that since this is a **.h** file, if you want your changes to compile, you will need to make a change to the file **src/DNS.c** so the compiler recompiles the DNS file. The change can be as simple as adding a whitespace character and then removing it. **Don't use the undo button to remove the change!** The change needs to be removed manually by using the backspace character so the editor saves the file with an updated timestamp.

Changing Which Palettes Are Faded

Open up the DNS.h file and search for OW_BG_PAL_0. Here is a listing of all palettes that can be faded. OW_BG_PAL_0 through OW_BG_PAL_15 represent the palettes of the background in the overworld. By default, OW BG palettes 0 - 11 are set to be faded, but 12 can be faded as well by making the following change:

```
#define OW_BG_PAL_12 FALSE  ⟶  #define OW_BG_PAL_12 TRUE
```

Similarly, whether any other palette is faded can be changed by changing its value from TRUE to FALSE or vice versa. It is **NOT** recommended to fade palettes 13-15 as these are the colours used for the menus and text boxes.

Other than the OW backgrounds, the OW_SPRITE_PAL represent the sprites in the overworld, the BBG_PAL defines represent the background in battle, and the B_SPRITE_PAL defines represent the sprites in battle.

Changing the Colours Faded Throughout the Day

In the same file is a table representing the actual fading colours (*gDNSNightFadingByTime*). Currently the table only has entries from 12 AM - 7:59 AM and 5 PM - 11:59 PM. The rest of the day no fading changes can be seen. If you would like to add fading for more time during the day, simply look for the line *Day has no fade* and start adding new entries there. For example, adding an entry for 8 AM - 8:59 AM:

```
[8]  = {
        {RGB(0, 24, 16), 1},     //8:00 AM
        {RGB(0, 24, 16), 1},     //8:10 AM
        {RGB(0, 24, 16), 1},     //8:20 AM
        {RGB(0, 24, 16), 1},     //8:30 AM
        {RGB(0, 24, 16), 1},     //8:40 AM
        {RGB(0, 24, 16), 1},     //8:50 AM
    },
```

The colour can change every ten minutes, so each of those colour indices represent a colour at that ten minute period.

Light Up Windows

There are currently two ways to handle light up windows. The first is more tedious. It involves leaving OW_BG_PAL_12 as FALSE (see *Changing Which Palettes Are Faded*) and make all tiles you'd like to light up use that palette. Then, use an on-entry script in Advance Map to do setmaptiles and place your light up windows only when the time is night. This can be an extreme annoyance and a time consumer, which is why the second, new method was developed.

This new method uses a table to fade certain palette colours if it nighttime.

1. To start off, find the offset of the tileset containing the palette you wish to fade. For this example, I'll be fading the windows of the player's door in *Palette Town*. Looking in Advance Map, the player's door uses tiles from *Tileset 1*, which has an offset of 0x82D4AAC.
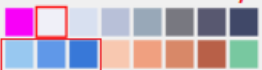
2. Now open the tile viewer and determine which colours are the ones you wish to fade. For my example, I've determined that the player's door uses palette 8, and the colours used are in indices 8, 9, and 10:

3. Now determine the colours you wish to change them to. It's okay to play around with these colours in Advance Map. Hitting *Apply* won't save anything permanently unless you exit out of the block editor and choose to save. For my example, I could going to change the window to a yellowy colour:
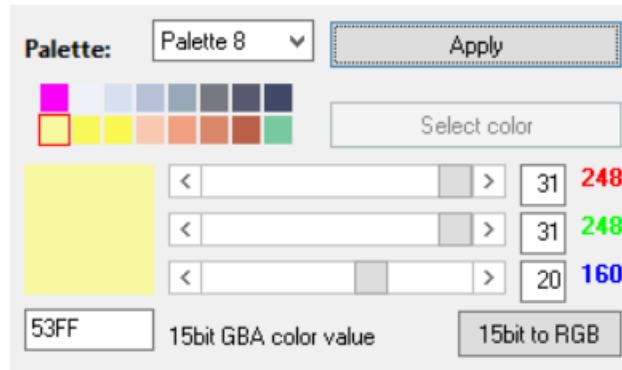
These colours can be represented by the RGB values of (31, 31, 20), (31, 31, 11), and (31, 31, 10).

4. Putting it all together now:

```
{ //Palette Town - Player's Door
    .tilesetPointer = 0x82D4AAC, //Tileset 1
    .paletteNumToFade = 8,
    .paletteIndicesToFade =
    {
        {8,  RGB(31, 31, 20)},
        {9,  RGB(31, 31, 11)},
        {10, RGB(31, 31, 10)},
        TILESET_PAL_FADE_END
    },
},
```

In the image *.tilesetPointer* is set to the offset of the tileset found in step 1, *.paletteNumToFade* is set to *8* representing the 8th palette. *.paletteIndicesToFade* has entries for index 8, 9, and 10, each with the corresponding colour they should be faded. After all that, is the line TILESET_PAL_FADE_END. **DO NOT FORGET THIS LINE!**

Some more examples have been left in the file to help guide you further if you need it.

Other Features Included

Save Expansion
//TO WRITE

Updated Flutes
The Black Flute and 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*.

Dex Nav[3]
A simplified Dex Nav 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 Pokémon they have caught/seen on the current map, and either press *Select* or *A* on a given Pokémon to search for it on the map. *Select* will save the Pokémon information to VAR_DEXNAV and allow the player to search for that Pokémon via *Select* from the overworld.

Dynamic Overworld Palettes[4]
Overworld sprites are now loaded in dynamically, allowing for more freedom with creating new overworld sprite palettes.

Ability Pop-Ups[5]
Whenever a Pokémon's ability activates in battle, a pop-up will appear showing the ability akin to Gen 5+ games.

Hidden Abilities[6]
Hidden abilities have been implemented akin

---

[3] Credits to FBI for this feature.
[4] Credits to Navenatox for this feature.
[5] Credits to DizzyEgg for this feature.
[6] Credits to azurile13 for this feature.

Expanded Trainer Class Names

Pokedex Screen Stats

The boring old size comparison when viewing a Pokémon's pokedex data has been replaced with a routine by DoesntKnowHowToPlay and Squeetz that displays the Pokémon's stats and abilities instead.

Turbo Boost

A pipelined routine has been written to replace the old main loop with one that now allows the game to run at over 1000% speed using the fast-forward button.

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 defined, remember to keep track of the Pokémon source from *Var 0x8003* before calling specials that manipulate Pokémon attributes.

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

Pokémon Specials

The following specials check or change Pokémon attributes. If SELECT_FROM_PC is defined, *Var 0x8003* will allow you to check/change data from PC boxed Pokémon if it is set to 1. Otherwise it will check/change from a party Pokémon.

Special 0x7 – EV/Contest Stat Checker

Details: Checks a party/boxed Pokémon'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 var.
Example Script:

```
setvar 0x8003 0x0  'Select from party
setvar 0x8004 0x3  '4th Pokémon in party
special2 LAST_RESULT 0x7
buffernumber 0x0 LAST_RESULT  'Buffer EV stat into [buffer1]
```

Special 0x8 – Pokémon IV Checker

Details: Checks a party/boxed Pokémon'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 var.

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 LAST_RESULT 0x7F  'Player exited without selecting
if 0x1 goto @DidNotSelect
setvar 0x8003 0x1  'From boxed mon
setvar 0x8005 0x1  'Check attack IV
special2 LAST_RESULT 0x8
buffernumber 0x0 LAST_RESULT  'Buffer attack IV to [buffer1]
```


Special 0x9 – Pokémon Ribbon Checker

Details: Checks a Pokémon's ribbons. The first 5 ribbons have values from 1 - 4 for Normal, Super, Hyper, and Master rank, so be sure to reference the correct bit value(s). See this page 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)	0x1
	Super (2) = bit 2 (0010)	0x2
	Hyper (3) = bits 1,2 (0011)	0x3
	Master (4) = bit 3 (0100)	0x4
Beauty Ribbons	Normal (1) = bit 4	0x8
	Super (2) = bit 5	0x10
	Hyper (3) = bit 4,5	0x18
	Master (4) = bit 6	0x20
Cute Ribbons	Normal (1) = bit 7	0x40
	Super (2) = bit 8	0x80
	Hyper (3) = bit 7,8	0xC0

	Master (4) = bit 9	0x100
Smart Ribbons	Normal (1) = bit 10	0x200
	Super (2) = bit 11	0x400
	Hyper (3) = bit 10,11	0x600
	Master (4) = bit 12	0x800
Tough Ribbons	Normal (1) = bit 13	0x1000
	Super (2) = bit 14	0x2000
	Hyper (3) = bit 13,14	0x3000
	Master (4) = bit 15	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 LAST_RESULT 0x9
        buffernumber 0x0 LAST_RESULT
        compare LAST_RESULT 0x1
        if 0x1 goto @IsAChampion
        'Else, Pokémon was not in the hall of fame

Special 0xA – Pokerus Timer Checker
Details: Checks the Pokérus virus timer on a Pokémon.
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: Pokérus time left to given var.

Example Script:
>   setvar 0x8003 0x0 'From party
>   setvar 0x8004 0x0 'First party Pokémon
>   special2 LAST_RESULT 0xA
>   buffernumber 0x0 LAST_RESULT 'Pokérus 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 – Poké Ball Checker
Details: Check the Poké Ball type of a Pokémon. The ball Ids can be found in **include/new/catching.h**.
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: Poké Ball type to given var.
Example Script:
>   writebytetooffset 0x1 0x0203B7AC 'Select from PC hack
>   Special 0x3C   'Store box/slot into vars 0x8000, 0x8001, respectively
>   waitstate
>   Compare LAST_RESULT 0x7F
>   If 0x1 goto @didNotSelect 'Player cancelled without selection
>   setvar 0x8003 0x1 'From box
>   special2 LAST_RESULT 0xB
>   buffernumber 0x1 LAST_RESULT '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 var.
Example Script:
>   setvar 0x8003 0x0 'From party
>   setvar 0x8004 0x5 'Last party mon
>   special2 LAST_RESULT 0xC
>   buffernumber 0x0 LAST_RESULT 'Buffer capture location to [buffer1]

Special 0xD – Happiness Checker

Details: Check the number of happiness points for a Pokémon.
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 var.
Example Script:

```
setvar 0x8003 0x0 'From party
setvar 0x8004 0x2 '3rd Pokémon
special2 LAST_RESULT 0xD
buffernumber 0x0 LAST_RESULT 'Buffer happiness to [buffer1]
compare LAST_RESULT 255
if 0x1 goto @maxedHappiness
```

Special 0xE – Hold Item Checker

Details: Check hold item value of a Pokémon.
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 to given var.
Example Script:

```
setvar 0x8003 0x0 'From party
setvar 0x8004 0x3 '4th Pokémon
special2 LAST_RESULT 0xE
buffernumber 0x0 LAST_RESULT 'Buffer to [buffer1]
```

Special 0xF – Add/Subtract to EVs

Details: Add or subtract values to Pokémon 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 Pokémon
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 Pokémon. 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 0 and 31 (0x1F).

Var 0x8006: IV value to set.

Returns: Nothing.

Example Script:

    setvar 0x8003 0x0 'From party
    setvar 0x8004 0x0 'First Pokémon
    setvar 0x8005 0x0 'HP IV
    setvar 0x8006 31 'Value to set
    special 0x10   'Maximize first party Pokémon's HP IV

Special 0x11 – Set Ribbons

Details: Set or clear a Pokémon'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 Pokémon
    setvar 0x8005 0x3 '3rd ribbon bit = Master Rank on Cool Ribbon
    Special 0x11 'Set the ribbon

Special 0x12 – Set Pokérus

Details: Set a Pokérus timer of a Pokémon.

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 Pokémon's Pokérus.
    special 0x12

Special 0x13 – Change Happiness

Details: Add or subtract to a Pokémon'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.
Example Script:

setvar 0x8003 0x0 'From party
setvar 0x8004 0x1 '2nd party Pokémon
setvar 0x8005 0x0150 'Subtracting 0x50, or 80 friendship points
special 0x13

Special 0x14 – Change Pokeball

Details: Set the ball type of a Pokémon.
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 Pokémon
setvar 0x8005 0x1 'Set to Master Ball
special 0x14

Special 0x15 – Change Hold Item

Details: Set the hold item of a Pokémon.
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 Pokémon 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 LAST_RESULT 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 LAST_RESULT 1
If 0x1 goto @Failed
```

Special 0x16 – Change Species

Details: Change the species of a Pokémon.
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.
Example Script:

```
setvar 0x8003 0x0 'From party
setvar 0x8004 0x0 'First Pokémon
setvar 0x8005 PKMN_CHARMANDER 'Set to Charmander
special 0x16
```

Special 0x17 – Change Attacks

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

> *Var 0x8004:* Pokémon 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 Pokémon from the menu, store slot to Var 0x8004
waitstate
compare LAST_RESULT 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 Pokémon.
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 var.
Example Script:

> setvar 0x8003 0x0 'From party
> setvar 0x8004 0x0 'Check first Pokémon
> special2 LAST_RESULT 0x18
> compare LAST_RESULT PKMN_RATTATA 'Check if first Pokémon if Rattata
> If 0x0 goto @NotCorrect

Special 0x19 – Check Attack PP

Details: Check a Pokémon 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 var.
Example Script:

> setvar 0x8003 0x0 'From Party
> setvar 0x8004 0x0 'First Pokémon
> setvar 0x8005 0x0 'First move
> special2 LAST_RESULT 0x19
> buffernumber 0x0 LAST_RESULT
> compare LAST_RESULT 0x0
> if 0x1 goto @NoPPLeft

Party Specials

Special 0x62 – Erase Pokémon
Details: Erase a Pokémon from your party, or the entire party.

Input:
   Var 0x8004: Slot to erase (0xF for entire party).
Returns: Nothing.
Example Script:
   setvar 0x8004 0xF 'Erase entire party
   Special 0x62

Special 0x63 – Status Checker
Details: Check the primary status of a Pokémon.
Input:
   Var 0x8004: Pokémon Slot
Returns: Status inflicted to given var.

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

Example Script:
   setvar 0x8004 0x0 'First party Pokémon
   special2 0x8004 0x63 'Get statuses to var 0x8004
   setvar 0x8005 0x80 'Badly poisoned
   special2 LAST_RESULT 0x42 'Var 0x8004 & Var 0x8005
   compare LAST_RESULT 0x1 '& will return 1 if the Pokémon has this status
   If 0x1 goto @BadlyPoisoned

Special 0x64 – Status Inducer
Details: Inflict a primary status on a party Pokémon.
Input:

Var 0x8004: Pokémon slot, or 0xF for entire party.

Var 0x8005: Status(es) to induce (see Special 0x63 for values).

Var 0x8006: 1 if status should only be given to Pokémon that can be afflicted with it (ie. No paralysis on Electric-types), 0 otherwise.

Returns: Nothing.
Example Script:

setvar 0x8004 0xF 'First party Pokémon

setvar 0x8005 0x20 'Freeze entire party

setvar 0x8006 0x1 'Don't freeze Ice-types of Pokémon with Magma Armor

special 0x64

Special 0x65 – Check Pokémon HP
Details: Check the amount of HP remaining for a party Pokémon.
Input:

Var 0x8004: Pokémon slot.

Returns: HP to given var.
Example Script:

setvar 0x8004 0x0 'First Pokémon

special2 LAST_RESULT 0x65

compare LAST_RESULT 0x0

if 0x1 goto @DeadPoke

Special 0x66 – Inflict Party Damage
Details: Inflict damage on/heal a Pokémon, or entire party.
Input:

Var 0x8004: Pokémon 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

Special 0x67 – Generate Random Battle Tower Team
Details: Generates a random Battle Tower ready team using BATTLE_TOWER_POKE_LEVEL. Teams are generated based on Pokémon found in **src/Tables/Frontier_Spreads.c**
Input: Technically BATTLE_TOWER_POKE_LEVEL.
Returns: Nothing.
Example Script: special 0x67

Key Specials

Special 0x2B – Check AB Buttons

Details: Check if *A* or *B* has been pressed
Input: Nothing.
Returns: To the given var:

0x0: Neither pressed.
0x1: *A* was pressed.
0x2: *B* was pressed.
0x3: Both *A* & *B* were pressed.

Example Script:

```
#org @Loop
special2 LAST_RESULT 0x2B
compare LAST_RESULT 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 var:

0x0: No direction is pressed.
0x1: *Up* is pressed.
0x2: *Left* is pressed.
0x3: *Down* is pressed.
0x4: *Right* is pressed.
0x5: *Up-left* is pressed.
0x6: *Up-right* is pressed.
0x7: *Down-left* is pressed.
0x8: *Down-right* is pressed.

Example Script:

```
#org @start
special2 LAST_RESULT 0x2C
compare LAST_RESULT 0x3 'Down pressed
if 0x1 goto @PressedDown
'etc
```

Special 0x2D – Check Start/Select

Details: Check if *Start*/*Select* are pressed
Input: Nothing
Returns: To given var:

0x0: Nothing pressed.
0x1: *Select* pressed.
0x2: *Start* pressed.
0x3: Both *Start* & *Select* pressed.

Example Script:

        special2 LAST_RESULT 0x2D
        compare LAST_RESULT 0x2 'Start pressed
        If 0x1 goto @PressedStart

Special 0x2E – Check L/R

Details: Check if *L/R* are pressed.
Input: Nothing.
Returns: To given var:

        0x0: Nothing pressed.
        0x1: *R* pressed.
        0x2: *L* pressed.
        0x3: *L* & *R* pressed.

Example Script:

        special2 LAST_RESULT 0x2E
        compare LAST_RESULT 0x1 'R pressed
        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 var:

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 LAST_RESULT 0x2F
        Compare LAST_RESULT 0x1 'A pressed
        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) (see *Special 0x2F* for bits).

Var 0x8005: Number of times to press it.

Returns: Nothing.

Example Script:

```
setvar 0x8004 0x1  'A-button
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 var holding value.

Var 0x8005: Second value or var holding value.

Returns:

Var 0x8004 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 LAST_RESULT 0x3E
buffernumber 0x0 0x8004 'Addition result in [buffer1]
compare LAST_RESULT 0x1
if 0x1 goto @overflow

Special 0x3F – Subtract Variables
Details: Subtract the values inside two variables.
Input:

Var 0x8004: First value or var holding value.

Var 0x8005: Second value or var holding value.

Returns:

Var 0x8004 or Variable Inside: Difference of *Var 0x8004 - Var 0x8005*.

Given Var: 1 if difference underflows 0x0, 0 otherwise.

Example Script:

setvar 0x8004 0x4059
setvar 0x8005 0x4050
special2 LAST_RESULT 0x3F '[var 0x4059] = [var 0x4059] – [var 0x4050]
Compare LAST_RESULT 0x1
If 0x1 goto @overflowed '[var 0x4059] > [var 0x4050]

Special 0x40 – Multiply Variables
Details: Multiply the values of two variables together.
Input:

Var 0x8004: First value or var holding value.

Var 0x8005: Second value or var 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 LAST_RESULT 0x40 '400*200 = 80000 = 0x13880 = (0xFFFF) + 0x3881
buffernumber 0x0 0x8004 'Value will be 0xFFFF, LAST_RESULT 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 var.
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   'Var8004 = [0000 0100] | [0001 0000] = 0001 0100 = 0x14
```

Special 0x44 – XOR Variables

Details: Bitwise XOR 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 = [0001 0010] ^ [0001 1000] = 0000 1010 = 0xA
```

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 0x58 – Buffer Swarm Text
Details: Buffers the map name where there is currently a swarm to *buffer1* and the species name where there is currently a swarm to *buffer2*.
Input: None.
Returns: Nothing.
Example Script:
> special 0x56
> msgbox @Saw 0x6 '[buffer1]! They said there's\na whole bunch of [buffer2] there!

Special 0x59 – Buffer Species Roaming Text
Details: Buffers the map name where the given roamer can be found to buffer1, and the species name of the roamer to buffer2.
Input:

Var 0x8000: Species

Returns: To given var 0 if the requested species isn't found roaming. 1 otherwise.

Example Script:

```
setvar 0x8000 PKMN_CHARMANDER 'Charmander should roam
setvar 0x8001 25 'Level 25
setvar 0x8002 0x1 'Can roam on land
setvar 0x8003 0x0 'Cannot roam on water
special 0x129 'Create roaming Pokemon
compare LASTRESULT 0x0
if 0x1 goto @TooManyRoamers
setvar 0x8000 PKMN_CHARMANDER 'Find map where Charmander is roaming
special2 LASTRESULT 0x57 'Buffer roaming text
compare LASTRESULT 0x0
if 0x1 goto @NotRoaming
msgbox @Saw 0x6 '[buffer1]! They said a\n[buffer2] appeared there!
```

Special 0x5A – Wild Data Switch

Details:
Input:
Returns:
Example Script:

Special 0x5B – Cancel Wild Data Switch

Details:
Input:
Returns:
Example Script:

Special 0x97 – Random Grass Battle

Details: Initiate a grass battle with a random Pokémon 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:

```
#org @start
special 0x8F   'The trainer position special
special2 LAST_RESULT 0x7F   'Returns 1 if ground battles, 2 if water battle
compare LAST_RESULT 0x1
if 0x1 goto @grass
compare LAST_RESULT 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 Pokémon, 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_CHARIZARD
    setvar 0x8005 100
    setvar 0x8006 ITEM_LEFTOVERS
    Special 0x156
```

Special 0xAC - Load Second Trainer Defeat Message
Details: If a battle against two opponents is being started from a flag, this special will load in the defeat text for the second trainer.
Input:
    Loadpointer 0x0: Pointer to defeat text.
Returns: Nothing
Example Script: See here.

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.
Example Script: special 0x46

Special 0x47 – Pause Timer
Details: Pauses the already-started timer.
Input: None.
Returns: Nothing.
Example Script: special 0x47

Special 0x48 – Resume Timer
Details: Resume a paused timer.
Input: None.
Returns: Nothing.
Example Script: special 0x48

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 var.
Example Script: special2 LAST_RESULT 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 LAST_RESULT 0x4A

Special 0x4B – Stop and Update Playtime
Details: Stop the timer and update playtime value.
Input: None.
Returns: Nothing.
Example Script: special 0x4B

Special 0x4C – Update Playtime
Details: Update the playtime. This is meant for functions that take a while to process that cause delay in playtime.
Input: None.
Returns: Nothing.
Example Script: special 0x4C

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 LAST_RESULT 0x4D
     Compare LAST_RESULT 0x1 'Is timer >=
     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.
Example Script: special 0x4E

Special 0x4F – Start Timer at a Time
Details: Restart the timer at the value stored with Special 0x4E.
Input: None.
Returns: Nothing.
Example Script: special 0x4F

Special 0x50 – Store Timer Value to Variable
Details: Store the timer value stored from Special 0x4E.
Input: None.
Returns: Nothing.
     Var 0x8006: Timer value.
Example Script: special 0x50

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.
Example Script:
     setvar 0x8006 200
     special 0x61

Safari Specials

Special 0x86 – Get Safari Balls
Details: Check Safari Ball quantity.
Input: None.
Returns:

> *Var 0x8004:* Normal Safari ball number.
> *Var 0x8005:* Extra ball slot number.
> *Given Var:* Combined number.

Example Script:

```
special2 LAST_RESULT 0x86
buffernumber 0x0 0x8004   'Normal number
buffernumber 0x1 0x8005   'Extra balls
buffernumber 0x2 LAST_RESULT   '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 LAST_RESULT 0x88
compare LAST_RESULT 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 LAST_RESULT 0x88   'Get current pedometer
Compare LAST_RESULT 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 var.
Example Script:

getplayerpos 0x8004 0x8005 'Player's current position
addvar 0x8004 0x2   'Check tile 2 steps to the right of the player
special2 LAST_RESULT 0x7E
compare LAST_RESULT 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 LAST_RESULT 0x7F   'Get tile attributes on player's current tile

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 JPAN's 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 LAST_RESULT 0x8A 'Get number of steps player has walked
> buffernumber 0x0 LAST_RESULT

PC Selection Specials

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

Special 0x1A – Store/Return Party Pokémon Data
Details: Save or Return party/boxed Pokémon 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 LAST_RESULT for success/failure, respectively
Script Example:

setvar 0x8002 0 'Store
setvar 0x8005 0 'Save first party mon data
Special 0x1A   'First party Pokémon data now in Enemy data slot 5

Special 0x1B – Swap Party/Boxed Pokémon 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 LAST_RESULT 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 'Select boxed mon, box stored to Var8000, slot to Var8001
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 Pokémon'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 Pokémon
Details: Check if Pokémon 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 Pokémon
Details: Nickname a Pokémon.
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 Pokémon

```
writebytetooffset 0x1 0x0203B7AC
special 0x3C 'Select boxed mon, box stored to Var8000, slot to Var8001
waitstate
setvar 0x8003 0x1
special 0x7C 'Buffer nickname
Msgbox @AskNickname 0x5  'Nickname [buffer1]?
Compare LAST_RESULT 0x1
If 0x0 goto @Nope
Special 0x7D 'Check traded mon
Compare LAST_RESULT 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 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 'See note below
       compare LAST_RESULT 0x0
       if 0x1 goto @selectedOption1
       compare LAST_RESULT 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 Pokémon'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 Pokémon

Details: Play the measure Pokémon game.
Input:

Var 0x8004: Party slot of Pokémon.
Var 0x8005: Holds the variable that stores measurements.
Var 0x8006: Species to evaluate.

Returns: To the given var:

1: Pokémon is not of selected type.
2: Pokémon is smaller.
3: Pokémon is bigger, also stores biggest value in variable in *Var 0x8005.*
4: Sizes are equal.

Special 0x9C – Old Man Battle

Details: Initiate an old man battle with a specific Pokémon species and level.
Input:

Var 0x8004: Species.
Var 0x8005: Level.

Returns: Nothing.
Example Script:

        setvar 0x8004 PKMN_BEEDRILL
        setvar 0x8005 50
        special 0x9C
        waitstate

Special 0x18B – Show Fossil Image

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

Name	Bytes	Description
Pointer Table - Defined in **src/config.h**		
Table can also be generated by the engine in **src/script_specials.c**. Search for *gFossilImageTable*.		
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 'At (0x0)
   setvar 0x8006 0x0 'At (0x0)
   Special 0x18B

Time-Based Specials

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                                          #org @AlreadyDid
setvar 0x8000 DAILY_EVENT_VAR                        setvar 0x8000 DAILY_EVENT_VAR
setvar 0x8001 0x0 'Don't set daily event var to done setvar 0x8001 0x0 'Minute difference
special2 LASTRESULT SP_DAILY_EVENT                   special2 LASTRESULT  SP_GET_TIME_DIFFERENCE
compare LASTRESULT 0x0                               buffernumber 0x0 LASTRESULT
if == goto @AlreadyDid                               msgbox @already '"\v\h02 minutes ago I did it."
setvar 0x8000 DAILY_EVENT_VAR                        callstd MSG_FACEPLAYER
special SP_UPDATE_TIME_IN_VARS                       end
msgbox @havenot '"I have not done 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