BULBAPEDIA

Save data structure (Generation I)

From Bulbapedia, the community-driven Pokémon encyclopedia.

This article is incomplete.
Please feel free to edit this article to add missing information and complete it.

This article does not yet meet the quality standards of Bulbapedia.
Please feel free to edit this article to make it conform to Bulbapedia norms and conventions.

The **save data structure** for Generation I is stored in the cartridge's volatile battery-backed RAM chip (SRAM), or as a ".sav" file by most emulators. The structure consists of 32 KB of data divided between 4 banks each 8KB, or 0x2000, and overall contains potentially hundreds of variables, though there are quite a few areas that are either completely left alone, only read from, or only written to. There are also a number of areas that are only used in certain game states that can't be saved to, including various runtime-only data.

Most areas can freely be written to with custom data or at the very least cleared out. Most of these areas will not be altered by the game unless the "Clear Save Dialog" is triggered on the title screen. Some sections of the data will load untouched into the in-game memory during gameplay and re-saved back on each save.

Contents

Sections of the data are protected with a simple integrity check using a checksum to detect data corruption in cases such as the power being lost during the saving process. PC boxes have additional checksums for each box's contents on the bank in addition to the normal whole-bank checksum.
The Hall of Fame is the only section of used data which doesn't have a checksum and furthermore lies on an unusual bank number because it's so large.

Save Data Structure

Banks Overview

Note: These values apply to the North American Pokémon Red, Blue, and Yellow games. They could be different for other releases.

The 32KiB save data, equal to 0x8000 bytes, is divided into 4 banks, each 8KiB in size, equal to 0x2000 bytes.

Bank	Offset	Name
0	0x0000	Scratch and Hall of Fame
1	0x2000	Main
2	0x4000	Boxes 1-6
3	0x6000	Boxes 7-12

Bank 0 - Scratch and Hall of Fame

Consists of 3 sprite decompression buffers which seem to contain misc or varying data and the Hall of Fame because of its size (4,800 bytes).

Bank 1 - Main

The primary bank for just about all data across the game, most is directly loaded into the in-game memory byte-by-byte. It contains hundreds of variables including a full byte-by-byte copy of the current PC box.

Bank 2 - Boxes 1-6

Storage for PC boxes 1-6

Bank 3 - Boxes 7-12

Storage for PC Boxes 7-12

Loading

When it comes to loading, the game only mainly focuses on bank #1, and much of it's contents are directly loaded byte-by-byte directly into WRAM at different addresses, as

such, different offsets can be applied to convert from WRAM to the save data and back. Those are provided where possible.

Other banks are loaded either on certain game states, on certain events, or as needed. Upon switching Pokémon boxes, for example, the contents of the new bank are copied to bank 1 from either bank 2 or 3 and events that use hall of fame data causes either one of or the latest Hall of Fame record to be loaded into WRAM.

Checksum

Used to validate the integrity of saved data.

The checksum in Generation I is only 8 bits and has a single copy of it. If this value is incorrect, the error message "The file data is destroyed!" will appear after the title screen, and the continue option will not appear in the menu.

The algorithm used to calculate the checksum is as follows:

- Initialize the checksum to 0
- For every byte from 0x2598 to 0x3522, inclusive, add its value to the checksum
- Invert the bits of the result

An equivalent way to achieve the same result is:

- Initialize the checksum to 255
- For every byte from 0x2598 to 0x3522, inclusive, subtract its value from the checksum

The checksum neatly validates the data it encompasses but there are several checksums for different sections of data in different banks. Those are all described below.

Bank 0

This is the Hall of Fame and scratch bank. It can essentially be regarded mostly as a misc bank with some large data, various scratch buffers used only occasionally during run-time, and lots and lots of unused space. None of the data in this bank is validated or checked for corruption.

The bank starts at address 0x0000 and is 0x2000 in size.

Offset	Size	Contents
0x0000	0x188	Sprite Buffer 0
0x0188	0x188	Sprite Buffer 1
0x0310	0x188	Sprite Buffer 2
0x0498	0x100	Unused
0x0598	0x12C0	Hall of Fame
0x1858	0x7A8	Unused

Scratch Buffers

During gameplay, the game sometimes uses these buffers as extra graphical buffers for misc stuff. As far as has been observed, they can safely be cleared, although they are used occasionally during gameplay so anything cleared or written to them will just be overwritten.

Unused

Unused here refers to data that the game completely does not touch in any way. Also none of the data is ever loaded from or saved to. The only way the game can ever be made to wipe or do anything to these areas is by invoking the hidden "Clear Save File" dialog manually yourself at the title screen.

Hall of Fame

Refer to the Hall of Fame data structure above which goes into all the detail. The Hall of Fame code is a bit messy and it's memory structure the same, memory load addresses can't be provided well.

Bank 1

This is the prime bank that all of the information is stored in that's ever needed or used in gameplay. Bank 2 and 3 really pull into data into Bank 1 when needed or are occasionally updated.

Furthermore most sections in this bank directly correspond byte-by-byte to their counterparts in in-game memory meaning you can apply a simple offset to translate the save data address to and from the in-game memory address. It also means unused areas in such sections will be loaded and available in-game and even saved back to save data.

As such, this is a mammoth of a bank and contains hundreds of variables.

The bank starts at address 0x2000 and is 0x2000 in size.

Offset	Size	Contents	Copied To	Diff
0x2000	0x598	Unused		
0x2598	0xB	Player Name	0xD158	0xABC0
0x25A3	0x789	Main Data	0xD2F7	0xAD54
0x2D2C	0x200	Sprite Data	0xC100	0x93D4
0x2F2C	0x194	Party Data	0xD163	0xA237
0x30C0	0x462	Current Box Data	0xDA80	0xA9C0
0x3522	0x1	Tileset Type	0xD0D4	0x9BB2
0x3523	0x1	Main Data Checksum		
0x3524	0xADC	Unused		

Unused

Unused here refers to data that the game completely does not touch in any way. Also none of the data is ever loaded from or saved to. The only way the game can ever be made to wipe or do anything to these areas is by invoking the hidden "Clear Save File" dialog manually yourself at the title screen.

Player Name

Player name, even though the game allocates space for 10 characters + Terminator, oddly enough, the in-game prompt only allows typing 7 possibly to fit within certain dialogue maybe.

Main Data

This area contains most of the entire memory state from WRAM and is copied byte-by-byte.

Sprite Data

This area contains detailed sprite information in 2 section for the map and is copied byte-by-byte. It's not the only sprite information, there are actually 2 other areas which contain more sprite information for the current map located in "Main Data"

Party Data

Contains information on the players current party and is copied byte-by-byte

Current box data

A byte-by-byte copy of the current PC box from either bank 2 or 3 and is byte-by-byte copied to memory on each load. In this way, the game only deals with 1 PC box in-memory at anytime and that PC box is copied here for easy and quick transfer back to memory on each load.

Tileset Type

The current tileset of where you are such as a specific place, store, dungeon, or just in the outside world.

Main Data Checksum

This checksum is calculated from Player Name through Tileset Type.

Main Data

Goes over all the elements inside of Main Data, one of the most comprehensive areas in the save data. To translate the offset to in-game memory and/or back simply use the difference number mentioned in the above table, 0xAD54.

Offset	Size	Contents
0x25A3	0x13	Pokédex Owned
0x25B6	0x13	Pokédex Seen
0x25C9	0x2A	Bag Items
0x25F3	0x3	Money
0x25F6	0xB	Rival Name
0x2601	0x1	Game Options
0x2602	0x1	Badges

0x2603	0x1	Padding
0x2604	0x1	Letter Delay
0x2605	0x2	Player ID
0x2607	0x1	Music ID
0x2608	0x1	Music Bank
0x2609	0x1	Contrast ID
0x260A	0x1	Current Map
0x260B	0x2	UL Corner Cur View Tile Block Map Ptr
0x260D	0x1	Y Coord
0x260E	0x1	X Coord
0x260F	0x1	Y Block Coord
0x2610	0x1	X Block Coord
0x2611	0x1	Last Map
0x2612	0x1	Unused
0x2613	0x1	Current Tileset
0x2614	0x1	Map Height Blocks
0x2615	0x1	Map Width Block
0x2616	0x2	Map Data Pointer
0x2618	0x2	Map Text Pointer
0x261A	0x2	Map Script Pointer
0x261C	0x1	Map Connections
0x261D	0xB	Map Connections North
0x2628	0xB	Map Connections South
0x2633	0xB	Map Connections West
0x263E	0xB	Map Connections East
0x2649	0xB	Sprite Set IDs
0x2654	0x1	Sprite Set ID
0x2655	0x4	Object Data Pointers Tmp
0x2659	0x1	Out of Bounds Tile
0x265A	0x1	Warp Count
0x265B	0x80	Warp Entries
0x26DB	0x1	Warp Destination ID
0x271C	0x1	Pikachu Friendship[Y]
0x275B	0x80	Padding
0x275C	0x1	Sign Count
0x275D	0x20	Sign Coords
0x277D	0x10	Sign Text IDs
0x278D	0x1	Sprite Count
0x278E	0x1	Y Offset since last special warp
0x278F	0x1	X Offset since last special warp
0x2790	0x20	Sprite Data
0x27B0	0x20	Sprite Extra Data
0x27D0	0x1	Map 2×2 Meta Height
0x27D1	0x1	Map 2×2 Meta Width

0x27D2	0x2	Map View VRAM Pointer
0x27D4	0x1	Player Movement Direction
0x27D5	0x1	Player Last Stop Direction
0x27D6	0x1	Player Direction
0x27D7	0x1	Tileset Bank
0x27D8	0x2	Tileset Block Pointer
0x27DA	0x2	Tileset GFX Pointer
0x27DC	0x2	Tileset Collision Pointer
0x27DE	0x3	Tileset Talking over Tiles
0x27E1	0x5	Tileset Grass Tiles
0x27E6	0x68	Box Items
0x284C	0x2	Current Box Number
0x284E	0x1	Hall of Fame Record Count
0x284F	0x1	Unused
0x2850	0x2	Slot Coins
0x2852	0x20	Missable Objects
0x2872	0x7	Padding
0x2879	0x1	Scratch
0x287A	0x22	Missable List
0x289C	0x100	Completed Scripts
0x299C	0xE	Owned Hidden Items
0x29AA	0x2	Owned Hidden Coins
0x29AC	0x1	Walking, Biking, or Surfing
0x29AD	0x10	Padding
0x29B7	0x2	Towns visited
0x29B9	0x2	Safari Steps
0x29BB	0x1	Fossil Item Given ID
0x29BC	0x3	Fossil Pokémon Result ID
0x29BF	0x1	Enemy Pokémon or Trainer Class
0x29C0	0x1	Player Jumping Y Screen Coords
0x29C1	0x1	Rival first partner Pokémon
0x29C2	0x1	Padding
0x29C3	0x1	Player first partner Pokémon
0x29C4	0x1	Boulder Sprite Index
0x29C5	0x1	Last Blackout Map
0x29C6	0x1	Destination Map
0x29C7	0x1	Unused
0x29C8	0x1	Tile in front of Boulder or Collision
0x29C9	0x1	Dungeon Warp Destination
0x29CA	0x1	Which Dungeon Warp Used
0x29CB	0x9	Unused
0x29D4	0x1	Various Flags 1
0x29D5	0x1	Padding
0x29D6	0x1	Defeated Gyms
0x29D7	0x1	Padding

0x29D8	0x1	Various Flags 2
0x29D9	0x1	Various Flags 3
0x29DA	0x1	Various Flags 4
0x29DB	0x1	Padding
0x29DC	0x1	Various Flags 5
0x29DD	0x1	Padding
0x29DE	0x1	Various Flags 6
0x29DF	0x1	Various Flags 7
0x29E0	0x2	Defeated Lorelei
0x29E2	0x1	Various Flags 8
0x29E3	0x2	In-Game Trades
0x29E5	0x1	Padding
0x29E7	0x1	Warped from Warp
0x29E8	0x1	Warped from Map
0x29E9	0x2	Padding
0x29EB	0x1	Card key door Y
0x29EC	0x1	Card key door X
0x29ED	0x2	Padding
0x29EF	0x1	First Trash Can Lock
0x29F0	0x1	Second Trash Can Lock
0x29F1	0x2	Padding
0x29F3	0x140	Completed Game Events
0x2B33 (U)	0x1	Grass Rate
0x2B33 (U)	0x1	Link Trainer
0x2B34	0xB	Grass Pokémon
0x2B3F (U)	0x9	Grass Pokémon
0x293F (U)	0x9	Link Data
0x2B48	0x1	Enemy Party Count
0x2B49	0x7	Enemy Party Pokémon
0x2B50 (U)	0x1	Water Rate
0x2B51 (U)	0x1	Water Pokémon
0x2B50 (U)	0x1	Enemy Partial Party Data
0x2CDC	0x2	Trainer Header Pointer
0x2CDE	0x6	Padding
0x2CE4	0x1	Opponent ID after wrong answer
0x2CE5	0x1	Current Map Script
0x2CE6	0x7	Padding
0x2CED	0x1	Play Time Hours
0x2CEE	0x1	Play Time Maxed
0x2CEF	0x1	Play Time Minutes
0x2CF0	0x1	Play Time Seconds
0x2CF1	0x1	Play Time Frames
0x2CF2	0x1	Safari Game Over
0x2CF3	0x1	Safari Ball Count
0x2CF4	0x1	Daycare In-Use

0x2CF5	0xB	Daycare Pokémon Name
0x2D00	0xB	Daycare Original Trainer
0x2D0B	0x1	Daycare Pokémon

Padding

Padding is essentially extra space assigned to an area in memory. In other words, the space is larger than the variables makeup and it could be at the start, end, middle, or some combination. It's unknown truly if the padding is ever used, there are certainly many ways for it to be used such as in various game unions with shared space. Therefore it could be considered possibly unused.

Pokédex Seen/Owned

Represents the specific Pokédex entries that have been either seen or owned during gameplay.

Pokémon are indexed by their usual Pokédex order, meaning the order is the same as in the National Pokédex. However, indexes begin counting at 0, rather than 1.

1 bit is used to represent whether a given Pokémon has been seen/owned. Bits are ordered within bytes from lowest to highest, starting with the first byte. Therefore, the exact bit can be extracted from the list using the following formula:

Bit = (Data[RoundDown(PokéNum / 8)] / 2 ^ (PokéNum Mod 8)) AND 1

Or in C-style code (shift occurs before other bitwise operations):

Bit = Data[PokéNum >> 3] >> (PokéNum & 7) & 1;

Example:

Let us say that we want to know whether #120 Staryu has been seen/owned:

- PokéNum becomes 119, since it is 0-based.
- The byte of the list in which bit 119 is located is = 119 / 8 = 14
- The bit within that byte is = 119 Mod 8 = 7
- Dividing the byte value by 2 ^ Bit, or shifting right by Bit, moves the bit to the least-significant position
- Performing a bitwise AND with 1 will remove all but the least-significant bit

Entry #152:

As the bit lists are packed into 19 bytes, there is actually space for 152 entries. The last entry, bit 7 of byte 18, does in fact represent an entry #152 in the Pokédex. However, it is simply a copy of Kangaskhan.

Bag Items

Follows Item List Data Structure with a capacity of 20.

Money

This is in Binary Coded Decimal, in other words, it's interpreted as decimal even if viewed as hexadecimal so don't use Hexadecimal A-F.

Represents how much money the character has. The figure is a 6-digit number, 2 digits per byte, encoded as binary-coded decimal, where each digit is allocated a full 4 bits.

This figure is still big-endian.

Essentially it's interpreted as decimal even if viewed as hexadecimal so don't use Hexadecimal A-F.

Rival Name

Rival name, even though the game allocates space for 10 characters + Terminator, oddly enough, the in-game prompt only allows typing 7 possibly to fit within certain dialogue maybe.

Game Options

Options are stored in one byte:

- bit 7 (MSB): battle effects ('1' for **No**, '0' for **Yes**)
- bit 6: battle style ('1' for **Set**, '0' for **Switch**)
- bit 4: sound ('0' for **Mono**, '1' for **Stereo**)
- bit 2-0: text speed ('001' for **Fast**, '011' for **Normal**, '101' for **Slow**)

In Pokémon Yellow:

- bit 5-4: sound ('00' for **Mono**, '01' for **Earphone1**, '10' for **Earphone2**, '11' for **Earphone3**)

Badges

The eight badges are stored on eight bits, one bit for each badge, being '1' is the badge is acquired, '0' otherwise.

From MSB to LSB, badges are in this order: Boulder, Cascade, Thunder, Rainbow, Soul, Marsh, Volcano, Earth.

Pikachu Friendship

Represents the friendship level of the partner Pikachu. For Red and Blue, this value is unused.

Letter Delay

```
bit 0: If 0, limit the delay to 1 frame. Note that this has no effect if
       the delay has been disabled entirely through bit 1 of this variable
       or bit 6 of Various Flags 5
bit 1: If 0, no delay.
```

Player ID

Randomly generated 16-bit id for the player and their Pokémon most used in Trades and whatnot.

Music ID

Which music or sound plays in this map

Music Bank

Which bank the music or sound is in

Contrast ID

The game contrast, really it's ever only used for the one cave that requires flash because without flash in the cave the contrast would be really bad.

“ Offset subtracted from FadePal4 to get the background and object palettes for the current map. Normally, it is 0. it is 6 when Flash is needed, causing FadePal2 to be used instead of FadePal4 ”

Current Map

Current Map ID the player was last at

UL Corner Cur View Tile Block Ptr

“pointer to the upper left corner of the current view in the tile block map”

X/Y Coord

Player X & Y Coordinates

X/Y Block Coord

Same as X & Y Coord but in blocks instead

Last Map

Last map the player visited

Current Tileset

Current tileset in map

Map Height/Width Blocks

Height and width of the map in blocks

Map Data/Text/Script Pointers

Pointers to various code for the current map to function as expected

Map Connections and NSEW Connections

A bit field describing the connections for the current map, are there any connecting maps to the map your on?

If there are, the details will be in the specific connection data mentioned in the connection data structure.

I must forewarn that maps and map connections can be a very complicated subject and often requiring some specific math formulas.

Sprite Set ID(s)

“ Sprite set for the current map (11 sprite picture ID's) ” followed by “sprite set ID for the current map”

Object Data Pointers Tmp

Unknown, most likely just a scratch 16-bit value for various object pointers.

Out of Bounds Tile

You have a map defined and then you have the are outside the map that will crash the game if moved to. This is the tile used to fill in all invalid areas of the current map.

“The tile shown outside the boundaries of the map”

Warp Count

Number of warps on the current map

Warp Entries

List of warp entries on the current map, specific format unknown

Warp Destination ID

Unknown but here's the description which talks about a particular value of this id.

“If $ff, the player's coordinates are not updated when entering the map”

Sign Count

Number of signs on the map, up to 16

Coordinates of the signs

Follows sign coordinate data structure, allows for up to 16 signs.

Sign Text Ids

Text ID of each sign, format is unknown but there are 16 bytes so it's assumed that each byte refers to a single text block.

Sprite Count

Number of sprites on the current map

X/Y Offset since last special warp

“These two variables track the X and Y offset in blocks from the last special warp used they don't seem to be used for anything”

Sprite Data/Extra

Various sprite data for up to 16 sprites, follows data structure on sprite data.

Map 2×2 Meta Height and Width

Map height and width in 2×2 meta tiles

Map View VRAM Pointer

“The address of the upper left corner of the visible portion of the BG tile map in VRAM”

Player Direction

“"moving" refers to both walking and changing facing direction without taking a step.”

Player Moving

“if the player is moving, the current direction. If the player is not moving, zero. Map scripts write to this in order to change the player's facing direction”

Plater Last Stop Direction

“The direction in which the player was moving before the player last stopped”

Player Direction

“If the player is moving, the current direction, if the player is not moving, the last the direction in which the player moved”

Tileset Bank

Bank current tileset can be found in

Tileset Pointers

Various pointers to how the tiles should be managed and treated in-game. Some extra descriptions are quoted below:

Tileset Block Pointer

“Maps blocks (4×4 tiles) to tiles”

Tileset Collision Pointer

“List of all walkable tiles”

Tileset Talking over Tiles

Tiles that can be talked over, as in, a tile between you and an NPC where you can still interact with the NPC even though your not directly in front of them. List of such valid tiles.

Tileset Grass Tiles

Unknown, presumably tiles where wild Pokémon can be found and which engages in a battle since more than just grass triggers a wild battle such as the mansion floor or cave floor.

Box Items

Items found in your PC's item box, follows the item box data structure and has a capacity of 50.

Current PC Box

Indicates which PC box is currently selected, minus 1. That is to say, box 1 is represented as 0, and box 12 is represented as 11.

```
bits 0-6: box number
bit 7: whether the player has changed boxes before
```

Hall of Fame Record Count

Hall many hall of fame victories you've had in this save file. The game can have up to 50 before it starts over and begins erasing the oldest records.

When loading the hall of fame record data it only pulls this record number temporarily into RAM and clears it out when done since it shares data space with important variables.

Slot Coins

How much coins player has left to gamble with

Represents how much coins the character has. The figure is a 4-digit number, 2 digits per byte, encoded as binary-coded decimal, where each digit is allocated a full 4 bits.

This figure is still big-endian.

Essentially it's interpreted as decimal even if viewed as hexadecimal so don't use Hexadecimal A-F.

Missable Objects

“bit array of missable objects. set = removed”

Scratch

“Temp copy of c1×2 (sprite facing/anim)”

Missable List

Follows the missable list data structure

Completed Scripts

Long array of completed scripts for all maps and even some NPCs, broken down further below.

Owned Hidden Items/Coins

Bit array of hidden items and coins player has found

Walking, Biking, or Surfing

Is the player walking, biking, or surfing currently?

Towns visited

Bit array of towns player has ever visited

Safari Steps

Number of steps for the safari zone taken so far

“Starts at 502”

Fossil Given and Result

“Item given to cinnabar lab” and “Pokémon that will result from the item”

Enemy Pokémon or Trainer Class

Shared space between 2 variables

1. Enemy Pokémon ID in Wild Pokémon Battle
2. Trainer class of Trainer Battle

Player Jumping Y Screen Coords

Unknown really, pertains to the sections of code that deal with the jumping mechanic.

Rival First Partner Pokémon

Pokémon Rival Picked

Player First Partner Pokémon

Pokémon Player Picked

Boulder Sprite Index

“Sprite index of the boulder the player is trying to push”

Last Blackout Map

Unknown

Destination Map

“Destination map (for certain types of special warps, not ordinary walking) ”

Tile in front of Boulder or Collision

“Used to store the tile in front of the boulder when trying to push a boulder. Also used to store the result of the collision check ($ff for a collision and $00 for no collision)”

Dungeon Warp Destination

“Destination map for dungeon warps”

Which dungeon warp used

“Which dungeon warp within the source map was used”

Various Flags 1

```
bit 0: using Strength outside of battle
bit 1: set by IsSurfingAllowed when surfing's allowed, but the caller resets it after checking the
result
bit 3: received Old Rod
bit 4: received Good Rod
bit 5: received Super Rod
bit 6: gave one of the Saffron guards a drink
bit 7: set by ItemUseCardKey, which is leftover code from a previous implementation of the Card Key
```

Defeated Gyms

“Redundant because it matches Earned Badges used to determine whether to show name on statue and in two NPC text scripts”

Various Flags 2

```
Bit 0: if not set, the 3 minimum steps between random battles have passed
Bit 1: prevent audio fade out
```

Various Flags 3

```
This variable is used for temporary flags and as the destination map when
warping to the Trade Center or Colosseum.

bit 0: sprite facing directions have been initialised in the Trade Center
bit 3: do scripted warp (used to warp back to Lavender Town from the top of the Pokémon tower)
bit 4: on a dungeon warp
bit 5: don't make NPCs face the player when spoken to
Bits 6 and 7 are set by scripts when starting major battles in the storyline,
but they do not appear to affect anything. Bit 6 is reset after all battles
and bit 7 is reset after trainer battles (but it's only set before trainer
battles anyway).
```

Various Flags 4

```
bit 0: the player has received Lapras in the Silph Co. building
bit 1: set in various places, but doesn't appear to have an effect
bit 2: the player has healed Pokémon at a Pokémon center at least once
bit 3: the player has a received a Pokémon from Prof. Oak
bit 4: disable battles
bit 5: set when a battle ends and when the player blacks out in the overworld due to poison
bit 6: using the link feature
bit 7: set if scripted NPC movement has been initialised
```

Various Flags 5

```
bit 0: NPC sprite being moved by script
bit 5: ignore joypad input
bit 6: print text with no delay between each letter
bit 7: set if joypad states are being simulated in the overworld or an NPC's movement is being scripted
```

Various Flags 6

```
bit 0: play time being counted
bit 1: remnant of debug mode? not set by the game code.
if it is set
1. skips most of Prof. Oak's speech, and uses NINTEN as the player's name and SONY as the rival's name
2. does not have the player start in floor two of the player's house (instead sending them to Last Map)
3. allows wild battles to be avoided by holding down B
bit 2: the target warp is a fly warp (bit 3 set or blacked out) or a dungeon warp (bit 4 set)
bit 3: used warp pad, escape rope, dig, teleport, or fly, so the target warp is a "fly warp"
bit 4: jumped into hole (Pokémon Mansion, Seafoam Islands, Victory Road) or went down waterfall (Seafoam
Islands), so the target warp is a "dungeon warp"
bit 5: currently being forced to ride bike (cycling road)
bit 6: map destination is Last Blackout Map (usually the last used Pokémon center, but could be the
player's house)
```

I will say don't get your hopes up with debug mode, it may work by setting it here given the timeline it loads the save data but the game may try to unset it since there are numerous measures in place particular when starting the game to ensure it's not set.

Various flags 7

```
bit 0: running a test battle
bit 1: prevent music from changing when entering new map
bit 2: skip the joypad check in CheckWarpsNoCollision (used for the forced warp down the waterfall in
the Seafoam Islands)
bit 3: trainer wants to battle
bit 4: use variable Current Map Script instead of the provided index for next frame's map script (used
to start battle when talking to trainers)
bit 7: used fly out of battle
```

Defeated Lorelei

“Bit 1: set when you beat Lorelei and reset in Indigo Plateau lobby, the game uses this to tell when Elite 4 events need to be reset”

Various Flags 8

```
bit 0: check if the player is standing on a door and make him walk down a step if so
bit 1: the player is currently stepping down from a door
```

```
bit 2: standing on a warp
bit 6: jumping down a ledge / fishing animation
bit 7: player sprite spinning due to spin tiles (Rocket hideout / Viridian Gym)
```

In-Game Trades

Bitset of completed in-game trades

Warped from warp/map

The warp or map you warped from

Card key door x/y

Unknown

First/Second Trash Can Lock

Lt. Surge Trash Can lock puzzle to battle surge, these are where the switches are located. It's unknown the data format or even if they'll regenerate upon reloading the game.

Completed Game Events

Bitfield of game events completed

Grass Rate

Wild Pokémon encounter rate for the "grass" areas of the game

Link Trainer

Unknown

Grass Pokémon

This is actually a 20 byte list of Wild Pokémon in order from most common to least common (Rarest) but the last 9 bytes cross-over into a shared space used for link battles using the game link since you're obviously not going to find wild battles there.

Link Data

Unknown

Enemy Party Count

How many Pokémon does the enemy have

Enemy Party Pokémon

Unknown since it's only 7 bytes. Perhaps the Species ID's

Water Rate

Chance of running into a wild Pokémon battle in the water areas of the current map.

Water Pokémon

Unknown since it's only 1 byte

Enemy partial party data

This follows the enemy partial party data structure

Trainer Header Pointer

Unknown

Opponent ID after wrong answer

“The trainer the player must face after getting a wrong answer in the Cinnabar gym quiz”

Current Map Script

“Index of current map script, mostly used as index for function pointer array and mostly copied from map-specific map script pointer and written back later.”

Playtime

Counts Hours, Minutes, Seconds, and Frames of playtime and includes a byte that determines whether the counter has hit the maximum time and therefore no longer counts. The maximum time countable is

- 255 Hours, 59 Minutes, 59 Seconds, and 59 Frames or
- 10 days, 15 hours, 59 minutes, 59 seconds, and 59 frames.

The maximum byte can be any number, if the game detects it's not zero it essentially disables the counter entirely. Internally the game sets it to 255 when the timer has reached the maximum count. Whatever value set here only effects enabling or disabling the playtime and is not calculated into the playtime.

Setting Minutes, Seconds, or Frames 60 or above will simply increment the next byte by 1 and reset the value. In other words setting seconds to 120 will only increment minutes by 1, not 2.

It's also important to note that the timer is always ticking even on the initial load/new game menu or when it's paused.

Safari Game Over

Safari Game over or not, the format and any further details are unknown

Safari Ball Count

Amount of Safari Balls the player has

Daycare

The daycare has 4 variables

- In-Use to determine if it's in use or not
- Pokémon Name following standard naming
- Original Trainer name - follow standard naming of the Original Trainer
- Pokémon - Pokémon stored in the daycare, just uses the Box Data format alone

Completed Scripts

The details aren't fully understood but for all areas of the game it covers the completed scripts and is possibly bit flags.

Offset	Size	Contents
0x289C	0x1	Oaks Lab
0x289D	0x2	Palette Town
0x289F	0x1	Rival's House
0x28A0	0x3	Viridian City
0x28A3	0x1	Pewter City
0x28A4	0x1	Route 3
0x28A5	0x2	Route 4
0x28A7	0x1	Viridian Gym
0x28A8	0x1	Pewter Gym
0x28A9	0x1	Cerulean Gym
0x28AA	0x1	Vermilion Gym
0x28AB	0x1	Celadon Gym
0x28AC	0x1	Route 6
0x28AD	0x1	Route 8
0x28AE	0x1	Route 24
0x28AF	0x1	Route 25
0x28B0	0x1	Route 9
0x28B1	0x1	Route 10
0x28B2	0x1	Mt. Moon 1
0x28B3	0x1	Mt. Moon 3
0x28B4	0x1	S.S. Anne 8
0x28B5	0x1	S.S. Anne 9
0x28B6	0x2	Route 22
0x28B8	0x1	Player's House 2
0x28B9	0x1	Viridian Market
0x28BA	0x1	Route 22 Gate
0x28BB	0x8	Cerulean City
0x28C3	0x1	S.S. Anne 5
0x28C4	0x1	Viridian Forest
0x28C5	0x1	Museum 1
0x28C6	0x1	Route 13
0x28C7	0x1	Route 14
0x28C8	0x1	Route 17
0x28C9	0x1	Route 19
0x28CA	0x1	Route 21

0x28CB	0x1	Safari Zone Entrance
0x28CC	0x1	Rock Tunnel 2
0x28CD	0x2	Rock Tunnel 1
0x28CF	0x1	Route 11
0x28D0	0x1	Route 12
0x28D1	0x1	Route 15
0x28D2	0x1	Route 16
0x28D3	0x1	Route 18
0x28D4	0x1	Route 20
0x28D5	0x1	S.S. Anne 10
0x28D6	0x1	Vermilion City
0x28D7	0x1	Pokémon tower 2
0x28D8	0x1	Pokémon tower 3
0x28D9	0x1	Pokémon tower 4
0x28DA	0x1	Pokémon tower 5
0x28DB	0x1	Pokémon tower 6
0x28DC	0x1	Pokémon tower 7
0x28DD	0x1	Rocket Hideout 1
0x28DE	0x1	Rocket Hideout 2
0x28DF	0x1	Rocket Hideout 3
0x28E0	0x2	Rocket Hideout 4
0x28E2	0x1	Route 6 Gate
0x28E3	0x2	Route 8 Gate
0x28E5	0x1	Cinnabar Island
0x28E6	0x2	Mansion 1
0x28E8	0x1	Mansion 2
0x28E9	0x1	Mansion 3
0x28EA	0x1	Mansion 4
0x28EC	0x2	Victory Road 3
0x28EE	0x1	Fighting Dojo
0x28EF	0x1	Silph Co 2
0x28F0	0x1	Silph Co 3
0x28F1	0x1	Silph Co 4
0x28F2	0x1	Silph Co 5
0x28F3	0x1	Silph Co 6
0x28F4	0x1	Silph Co 7
0x28F5	0x1	Silph Co 8
0x28F6	0x1	Silph Co 9
0x28F7	0x1	Hall of Fame Room
0x28F8	0x1	Rival
0x28F9	0x1	Lorelei
0x28FA	0x1	Bruno
0x28FB	0x1	Agatha
0x28FC	0x1	Unknown Dungeon 3
0x28FD	0x1	Victory Road 1

0x28FF	0x5	Lance
0x2904	0x1	Silph Co 10
0x2905	0x2	Silph Co 11
0x2907	0x1	Fuchsia Gym
0x2908	0x2	Saffron Gym
0x290A	0x1	Cinnabar Gym
0x290B	0x1	Celadon Game Corner
0x290C	0x1	Route 16 Gate
0x290D	0x1	Bill's House
0x290E	0x1	Route 5 Gate
0x290F (U)	0x2	Power Plant
0x290F (U)	0x2	Route 7 Gate
0x2911	0x1	S.S. Anne 2
0x2912	0x1	Seafoam Islands 4
0x2913	0x1	Route 23
0x2914	0x1	Seafoam Islands 5
0x2915	0x1	Route 18 Gate
0x2916	0x78	Padding
0x2964	0x56	Completed Scripts End Padding

Sprite Data

Covers sprite data on map. To translate the offset to in-game memory and/or back simply use the difference number mentioned in the above table, 0x93D4.

This goes into quite a bit more detail than some of the other sprite data in Main Data.

These follow the Sprite Data Structure

Offset	Size	Contents	Amount
0x2D2C	0x10	Player	
0x2D3C	0xF0	Sprite	15

These follow the Sprite Extra Data Structure

Offset	Size	Contents	Amount
0x2E2C	0x10	Player	
0x2E3C	0xF0	Sprite	15

Party Data

Trainers current party. To translate the offset to in-game memory and/or back simply use the difference number mentioned in the above table, 0xA237.

Basically refer to the full party data structure

Offset	Size	Contents

0x2F2C	0x194	Party Data

Current Box Data

a full byte-by-byte copy of the current box which is then loaded directly into memory every time for quick access. To translate the offset to in-game memory and/or back simply use the difference number mentioned in the above table, 0xA9C0.

Basically refer to the full box data structure

Offset	Size	Contents
0x30C0	0x462	Box Data

Normally, Pokémon are deposited and withdrawn from the Current Box data, which is within the checksum-validated region of the save data. When switching boxes, the data from the Current Box is copied to the corresponding PC Box data, then the data from the switched-to PC Box is transferred into the Current Box data.

Bank 2

This bank only contains boxes 1-6, when a box is selected its contents are copied and cached in bank 1. Think of bank 2 and 3 as warehouses where information is copied over to the more important areas when needed and only updated on save in case anything changed in the copied versions. Furthermore Bank 2 and 3 and completely 100% identical in structure.

Bank 2 and 3 have 2 different checksums, theres a whole-bank checksum similar to bank 1 that encompasses all used data on the bank and 6 individual checksums for the individual boxes data.

All Pokémon boxes follow the full box data structure

Bank 2 starts at address 0x4000 and is 0x2000 in size.

Offset	Size	Contents
0x4000	0x462	Pokémon Box 1
0x4462	0x462	Pokémon Box 2
0x48C4	0x462	Pokémon Box 3
0x4D26	0x462	Pokémon Box 4
0x5188	0x462	Pokémon Box 5
0x55EA	0x462	Pokémon Box 6
0x5A4C	0x1	All Checksums
0x5A4D	0x6	Individual Checksums
0x5A53	0x5AD	Unused

Unused

Unused here refers to data that the game completely does not touch in any way. Also none of the data is ever loaded from or saved to. The only way the game can ever be

made to wipe or do anything to these areas is by invoking the hidden "Clear Save File" dialog manually yourself at the title screen.

Bank 3

This bank only contains boxes 7-12, when a box is selected its contents are copied and cached in bank 1. Think of bank 2 and 3 as warehouses where information is copied over to the more important areas when needed and only updated on save in case anything changed in the copied versions. Furthermore Bank 2 and 3 and completely 100% identical in structure.

Bank 2 and 3 have 2 different checksums, theres a whole-bank checksum similar to bank 1 that encompasses all used data on the bank and 6 individual checksums for the individual boxes data.

All Pokémon boxes follow the full box data structure

Bank 3 starts at address 0x6000 and is 0x2000 in size.

Offset	Size	Contents
0x6000	0x462	Pokémon Box 7
0x6462	0x462	Pokémon Box 8
0x68C4	0x462	Pokémon Box 9
0x6D26	0x462	Pokémon Box 10
0x7188	0x462	Pokémon Box 11
0x75EA	0x462	Pokémon Box 12
0x7A4C	0x1	All Checksums
0x7A4D	0x6	Individual Checksums
0x7A53	0x5AD	Unused

Unused

Unused here refers to data that the game completely does not touch in any way. Also none of the data is ever loaded from or saved to. The only way the game can ever be made to wipe or do anything to these areas is by invoking the hidden "Clear Save File" dialog manually yourself at the title screen.

Data Types

There are a few notable data types found in the save data, Unless otherwise noted, integer values occupy the specified number of bytes, and are big-endian and either unsigned or two's complement.

Text

Text data is stored in a proprietary encoding. Fixed-length user-input strings are terminated with 0x50. If a fixed-length string is terminated before using its full capacity, the contents of the remaining space are ignored and can be considered unused.

Hall of Fame

The Hall of Fame has an interesting data structure because it's the only Pokémon data structure that's so stripped down.

The game allocates space for 50 Hall of Fame Records, each record contains 6 entries for each Pokémon and each Pokémon entry contains 16 bytes. In code, it only uses the first 13 bytes out of 16 so it's assumed the last 3 are unused or padding.

Hall of Fame Structure

Offset	Size	Contents	Amount
0x00	0x60	Hall of Fame Records	50

Hall of Fame Record Entry Structure

Offset	Size	Contents	Amount
0x00	0x10	Pokémon	6

Hall of Fame Pokémon Structure

Offset	Size	Contents	Amount
0x00	0x1	Species ID	
0x01	0x1	Level	
0x02	0xB	Pokémon Name	
0x02	0x3	Padding	

Lists

Lists have 3 parts to them, a count of list entries, the list entries themselves, and the list ending. If the list has a count of 0 then there won't be a list of entries and thus the list count will read 0 followed immediately by the terminator which is 0xFF

A list entry consists of 2 bytes, the item id and the amount of such item from 1-99.

List

Offset	Size	Contents	Amount
0x00	0x1	Count	
0x01	0x2	Entries	...20
0x01 or <0x01 + 2 * Count>	0x1	End (0xFF)	

List Entry

Offset	Size	Contents	Amount
0x00	0x1	Index	
0x01	0x1	Amount	

Index refers to the item's index number.

Amount refers to the amount of the item that the player possesses and has valid values from 1 to 99.

Missable Lists

There is a varion of the standard list format above for the missable lists, it's the same as above only instead of id and amount it's instead id and index. Furthermore there's no count byte and thus there may not be a terminator byte. The details aren't fully understood beyond the list structure and it's found in only 1 place in the code within bank 1.

Offset	Size	Contents	Amount
0x00	0x1	ID	
0x01	0x1	Index	

Map Connection Entry

A great deal of various map information is stored within the Save Data, in 1 place there exists connection data for North, South, East, and West connecting maps where applicable.

Note: Maps can be a very complicated subject most especially when it comes to connecting maps. Instead the data structure will simply be presented as-is.

Offset	Size	Contents	Amount
0x00	0x1	Map Pointer	
0x01	0x2	Strip Source	
0x03	0x2	Strip Destination	
0x05	0x1	Strip Width	
0x06	0x1	Connected Map Width	
0x07	0x1	Connected Map Y Align	
0x08	0x1	Connected Map X Align	
0x09	0x2	Connected Map View Pointer	

Sign Coordinates

Amongst the map information is a section to hold the locations of signs, this is the data format for that.

Offset	Size	Contents	Amount
0x00	0x1	Y	
0x01	0x1	X	

Sprite Data

The save data has an entire section dedicated to the current state of the sprites on screen. Furthermore there are a couple of extra areas for sprites on the map noted further below. Here is the 2 structures found in the sprite section.

Note: Sprite Data is also a pretty complicated topic, and like maps, the data structure itself will just be given here.

Basic Sprite Data

Offset	Size	Contents	Amount
0x00	0x1	Picture ID	
0x01	0x1	Movement Status	
0x02	0x1	Image Index	
0x03	0x1	Y Step Vector	
0x04	0x1	Y Pixels	
0x05	0x1	X Step Vector	
0x06	0x1	X Pixels	
0x07	0x1	Intra Animation Frame Counter	
0x08	0x1	Animation Frame Counter	
0x09	0x1	Facing Direction	
0x0A	0x6	Unused padding	

Extra Sprite Data

Offset	Size	Contents	Amount
0x00	0x1	Walk Animation Counter	
0x01	0x1	Unused padding	
0x02	0x1	Y Disp	
0x03	0x1	X Disp	
0x04	0x1	Map Y	
0x05	0x1	Map X	
0x06	0x1	Movement Byte	
0x07	0x1	Grass Priority	
0x08	0x1	Movement Delay	
0x09	0x5	Unused Padding	
0x0E	0x1	Image Base Offset	
0x0F	0x6	Unused padding	

Map Sprite Data

There are 2 kinds of sprite data structures the map uses, one for basic information and one for extended information. They're always side-by-side.

Sprite Basic Data Entry

Offset	Size	Contents	Amount
0x00	0x1	Movement	
0x01	0x1	Text ID	

Sprite Extended Data Entry

Offset	Size	Contents	Amount
0x00	0x1	Trainer Class or Item ID	
0x01	0x1	Trainer Set ID	

Pokémon

Pokémon data structures is quite large and comprehensive and there are actually 2 kinds of Pokémon data structures.

When a Pokémon is in a PC or in the Day Care (Since the Daycare follows the same PC Box data structure), it's given the base data structure which is 33 bytes and contains base information needed to calculate other information (To save space).

When you withdraw a Pokémon out of the PC that information is calculated and placed in an extended data structure that's a total of 44 bytes from the original 33 so it's immediately ready when needed. This calculated information gets destroyed when the Pokémon is placed back into the PC.

Lists of Pokémon in the save data follow the same basic format, but there are slight variations of the list that are found in the save data. These types differ in their maximum capacity and in the way they store the main Pokémon data.

1. The player's party and has a capacity of 6 and uses the full Pokémon data structure (with a size of 44 bytes).
2. A PC Box and has a capacity of 20 and uses a truncated version of the Pokémon data structure (only the first 33 bytes).
3. The Daycare is treated as a single Pokémon capacity box and thus contains only the Pokémon Data structure and none of the "list" aspects to it.

Full Box Data

Pokémon Box Data Structures is sometimes used alone such as at the Daycare however when dealing with a PC that can handle 20 Pokémon per box, there is some additional data around that.

Offset	Size	Contents	Amount
0x00	0x1	Pokémon Count	
0x01	0x1	Species ID	20
0x15	0x1	Unused Padding	
0x16	0x21	Pokémon Box Data	20
0x2AA	0xB	Original Trainer Names	20
0x386	0xB	Pokémon Names	20

Pokémon count This is the number of Pokémon entries actually being used in the list.

Species ID

This is a list of species indexes for each Pokémon in this list, 1 byte per index, with a `0xFF` terminator following the last used entry in the list.

This list of species is used by the party screen and the PC's Box management interface.

Pokémon Box Data This is the Pokémon data structure for each Pokémon in this list, 44 or 33 bytes per index depending on if the whole list is for the party or a PC Box.

OT names This is a list of text strings for the Original Trainer of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

Name

Pokémon Nicknames is any name given to the Pokémon, if the name perfectly matches up with the species name (case-sensitive) it will be treated as if it had no nickname.

This is a list of text strings for the name of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

A Pokémon's name is a nickname if it does not perfectly match the default name for a Pokémon (typically all uppercase) with any unused bytes of the entry's 11-byte capacity filled with `0x50` terminators.

Full Party Data

This is exactly the same as the Full Box data but formatted for a party of up to 6.

Offset	Size	Contents	Amount
0x00	0x1	Pokémon Count	
0x01	0x1	Species ID	6
0x7	0x1	Unused Padding	
0x8	0x2C	Pokémon Party Data	6
0x110	0xB	Original Trainer Names	6
0x152	0xB	Pokémon Names	6

Pokémon Party Data This is the Pokémon data structure for each Pokémon in this list, 44 or 33 bytes per index depending on if the whole list is for the party or a PC Box.

OT names This is a list of text strings for the Original Trainer of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

Name

Pokémon Nicknames is any name given to the Pokémon, if the name perfectly matches up with the species name (case-sensitive) it will be treated as if it had no nickname.

This is a list of text strings for the name of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

A Pokémon's name is a nickname if it does not perfectly match the default name for a Pokémon (typically all uppercase) with any unused bytes of the entry's 11-byte capacity filled with `0x50` terminators.

Partial Party Data

There does exist one area which features a stripped down full party data. It's mostly just missing much of the upper data.

Offset	Size	Contents	Amount
0x0	0x21	Pokémon Party Data	6
0xC6	0xB	Original Trainer Names	6
0x108	0xB	Pokémon Names	6

Pokémon Party Data This is the Pokémon data structure for each Pokémon in this list, 44 or 33 bytes per index depending on if the whole list is for the party or a PC Box.

OT names This is a list of text strings for the Original Trainer of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

Name

Pokémon Nicknames is any name given to the Pokémon, if the name perfectly matches up with the species name (case-sensitive) it will be treated as if it had no nickname.

This is a list of text strings for the name of each Pokémon in this list, 11 bytes per name. Each name can contain from 1 to 10 characters, terminated by `0x50`.

A Pokémon's name is a nickname if it does not perfectly match the default name for a Pokémon (typically all uppercase) with any unused bytes of the entry's 11-byte capacity filled with `0x50` terminators.

Item Box

The PC Box for items follows the same rules as a regular list only instead of 20 items, it's space allotted for 50 items with 50 being the maximum count.

Offset	Size	Contents	Amount
0x00	0x1	Count	
0x01	0x2	Entries	...50
0x01 or <0x01 + 2 * Count>	0x1	End (0xFF)	

Credits

Massive credit goes to the Pokémon Red/Blue source code found here (https://github.com/pret/pokered). Specifically 2 files greatly helped wram (https://github.com/pret/pokered/blob/master/ram/wram.asm) and sram (https://github.com/pret/pokered/blob/master/ram/sram.asm). But several source code files needed to be referenced in the project for better understanding.

Much of the descriptions were also used from there and those 2 files.

Permission was directly obtained from the team and project admins for these references.

Apart from educational references to the project and the references descriptions all other content is completely original.

Related articles

Data structure in the Pokémon games	
General	Character encoding
Generation I	Pokémon species • Pokémon • Poké Mart • Character encoding (Stadium) • Save
Generation II	Pokémon species • Pokémon • Trainer • Character encoding (Stadium • Korean) • Save
Generation III	Pokémon species (Evolution • Pokédex • Type chart) Pokémon (substructures) • Move • Contest • Contest move • Item Trainer Tower • Battle Frontier • Character encoding (GameCube) • Save
Generation IV	Pokémon species (Evolution • Learnsets) Pokémon • Save • Character encoding (Wii)
Generation V–present	Character encoding
Generation VIII	Save
TCG GB and GB2	Character encoding

This data structure article is part of **Project Games**, a Bulbapedia project that aims to write comprehensive articles on the Pokémon games.

Retrieved from "https://bulbapedia.bulbagarden.net/w/index.php?title=Save_data_structure_(Generation_I)&oldid=4231991"

This page was last edited on 1 February 2025, at 01:19.
Content is available under Attribution-NonCommercial-ShareAlike 2.5. (see Copyrights for details)