Menu Search Bulbapedia

BULBAPEDIA

Save data structure (Generation II)

From Bulbapedia, the community-driven Pokémon encyclopedia.

The **save data structure** for Generation II is stored in the cartridge's battery-backed RAM chip (SRAM), or as a ".sav" file by most emulators. The structure consists of 32 kiB of data, though not every byte is used. Emulators may append additional data for the purposes of maintaining real-time clock operations.

Two regions of the save data have their integrity validated via checksums. These regions contain all of the information that directly pertains to the player and their Pokémon. Additional information pertinent to the save file is also present in the other regions of the data.

The 32KiB save data, equal to

Contents

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

Data types

Unless otherwise noted, integer values occupy the specified number of bytes, and are big-endian and either unsigned or two's complement.

Text data is stored in a proprietary encoding.

Item lists

Item lists in the save data follow a specific format.

Lists have entries of 2 bytes each as well as a capacity. The total size of the list data in bytes, is $capacity \times 2 + 2$.

For example, the player's pocket inventory can hold 20 item entries, so the size of the list is $20 \times 2 + 2 = 42$ bytes.

Offset	Size	Contents
`0x00`	1	Count
`0x01`	2 × Capacity	Item entries
...	1	Terminator

Count

The number of item entries actually being represented in the list. Note that count and capacity are separate.

Entries

The exact data for each item entry in the list.

Terminator

The byte following the last item entry, according to *Count*, must always be a terminator, which is byte value `0xFF`.

This spare byte is present at the end of the list data to handle the list being filled to capacity.

Entry format

The *Entry* record has the following format:

Offset	Size	Contents
`0x00`	1	Count
`0x01`	1	Index

Count

The amount of that particular item. This value must be between 1 and 99 inclusive.

Index

The item's index.

Pokémon lists

Lists of Pokémon in the save data follow a particular format.

Lists have entries of varying sizes, and a capacity. The total size of the list data in bytes is $(capacity \times (size + 23) + 2)$ in a save file from an English game, and $(capacity \times (size + 13) + 2)$ in a save file from a Japanese game.

For example, the player's Pokémon team contains 6 entries and each entry is 48 bytes in size, so the size of that list is $6 \times (48 + 23) + 2 = 428$ bytes in an English save file, and $6 \times (48 + 13) + 2 = 368$ bytes in a Japanese one.

Offset	Size	Contents
`0x0000`	1	Count
`0x0001`	Capacity + 1	Species
...	Capacity × Size	Pokémon
...	Capacity × 11	OT Names
...	Capacity × 11	Names

Count

The number of Pokémon entries actually being represented in the list. Note that the count and the count and the capacity are not the same thing—a Pokémon list may have a capacity of, say, 30 Pokémon but only use a few of those slots. The count tells how many are actually used.

Species

A list of species indexes, one for each Pokémon in the list. This is used by the team menu as well as the PC management interface.

The byte following the last species entry, according to *Count*, must always be a terminator, which is the byte `0xFF`. This means the species section is one byte longer than just the index numbers would make it.

If an entry in this field is set to `0xFD`, then the corresponding Pokémon is in an Egg.

Pokémon

The exact data for each Pokémon entry in the list. For the format, please refer to: Pokémon data structure (Generation II)

- For party Pokémon, the entry size is the full 48 bytes as documented in that article.
- For PC Pokémon, only the first 32 bytes are used, meaning everything after *Level* is not included. Instead, those values are regenerated upon withdrawing a Pokémon from the PC. This is the basis of the Box trick.

OT names

Text strings representing the names of the original Trainers for each Pokémon entry. Each name can contain up to 10 characters in a save file from an English game, and up to 5 characters in a save file from Japanese one. Following the 10 or 5 bytes, there is always one `0xFF` byte, for a total of 11 or 6 bytes per name.

Names

Text strings representing the names for each Pokémon entry. Each name can contain up to 10 characters in a save file from an English game, and up to 5 characters in a save file from Japanese one. Following the 10 or 5 bytes, there is always one `0xFF` byte, for a total of 11 or 6 bytes per name.

A name is considered a "nickname" if it does not perfectly match the default name for a Pokémon. The default name follows these rules:

- The first however many characters must match the species name exactly. This is typically all-uppercase.
- The remainder of the string must be all terminator characters, aka `0x50`.

Therefore, if a Pokémon with a 9- or 10-letter species name, such as Charmander, is given a nickname that matches the species name, the nickname will not be retained should that Pokémon evolve.

File structure

Known data within the save file can be found at the following offsets within the data, such that offset 0 is the first byte of an emulator ".sav" file.

Although all data appears twice in the save file, only the primary copy is documented below. For more information, see the Checksums section.

English and Japanese save files are quite a bit different, mostly due to Japanese Generation II games having 9 boxes in the PC of 30 Pokémon each, while the English games have 14 boxes of 20 each. The save format also differs between game versions, as Crystal has more features than Gold and Silver.

English Offset		Size	Japanese Offset		Size	Contents
GS	C		GS	C		
0x2000	0x2000	8	0x2000	0x2000	8	Options
0x2008	0x2008	3	0x2008	0x2008	3	Player Trainer ID
0x200B	0x200B	11	0x200B	0x200B	6	Player name
0x2016	0x2016	11	0x2011	0x2011	6	Unused (Player's mom name)
0x2021	0x2021	11	0x2017	0x2017	5	Rival name
0x202C	0x202C	11	0x201B	0x201B	6	Unused (Red's name)
0x2030	0x2030	5	0x2021	0x2021	6	Unused (Green's name)
0x2042	0x2042	1	0x2029	0x2029	1	Daylight savings
0x2053	0x2053	4	0x2034	0x2035	4	Time played
0x206B	0x206A	1	0x204C	0x204C	1	Player palette
0x23E2	0x23E3	2	0x23C3	0x23C5	2	Game Coins
0x23DB	0x23DC	3	0x23BC	0x23BE	3	Money
0x23E4	0x23E5	1	0x23C5	0x23C7	1	Johto Badges
0x23E5	0x23E6	1	0x23C6	0x23C8	1	Kanto Badges
0x23E6	0x23E7	57	0x23C7	0x23C9	57	TM pocket
0x241F	0x2420	42	0x2400	0x2402	42	Item pocket item list
0x2449	0x244A	27	0x242A	0x242C	27	Key item pocket item list
0x2464	0x2465	26	0x2445	0x2447	26	Ball pocket item list
0x247E	0x247F	102	0x245F	0x2461	102	PC item list
0x2724	0x2700	3	0x2705	0x26E2	3	Current PC Box number
0x2727	0x2703	127	0x2708	0x26E5	81	PC Box names
0x2851		2				Lucky Number Show weekly number
	0x2780	28		0x2735	28	Saved map header
0x2868	0x2843	4	0x281C	0x27F8	4	Player location
0x286C	0x2847	30	0x2820	0x27FC	30	Saved map tiles
0x288A	0x2865	428	0x283E	0x281A	368	Party Pokémon list
0x2A4C	0x2A27	32	0x29CE	0x29AA	32	Pokédex owned
0x2A6C	0x2A47	32	0x29EE	0x29CA	32	Pokédex seen
0x2AA9		11				Pokémon Day Care Pokémon #1 Name
0x2ABF		32				Pokémon Day Care Pokémon #1 Data
0x2AC8		2				Pokémon Day Care step count

0x2AE0		2				Pokémon Day Care incr/decr per step
0x2B01		2				Pokémon Day Care step count
0x2AE2		11				Pokémon Day Care Pokémon #2 Name
0x2AF8		32				Pokémon Day Care Pokémon #2 Data
0x2D6C	0x2D10	1102	0x2D10	0x2D10	1352	Current Box Pokémon list
N/A	0x3E3D	1	*N/A*	0x8000	1	Player gender
0x4000	0x4000	1102	0x4000	0x4000	1352	PC Box 1 Pokémon list
0x4450	0x4450	1102	0x454A	0x454A	1352	PC Box 2 Pokémon list
0x48A0	0x48A0	1102	0x4A94	0x4A94	1352	PC Box 3 Pokémon list
0x4CF0	0x4CF0	1102	0x4FDE	0x4FDE	1352	PC Box 4 Pokémon list
0x5140	0x5140	1102	0x5528	0x5528	1352	PC Box 5 Pokémon list
0x5590	0x5590	1102	0x5A72	0x5A72	1352	PC Box 6 Pokémon list
0x59E0	0x59E0	1102	0x6000	0x6000	1352	PC Box 7 Pokémon list
0x6000	0x6000	1102	0x654A	0x654A	1352	PC Box 8 Pokémon list
0x6450	0x6450	1102	0x6A94	0x6A94	1352	PC Box 9 Pokémon list
0x68A0	0x68A0	1102		*N/A*		PC Box 10 Pokémon list
0x6CF0	0x6CF0	1102		*N/A*		PC Box 11 Pokémon list
0x7140	0x7140	1102		*N/A*		PC Box 12 Pokémon list
0x7590	0x7590	1102		*N/A*		PC Box 13 Pokémon list
0x79E0	0x79E0	1102		*N/A*		PC Box 14 Pokémon list
0x2D69	0x2D0D	2		0x2D0D	2	Checksum 1
0x7E6D	0x1F0D	2		0x7F0D	2	Checksum 2

Player Trainer ID

Represents the Trainer ID number in two bytes, which can range from 0 to 65535. In Crystal, the boy player character will always have an even number ID, while the girl's ID will always be odd.

Player name

Represents text strings that can be from 1 to 7 characters in length, although the save structure allocates 11 bytes for the name. In the Japanese version, the amount of characters is reduced from 7 to 5, and the save structure allocates 6 bytes instead of 11.

The first 8 bytes contain the name with any leftover equal to `0x50`.

Since the name can be 7 bytes at most, the eighth byte, or sixth byte in the Japanese version, will always be `0x50`.

The remaining 3 bytes are all `0x00`.

Rival name

Represents text strings that can be from 1 to 7 characters in length, although the save structure allocates 11 bytes for the name. In the Japanese version, the amount of characters is reduced from 7 to 5, and the save structure allocates 6 bytes instead of 11.

The first 8 bytes contain the name with any leftover equal to `0x50`.

Since the name can be 7 bytes at most, the eighth byte, or sixth byte in the Japanese version, will always be `0x50`.

The remaining 9th, 10th, and 11th bytes are equal to `0x86`, `0x91`, `0x84` respectively.

Daylight savings

Specifies whether daylight savings time (DST) is in effect.

The highest bit of this field is set to indicate DST is in effect.

The lower 7 bits of this field have unknown significance.

Time played

Specifies how much time has elapsed during gameplay.

This value is actually 4 1-byte values representing, in this order: the hours, minutes, seconds and "frames" that have elapsed. A frame is 1/60th of a second.

This timer is not halted when the game is paused, and also counts up on the main menu before selecting to continue a saved game.

Player palette

Specifies the colors of the player character.

From a technical standpoint, the lowest 3 bits of this field are transferred to OAM to select the colors when drawing the player character. This means that there are a total of 8 possible palettes:

- Red, `0x00`
- Blue, `0x01`
- Green, `0x02`
- Brown, `0x03`
- Orange, `0x04`
- Gray, `0x05`
- Dark Green, `0x06`
- Dark Red, `0x07`

From a practical standpoint, this value is set by the game depending on whether the player character is a boy or a girl:

- For boy characters, this is set to `0x00` (red)
- For girl characters, this is set to `0x01` (blue)

Despite only being able to make boy characters in Gold and Silver, this field is still present and functional.

Johto Badges

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

From MSB to LSB, badges are in this order: Zephyr, Insect, Plain, Fog, Storm, Mineral, Glacier, Rising.

TM pocket

The items that the player has in their TM Pocket inventory.

Offset	Size	Contents
`0x00`	50	TMs list
`0x32`	7	HMs list

TMs list

Each byte specifies the quantity of the corresponding TM that the player is holding. Should be 0 to 99.

Indexes match item numbers, meaning 0 corresponds with TM01 and 49 corresponds with TM50.

HMs list

Each byte specifies the quantity of the corresponding HM that the player is holding. Should be 0 to 1.

Indexes match item numbers, meaning 0 corresponds with HM01 and 6 corresponds with HM07.

Item pocket item list

The items that the player has in their item pocket inventory.

Stored as an Item list with a capacity of 20.

Key item pocket item list

The items that the player has in their Key Item Pocket inventory.

Stored as an Item list with a capacity of 26.

Ball pocket item list

The items that the player has in their Ball Pocket inventory.

Stored as an Item list with a capacity of 12.

PC item list

The items that the player has stored in the PC.

Stored as an Item list with 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 14 is represented as 13.

The lowest 4 bits of this value are the box index.

PC Box names

The 9 (Japanese game) or 14 (English game) box names. Each name is a string between 1 and 8 characters plus a terminator byte, for a total of 9 bytes each.

Lucky Number Show weekly number

This week's number for the Lucky Number Show. This field is populated the first time each week when the player listens to the Lucky Number Show or visits the Radio Tower to redeem their prize. If that has not yet occurred, this value will contain a previous week's lucky number.

Player location

The map bank and map number for the map the player is currently in, followed by the player's X/Y position. Defaults to `0x18` `0x07` `0x03` `0x03` upon starting a new game.

Party Pokémon list

The Pokémon that the player has in their party.

Stored as a Pokémon list with a capacity of 6 and an entry size of 48 bytes.

Pokédex owned, Pokédex seen

The Pokémon registered in the Pokédex as seen or owned are represented by a pair of 32-byte little-endian bit arrays (for a total of 256 bits). Each bit represents whether a particular Pokémon has been seen/owned or not. The first array represents whether the Pokémon has been owned or not, while the second represents whether the Pokémon has been seen or not.

Pokémon are listed in National Pokédex order, with bit 0 corresponding to #001 Bulbasaur, bit 1 corresponding to #002 Ivysaur, and so on, up to bit 250 corresponding to

#251 Celebi. Bits 251-255 are unused.

PC Box Pokémon lists

The Pokémon that the player has stored in PC boxes.

Stored as Pokémon lists with a capacity of 20 Pokémon in English save files and 30 in Japanese ones, and an entry size of 32 bytes. After every list is the two bytes `FF00`.

Normally, Pokémon are deposited and withdrawn from the Current Box list, 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.

Player gender

Specifies the gender of the player character:

- For boy characters, this is set to `0x00`
- For girl characters, this is set to `0x01`

This field is not within the checksum-validated region.

Checksums

Used to validate the integrity of saved data.

Player data in Generation II is stored in the save file twice. The primary copy is located at `0x2009`, and a secondary copy is stored elsewhere in the file. Checksums are performed on both copies and stored in the data.

- If only one checksum is correct, then the information from that copy of the data will be used.
- If both values are incorrect, the player will be forced to start a new game.

The checksums are simply the 16-bit sum of all byte values of the corresponding byte regions. Checksums are stored as little-endian.

Gold and Silver

The secondary data copy in Gold and Silver is not contiguous like the primary copy is. Instead, it is split across 5 sections and stored at different locations in the save file. The following table shows which sections of the primary copy are relocated for the secondary copy:

Primary		Secondary	
From	To	From	To
0x2009	0x222E	0x15C7	0x17EC
0x222F	0x23D8	0x3D96	0x3F3F
0x23D9	0x2855	0x0C6B	0x10E7

0x2856	0x2889	0x7E39	0x7E6C
0x288A	0x2D68	0x10E8	0x15C6

Calculating the checksums, therefore, can be done as follows:

- Sum the bytes from `0x2009` to `0x2D68` and store the result at `0x2D69`
- Sum the bytes from `0x0C6B` to `0x17EC`, `0x3D96` to `0x3F3F` and `0x7E39` to `0x7E6C`, and store the result at `0x7E6D`

For the Japanese games, the both checksum ranges are contiguous. The primary range goes from `0x2009` to `0x2C8B` and is stored at `0x2D0D`. The secondary range goes from `0x7209` to `0x7E8B` and is stored at `0x7F0D`.

Crystal

The secondary data copy in Crystal is a byte-for-byte match of the primary copy. The following table shows which regions of the save file are occupied by each copy:

Primary		Secondary	
From	To	From	To
0x2009	0x2B82	0x1209	0x1D82

Calculating the checksums, therefore, can be done as follows:

- Sum the bytes from `0x2009` to `0x2B82` and store the result at `0x2D0D`
- Sum the bytes from `0x1209` to `0x1D82` and store the result at `0x1F0D`

For Japanese games, the secondary partition is from `0x7209` to `0x7D3A`. However both checksums are calculated using `0xADA`(2778) bytes from the initial location as opposed to the 2937 bytes in the English version, so the Japanese checksums can be calculated as follows:

- Sum the bytes from `0x2009` to `0x2AE2` and store the result at `0x2D0D`
- Sum the bytes from `0x7209` to `0x7CE2` and store the result at `0x7F0D`

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

Retrieved from "https://bulbapedia.bulbagarden.net/w/index.php?title=Save_data_structure_(Generation_II)&oldid=4433435"