Add a new Pokémon

Jump to bottom

Rangi edited this page 8 days ago · 18 revisions

This tutorial is for how to add a new species of Pokémon, allowing up to 253 species. As

1. Define a species constant

Edit constants/pokemon_constants.asm:

```
; pokemon ids
; indexes for:
; - PokemonNames (see data/pokemon/names.asm)
; - BaseData (see data/pokemon/base_stats.asm)
; - EvosAttacksPointers (see data/pokemon/evos_attacks_pointers.asm)
; - EggMovePointers (see data/pokemon/egg_move_pointers.asm)
; - PokemonCries (see data/pokemon/cries.asm)
; - MonMenuIcons (see data/pokemon/menu_icons.asm)
; - PokemonPicPointers (see data/pokemon/pic_pointers.asm)
; - PokemonPalettes (see data/pokemon/palettes.asm)
; - PokedexDataPointerTable (see data/pokemon/dex_entry_pointers.asm)
; - AlphabeticalPokedexOrder (see data/pokemon/dex_order_alpha.asm)
; - EZChat_SortedPokemon (see data/pokemon/ezchat_order.asm)
; - NewPokedexOrder (see data/pokemon/dex_order_new.asm)
; - Pokered_MonIndices (see dat

                                                          sm)
; - BitmasksPointers (see gfx/pokemon/bitmask_pointers.asm)
; - FramesPointers (see gfx/pokemon/frame_pointers.asm)
; - Footprints (see gfx/footprints.asm)
	const_def 1
	const BULBASAUR  ; 01
	...
	const MEW        ; 97
JOHTO_POKEMON EQU const_value
	const CHIKORITA  ; 98
	...
	const CELEBI     ; fb
+	const MUNCHLAX   ; fc
NUM_POKEMON EQU const_value + -1
-	const MON_FC     ; fc
	const EGG        ; fd
	const MON_FE     ; fe
```

Some things to notice here:

- Species constants are in national dex order; in fact, they're used as Pokédex IDs, so Munchlax is going to appear as #252.
- `JOHTO_POKEMON` is defined right before the first Johto Pokémon, and `NUM_POKEMON` is defined right after the last non-Egg Pokémon.
- There are a lot of data tables associated with Pokémon species! We'll go over them one by one.

2. Give it a name

Edit data/pokemon/names.asm:

```
PokemonNames::
	db "BULBASAUR@"
	...
	db "CELEBI@@@@"
-	db "?????@@@@@"
+	db "MUNCHLAX@@"
	db "EGG@@@@@@@"
	db "?????@@@@@"
	db "?????@@@@@"
	db "?????@@@@@"
```

All the names are exactly 10 characters long, with "@" as padding. Names that are 10 characters long anyway, like "CHARMELEON", don't have any padding.

(Aside: if you make names lowercase, be careful—going from "FARFETCH'D" to "Farfetch'd" will lower the character count by 1 since `'d` is a single character, so that would need to become `"Farfetch'd@"` .)

3. Define its b

[redacted]

means:

r declaring which ROM bank the Pokédex entry is stored it. It's
ce the exact species ID is insignificant, but it's not a big deal;
here and move on.

ense, Speed, Special Attack, and Special Defense, each from 1

ndary types. Single-type Pokémon just repeat their type

formula for capturing wild Pokémon. Lower rates are harder to
, 255, is used for common Pokémon like Hoothoot and Rattata;
nd Ho-Oh get as low as 3.

- **base experience yield:** A factor in the for values give more experience. As usual, since this is a single byte (`db`), its maximum is 255. (Chansey and Blissey both have 255 base exp.)
- **items:** A common item (23% chance) and a rare item (2% chance) that might be held by this Pokémon in the wild.
- **gender ratio:** The likelihood that an Egg of this Pokémon will be female. Lower values are more likely to be male, with 0 being 100% male, 254 being 100% female, and 255 being genderless (like Magnemite or Staryu). There are named constants for the few values that get officially used:
  - `GENDER_F0` : 100% male
  - `GENDER_F12_5` : 7/8 male, 1/8 female
  - `GENDER_F25` : 3/4 male, 1/4 female
  - `GENDER_F50` : 1/2 male, 1/2 female
  - `GENDER_F75` : 1/4 male, 3/4 female
  - `GENDER_F100` : 100% female
  - `GENDER_UNKNOWN` : genderless
- **unknown 1:** An unused value that's always 100.
- **step cycles t**

- **front.dimensions:** The size of the Pokémon's front sprite. The front.dimensions file contains a single byte that's one of three valid values: $55 for a 40x40-pixel (5x5-tile) sprite, $66 for 48x48 pixels (6x6 tiles), or $77 for 56x56 pixels (7x7 tiles). It's automatically generated from the front.png sprite image (we'll get to this later).
- **padding:** Four unused bytes that are always 0s. (In the G/S prototype, these values were pointers to the front and back sprites.)
- **growth rate:** The formula that's used to determine experience point thresholds for reaching each level. (The formula coefficients are defined in data/growth_rates.asm.) Valid values:
  - `GROWTH_MEDIUM_FAST` : $exp = L^3$ (1,000,000 exp = level 100)
  - `GROWTH_SLIGHTLY_FAST` : $exp = (3/4)L^3 + 10L^2 - 30$ (849,970 exp = level 100); unused
  - `GROWTH_SLIGHTLY_SLOW` : $exp = (3/4)L^3 + 20L^2 - 70$ (949,930 exp = level 100); unused
  - `GROWTH_MEDIUM_SLOW` : $exp = (6/5)L^3 - 15L^2 + 100L - 140$ (1,059,[redacted]
    00,000 exp = level 100)
  - `GROWTH_SLOW` : $exp = (5/4)L^3$ (1,250,000 exp = level 100)
- **egg groups:** The two egg groups this Pokémon is in. Two Pokémon have to share an Egg group to breed. The 15 valid Egg groups (including `EGG_NONE` for sterile Pokémon like babies and legendaries) are defined in constants/pokemon_data_constants.asm.
- **TM/HM learnset:** A list of which TMs, HMs, and tutor moves this Pokémon can learn, passed to the `tmhm` macro. Valid values are defined in constants/item_constants.asm with the `add_tm` , `add_hm` , and `add_mt` macros.

Then edit data/pokemon/base_stats.asm:

```
 BaseData::
 INCLUDE "data/pokemon/base_stats/bulbasaur.asm"
 ...
 INCLUDE "data/pokemon/base_stats/celebi.asm"
+INCLUDE "data/pokemon/base_stats/munchlax.asm"
```

4. Define its evolutions and level-up learnset

Edit data/pokemon/evos_attacks_pointers.asm:

```
 [redacted]
                 asaurEvosAttacks
        ...
        dw CelebiEvosAttacks
+       dw MunchlaxEvosAttacks
```

Then edit data/pokemon/evos_attacks.asm:

[redacted]

```
             29, HEADBUTT
+       db 36, SCREECH
+       db 36, REST
+       db 43, BODY_SLAM
+       db 50, ROLLOUT
+       db 57, HYPER_BEAM
+       db 0 ; no more level-up moves
```

The comment at the top of evos_attacks.asm explains the data structure:

```
EvosAttacks::
; Evos+attacks data structure:
; - Evolution methods:
;    * db EVOLVE_LEVEL, level, species
;    * db EVOLVE_ITEM, used item, species
;    * db EVOLVE_TRADE, held item (or -1 for none), species
;    * db EVOLVE_HAPPINESS, TR_* constant (ANYTIME, MORNDAY, NITE), species
;    * db EVOLVE_STAT, level, ATK_*_DEF constant (LT, GT, EQ), species
; - db 0 ; no more evolutions
; - Learnset (in increasing level order):
;    * db level, move
; - db 0 ; no more level-u
```

```diff
 EggMovePointers::
        dw BulbasaurEggMoves
        ...
        dw AerodactylEggMoves
-       dw SnorlaxEggMoves
+       dw NoEggMoves
        d


          w MunchlaxEggMoves
```

7. Define its icon

Edit data/pokemon/menu_icons.asm:

```
 MonMenuIcons:
 	db ICON_BULBASAUR    ; BULBASAUR
 	...
 	db ICON_HUMANSHAPE  ; CELEBI
+	db ICON_SNORLAX     ; MUNCHLAX
```

Valid icons are in constants/icon_constants.asm. They're used in the party menu and Day-Care.

8. Define its Pokédex entry

Create **data/pokemon/dex_entries/munchlax.asm**:

```
+	db "BIG EATER@" ; species name
+	dw 200, 2315 ; height, weight
+
+	db   "In its desperation"
+	next "to gulp down food,"
+	next "it forgets about"
+
+	page "the food it has"
+	next "hidden under its"
+	next "fur.@"
```

- The species name can be up to 10 characters (technically 11 will still fit in the Pokédex window). Be sure to end it with a "@".
- The height and weight values are in imperial units. 200 = 2 feet 0 inches; 2315 = 231.5 pounds.
- The entry text is in two pages; the first starts with `db`, the second with `page`. Each page can fit three lines with 18 characters each. Here it's visual size that matters— `"#MON"` counts as 7 characters because it prints as "POKéMON". Again, be sure to end it with a "@".

Edit data/pokemon/dex_entry_pointers.asm:

```
 PokedexDataPointerTable:
 ; entries correspond to constants/pokemon_constants.asm
 	dw BulbasaurPokedexEntry
 	...
 	dw CelebiPokedexEntry
+	dw MunchlaxPokedexEntry
```

Then edit data/pokemon/dex_entries.asm:

```
 SECTION "Pokedex Entries 193-251", ROMX

 PokedexEntries4::
 YanmaPokedexEntry::      INCLUDE "data/pokemon/dex_entries/yanma.asm"
 ...
 CelebiPokedexEntry::     INCLUDE "data/pokemon/dex_entries/celebi.asm"
+MunchlaxPokedexEntry::   INCLUDE "data/pokemon/dex_entries/munchlax.asm"
```

Each of the four sections in dex_entries.asm holds 64 Pokédex entries. Each section is in a different ROM bank, decided by which range the Pokémon's species ID is in: 1–64, 65–128, 129–192, or 193–256. The order of entries within each section doesn't actually matter, since they're accessed directly via `PokedexDataPointerTable`, but avoid confusion and just keep them in numerical order.

Now the Pokédex entry exists, and will be listed correctly in "Old Pokédex Mode" (the Gen 2 equivalent of national order). But we need to define the new Pokémon's place in "New Pokédex Mode (regional order) and "A to Z Mode" (alphabetical order).

Edit data/pokemon/dex_order_new.asm:

```
 NewPokedexOrder:
 	db CHIKORITA
 	...
 	db AERODACTYL
+	db MUNCHLAX
 	db SNORLAX
 	db BULBASAUR
 	...
 	db CELEBI
```

Then edit data/pokemon/dex_order_alpha.asm:

```
 AlphabeticalPokedexOrder:
 	db ABRA
 	...
 	db MUK
+	db MUNCHLAX
 	db MURKROW
```

```
    ...
    db ZUBAT
```

That's all for the Pokédex.

9. Design its footprint

Create **gfx/footprints/munchlax.png**:

Then edit gfx/footprints.asm:

```
 ; Footprints are 2x2 tiles each, but are stored as a 16x64-tile image
 ; (32 rows of 8 footprints per row).
 ; That means there's a row of the top two tiles for eight footprints,
 ; then a row of the bottom two tiles for those eight footprints.

 ; These macros help extract the first and the last two tiles, respectively.
 fo



                      to Pokémon species, two apiece, 8 tops then 8 bottoms

 ...
 ; 249-256 top halves
 INCBIN "gfx/footprints/lugia.1bpp",      footprint_top
 INCBIN "gfx/footprints/ho_oh.1bpp",      footprint_top
 INCBIN "gfx/footprints/celebi.1bpp",     footprint_top
-INCBIN "gfx/footprints/252.1bpp",        footprint_top
+INCBIN "gfx/footprints/munchlax.1bpp",   footprint_top
 INCBIN "gfx/footprints/253.1bpp",        footprint_top
 INCBIN "gfx/footprints/254.1bpp",        footprint_top
 INCBIN "gfx/footprints/255.1bpp",        footprint_top
 INCBIN "gfx/footprints/256.1bpp",        footprint_top
 ; 249-256 bottom halves
 INCBIN "gfx/footprints/lugia.1bpp",      footprint_bottom
 INCBIN "gfx/footprints/ho_oh.1bpp",      footprint_bottom
 INCBIN "gfx/footprints/celebi.1bpp",     footprint_bottom
-INCBIN "gfx/footprints/252.1bpp",        footprint_bottom
+INCBIN "gfx/footprints/munchlax.1bpp",   footprint_bottom
 INCBIN "gfx/footprints/253.1bpp",        footprint_bottom
 INCBIN "gfx/footprints/254.1bpp",        footprint_bottom
 INCBIN "gfx/footprints/255.1bpp",        footprint_bottom
 INCBIN "gfx/footprints/256.1bpp",        footprint_bottom
```

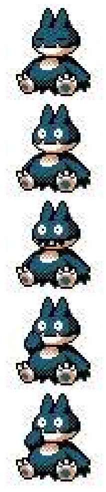
Notice how the footprints are broken into top and bottom halves; you may want to correct this design flaw. (It won't have a visible consequence on the game, but it makes for cleaner code and data.)

(Running `make` later will automatically create munchlax.1bpp from munchlax.png. Since it's a 1bpp file (**one b**it **p**er **p**ixel) you can only use black and white in the image.)

10. Design its sprites and animation

This is a bit complicated. We need a front sprite with an animation sequence; a back sprite; and normal and shiny color palettes for both.

Create **gfx/pokemon/munchlax/front.png**:



An

Now create **gfx/pokemon/munchlax/anim.asm**:

```
+       frame 0, [illegible]
+       frame 1, [illegible]
+       frame 0, [illegible]
+       frame 2, [illegible]
+       frame 3, [illegible]
+       frame 4, [illegible]
+       frame 3, [illegible]
+       frame 3,[illegible]
+       fr
```

- **frames.asm:** A sequence of lists that declare the tile IDs with which to fill in the changed spots in each frame.

There are two main reasons to pay attention to these auto-generated files. One, you want to make sure that front.gbcpal got its colors in the right order, and have shiny.pal match it. Two, if your animation appears corrupt, you want to make sure that frames.asm isn't using too high tile IDs. A 40x40 sprite can use IDs $00 to $31; a 48x48 sprite can use up to $47; a 56x56 sprite can use up to $61. If you see IDs above those limits, edit your front.png frames so they use fewer unique tiles, or follow this tutorial to allow any animation to use 255 tiles ($00 to $FF but skipping $7F).

(Older versions of pokecrystal generated normal.pal and normal.gbcpal files instead of front.gbcpal, but this was redundant work and could mislead users into editing normal.pal directly, so as of September 2018 they were removed.)

Anyway, all that's left is to `INCLUDE` and point to the new data!

11. Include and point to the sprite and animation data

Edit data/pokemon/pic_pointers.asm:

```
PokemonPicPointers::
; entries correspond to Pokémon species, two apiece
	dba_pic BulbasaurFrontpic
	dba_pic BulbasaurBackpic
	...
	dba_pic Cel
 SECTION "Pics 19", ROMX

-; Seems to be an accidental copy of the previous bank
-
-INCBIN "gfx/pokemon/spinarak/back.2bpp.lz"
-...
-INCBIN "gfx/pokemon/unown_r/back.2bpp.lz"
+MunchlaxFrontpic: INCBIN "gfx/pokemon/munchlax/front.animated.2bpp.lz"
+MunchlaxBackpic:  INCBIN "gfx/pokemon/munchlax/back.2bpp.lz"
```

(If you *don't* fix the `dba_pic` design flaw, you'll have to put your sprites in the "Pics *N*" sections, which are compatible with `dba_pic`. "Pics 19" isn't used for

```
-       RGB 30, 26, 11
-       RGB 23, 16, 00
+INCBIN "gfx/pokemon/munchlax/front.gbcpal", middle_colors
+INCLUDE "gfx/pokemon/munchlax/shiny.pal"

 INCBIN "gfx/pokemon/egg/front.gbcpal", middle_colors
 INCLUDE "gfx/pokemon/egg/shiny.pal"

 ; 254
        RGB 30, 26, 11
        RGB 23, 16, 00
 ; 254 shiny
        RGB 30, 26, 11
        RGB 23, 16, 00

 ; 255
        RGB 23, 23, 23
        RGB 17, 17, 17
 ; 255 shi
```

Edit gfx/pokemon/anims.asm:

```
 PicAnimations:
 BulbasaurAnimation:  INCLUDE "gfx/pokemon/bulbas ur/anim


                                         on/egg/anim.asm"
```

Edit gfx/pokemon/idle_pointers.asm:

```
 AnimationIdlePointers:
        dw BulbasaurAni
        ...
        dw CelebiAnimationIdle
+       dw MunchlaxAnimationIdle
```

Edit gfx/pokemon/idles.asm:

```
 BulbasaurAnimationIdle:  INCLUDE "gfx/pokemon/bulbasaur/anim_idle.asm"
 ...
 CelebiAnimationIdle:     INCLUDE "gfx/pokemon/celebi/anim_idle.asm"
+MunchlaxAnimationIdle:   INCLUDE "gfx/pokemon/munchlax/anim_idle.asm"
 EggAnimationIdle:        INCLUDE "gfx/pokemon/egg/anim_idle.asm"
```

Edit gfx/pokemon/bitmask_pointers.asm:

```
 BitmasksPointers:
        dw BulbasaurBitmasks
        ...
        dw CelebiBitmasks
+       dw MunchlaxBitmasks
```

Edit gfx/pokemon/bitmasks.asm:

```
 BulbasaurBitmasks:  INCLUDE "gfx/pokemon/bulbasaur/bitmask.asm"
 ...
 CelebiBitmasks:     INCLUDE "gfx/pokemon/celebi/bitmask.asm"
+MunchlaxBitmasks:   INCLUDE "gfx/pokemon/munchlax/bitmask.asm"
 EggBitmasks:        INCLUDE "gfx/pokemon/egg/bitmask.asm"
```

Edit gfx/pokemon/frame_pointers.asm:

```
 FramesPointers:
        dw BulbasaurFrames
        ...
        dw CelebiFrames
+       dw MunchlaxFrames
```

Finally, edit gfx/pokemon/johto_frames.asm:

```
 JohtoFrames:
 ChikoritaFrames:  INCLUDE "gfx/pokemon/chikorita/frames.asm"
 ...
 CelebiFrames:     INCLUDE "gfx/pokemon/celebi/frames.asm"
```

+Mur

Egg

te t

ro

Unfortunately, you can't have more than 253 Pokémon. IDs are one byte each, so they have 256 possible values. Of those values, $00 indicates a lack of a Pokémon; $FF (−1) is an end-of-list marker; and $FD is `EGG` (though you can change this to $FE, as discussed above). If you value having a 254th Pokémon more than allowing breeding, you could replace `EGG`, but you would also have to carefully remove a lot of code that treats `EGG` specially.