

SRW GC Unit Editor by Dashman – User manual

This program is used to edit most Unit / Weapon / Character data in the game Super Robot Wars GC for the GameCube.

That's actually a lot of things that can be edited and many of them could use some explaining, so you might need this document.

Before using this program

Java

You need Java installed to run this. If you don't have it, go find the latest version of the Java Runtime Environment here:

https://www.java.com/es/download/ie_manual.jsp

Chances are, it won't be enough with that. If the program doesn't run with that, try installing the latest version of the Java Developer Kit as well. Get it from here:

<https://www.oracle.com/java/technologies/javase-jdk16-downloads.html>

That should allow you to run the executable (SRW GC Unit Editor.jar).

For a more comfortable way of using this program, I recommend you use Notepad++ and Excel / LibreOffice (or similar).

Obtaining the data file add02dat.bin

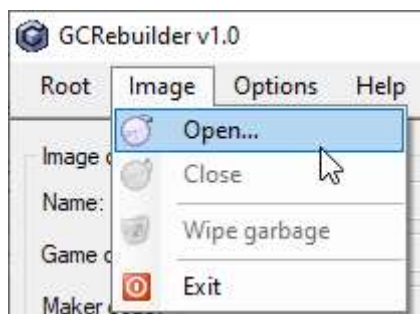
You need to extract the file **add02dat.bin** from the iso of the game in order to make your edits. To extract this file, you'll need to use the program GameCubeRebuilder. Grab it from here:

<https://www.romhacking.net/utilities/619/>

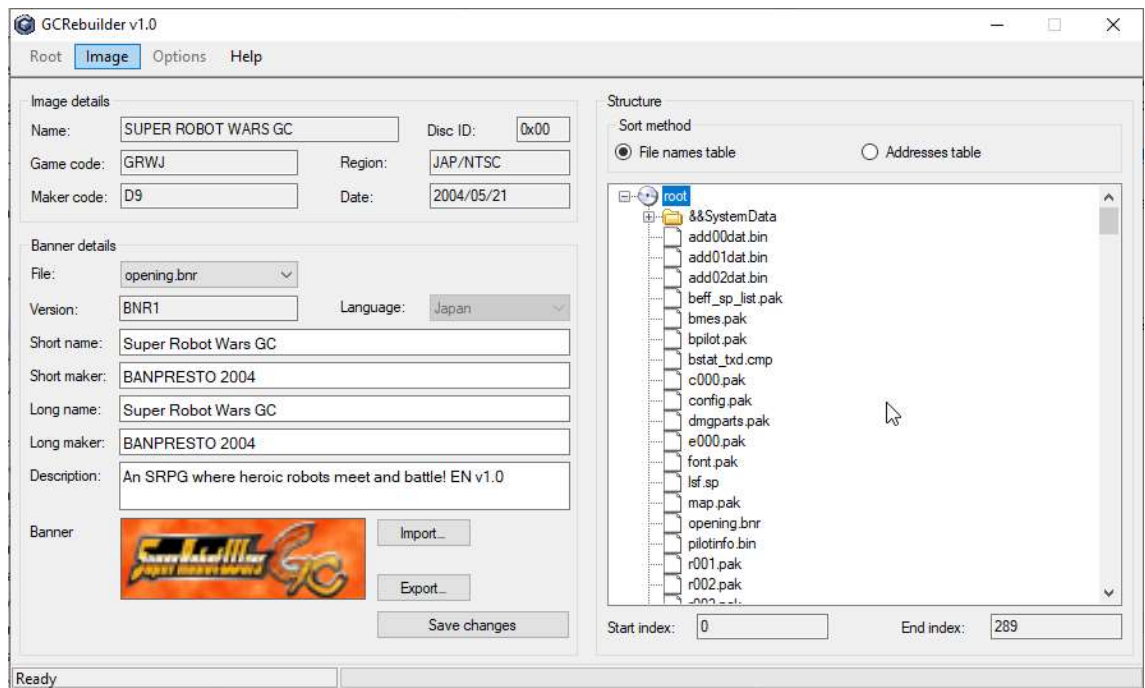
The program allows you to extract files from a GameCube iso file and, after modifying them, rebuild a new iso.

To **extract**, do the following:

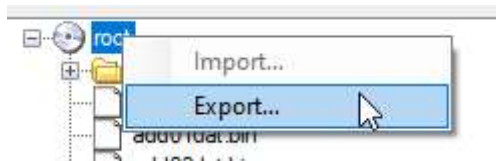
- 1) Use the option *Image* → *Open...*



Choose your iso file. The structure field will show the files inside the disc.



- 2) Right click on the disc in the structure and select "export".



Choose a folder of your liking and wait for the program to finish

- 3) That's it. Find the add02dat.bin file inside the folder you chose.

Later, when you want to **rebuild** your iso, you'll have to do the following:

- 1) From the program without any open ISO, use the option *Root* → *Open...*



Select the "root" folder where you extracted the files. The information in the UI should

look just like when you opened the ISO during the extraction phase.

2) Go to *Root* → *Save*

This won't actually save the iso file. It will just let you select where the iso will be created and the name it should have.

3) Select *Root* → *Rebuild*

This will actually create the iso file.

Only the files that were extracted will be used for rebuilding the iso. That is, if you created a file called "myrandomfile.txt" inside the folder, it won't go into the new iso file. However, if you replace one of the existing files with a modified one the changes will be taken when rebuilding.

4) Once it's finished, the iso will be ready to be run with Dolphin.

Notice that if you open the iso with Dolphin, the iso will be "locked" while Dolphin is running. This means that you can't rebuild the same iso again until you close Dolphin.

Running the program

The JAR file can be run like any executable (select it and press enter, double-click, etc.), but I recommend running it from a command window, as it will register any errors that the program might throw and I haven't been able to control.

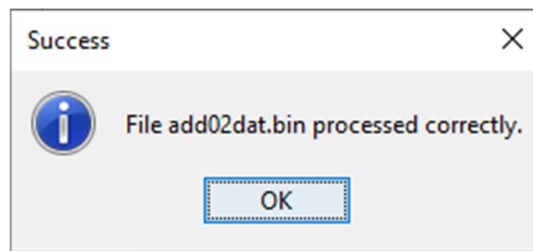
To execute the program via command window, run it like this:

```
java -jar "SRW GC Unit Editor.jar"
```

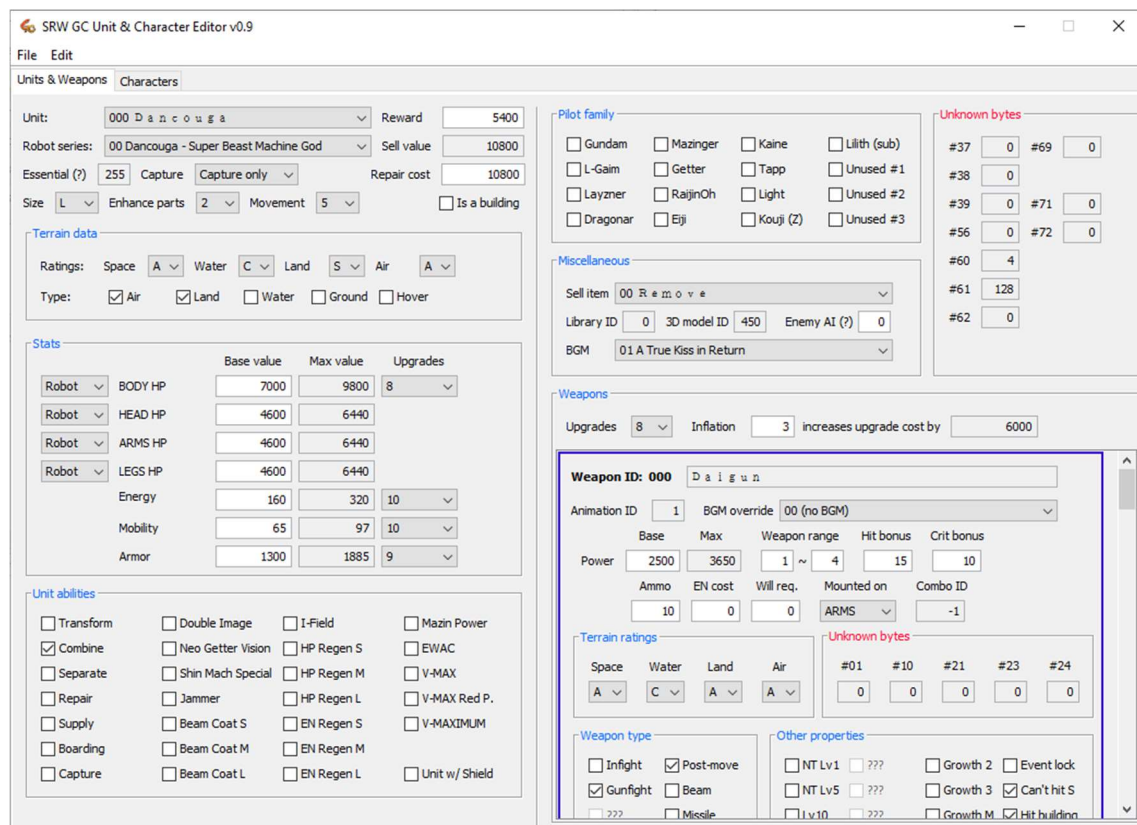
When you finally execute the program, you'll see a window like this one (might change depending on your OS):

There's no information yet loaded into the program. You have to load the add02dat.bin file next:

Browse your file and open it. You'll get a message indicating it was loaded correctly:



And after clicking OK you'll see you have data loaded:

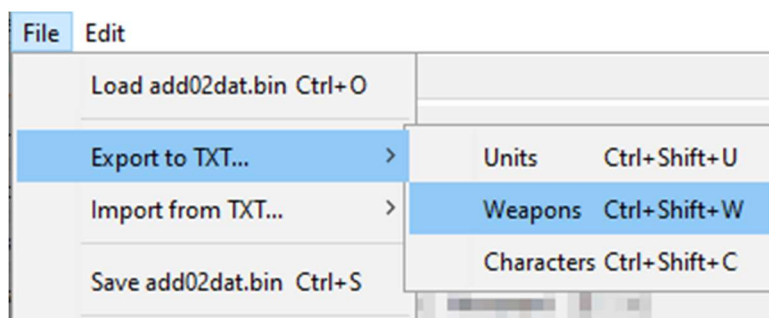


Now you could start editing away, but let me show you how to export and import data out of the program first.

Exporting / Importing data

Once you've loaded the `add02dat.bin` file, you can export its contents to txt files to modify the data more conveniently.

The following options should be unlocked in the menu now:



As you can see, you can export units, weapons and characters independently. Save whatever you want to edit to a txt file of your choice and open it with Notepad++. For example, this is what you would see for a weapons file:

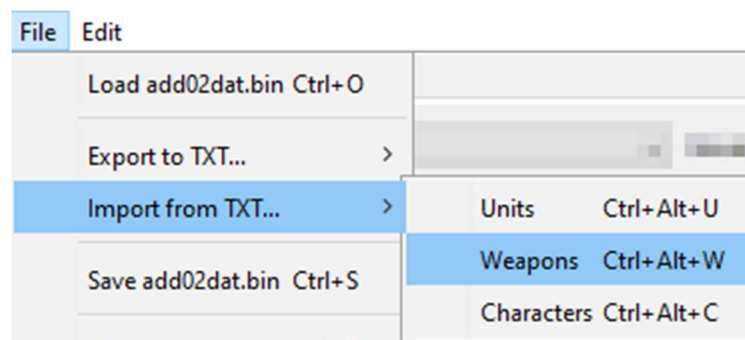
[illegible]

Not very useful like that. Select everything (Ctrl+A) and copy it (Ctrl+C). Open an Excel document and paste everything there (Ctrl+V):

	A	B	C	D	E	F	G	H	I
1	Unit ID	Unit name	Weapon ID	Weapon name	*Animation	BGM override	Base power	Min range	Max range
2	0	D a n c o	0	D a i g u	1	0	2500	1	4
3	0	D a n c o	1	P u n c h	2	0	3000	1	3
4	0	D a n c o	2	D a n c o	3	0	3400	1	1
5	0	D a n c o	3	D a n c o	4	0	4200	2	6
6	0	D a n c o	4	D a n c o	5	0	4500	3	7
7	0	D a n c o	5	D a n c o	6	0	4300	1	8
8	0	D a n c o	6	D a n c o	7	0	4800	1	5
9	1	F i n a l	7	D a i g u	1	0	2700	2	6
10	1	F i n a l	8	P u n c h	2	0	3200	1	5
11	1	F i n a l	9	D a n c o	3	0	3600	1	3

Now you can edit the information in whichever way you like. Add colors, filters, use formulas. Your choice.

When you're done and want to import changes back, simply copy everything from the Excel file into a txt file and use the Import from TXT... options:



This will update the data in the program for whatever you changed.

Some notes on editing data:

- The first columns displaying unit / weapon / character names are just for your convenience. You can change their content without issues.
- It goes without saying, but the program expects to read the same structure of columns in the import file than the one it created in the export file. Don't add or remove columns to the import file, the program will most likely give you an error or do something wrong.
- All editable columns in the txt files contain either integer numbers (no decimals) or the character X (**capital** X). Don't use anything else. Decimals in the file will give you an error when importing, so will characters (a-z, A-Z) and special symbols.
- Only certain columns will work with negative values. Keep reading to find out later or look at the data.

- All fields that are represented by comboboxes in the UI will be saved in the txt file as the selected index of that combobox (first entry is always 0). If you're unsure of what a number means, look at the UI.

Once you've imported the data you'll probably want to save your changes to a new add02dat.bin. You can do that with the option *File → Save add02dat.bin*.

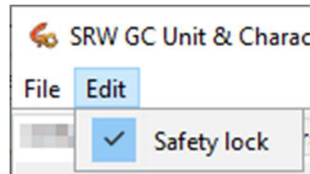
When saving the file, I don't recommend overwriting the original add02dat.bin. Keep an original copy of that file (rename it to add02dat-old.bin or something) and create your new one. Better safe than sorry.

Now with all of that out of the way, let's see all those things in the UI.

Safety

There are several fields in the UI that are initially not editable, either because their effect is not well known or because I've thought changing this value is more likely to break something than any other thing.

If you feel like experimenting, you can enable these fields by removing the safety lock:

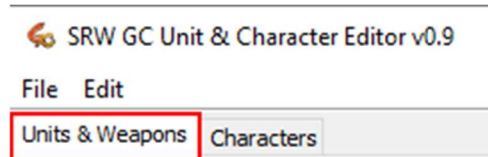


Some fields will remain non-editable though. These fields will contain calculated values that are not actually stored in the file and are just there for your convenience (sell value, max stats, etc).

There's no safety lock in the txt file, obviously, but all fields whose value is locked behind safety has their column name start with an asterisk (*).

Unit & Weapons tab

The user interface has been divided in two major tabs, one for units with their weapons and another one for characters. We'll start by looking at the units:



Many fields in this UI is pretty self-explanatory, so I'll only give details about what may not be quite obvious.

Starting from the top left, we get these fields with general info about the unit (that I had no idea where to group):

Unit:	000 Dancouga	Reward	5400
Robot series:	00 Dancouga - Super Beast Machine God	Sell value	10800
Essential (?)	255	Capture	Capture only
Size	L	Enhance parts	2
		Movement	5
			<input type="checkbox"/> Is a building

The **Unit combobox** (the one with “000 Dancouga” in the picture) is what allows you to jump to a different unit. If you select a different unit from the list, the UI will load the information of that unit. Any changes you’ve performed to that unit will be stored in memory.

The **Robot series** combobox changes the Robot series displayed in the Robot Library entry in game.

The field **Essential (?)** is not clear. This can have a value of either 0 or 255. Units with a 0 are typically units that you can capture, with a couple of exceptions (Ashura’s Gool and Kreuz Wahrheit). I *think* that all units with a 255 are either protected from capture or are taken away from your roster when starting a New Game+ (saving their upgrades). You find out.

The **Capture** combobox allows you to change between “capture only” and “capture & use” for the unit. There are other factors that affect the capture status of a unit in the game, so it won’t be enough just changing this value. Notice that there’s not an option for “cannot capture” (which might be affected by the Essential value).

Reward is the money you get from destroying the unit. **Sell value** is calculated as 2x the Reward. **Repair cost** is what you have to pay if your unit is blown up. Note that the **max value** admitted for Reward and Repair is **65535**.

Size is the size of the unit. Notice you can pick size SS even though there are no SS units in the game. And if we are to believe the weapon descriptions, no one can target an SS unit.

Enhance parts is the number of enhance parts the unit can equip. **Movement** is self-explanatory. Note that you can have a natural movement of 15, but bear in mind that in some SRW games (like 64), making a unit able to move more than 16 spaces crashes the game.

The “**Is a building**” check is for buildings (duh). A unit with this property shouldn’t be able to attack and shouldn’t have visible properties in-map, but I haven’t tested it. Give it a try.

Terrain data panel

Terrain data

Ratings: Space Water Land Air

Type: ☒ Air ☒ Land ☐ Water ☐ Ground ☐ Hover

The **terrain ratings** are self-explanatory.

Terrain types can affect how a unit moves or can be just informative. **Air** allows flight, **Land** might not do anything (haven’t tried), **Water** would allow normal movement in water (haven’t tried), **Ground** is actually short for underground and allows digging (like Getter 2), while **Hover** allows hover movement (over water).

Unit stats panel

Stats

		Base value	Max value	Upgrades
<input type="text" value="Robot"/>	BODY HP	<input type="text" value="13200"/>	<input type="text" value="23100"/>	<input type="text" value="15"/>
<input type="text" value="Robot"/>	HEAD HP	<input type="text" value="8800"/>	<input type="text" value="15400"/>	
<input type="text" value="Robot"/>	ARMS HP	<input type="text" value="8800"/>	<input type="text" value="15400"/>	
<input type="text" value="Robot"/>	LEGS HP	<input type="text" value="8800"/>	<input type="text" value="15400"/>	
	Energy	<input type="text" value="300"/>	<input type="text" value="750"/>	<input type="text" value="15"/>
	Mobility	<input type="text" value="125"/>	<input type="text" value="218"/>	<input type="text" value="15"/>
	Armor	<input type="text" value="1500"/>	<input type="text" value="2625"/>	<input type="text" value="15"/>

The stats themselves are pretty self-explanatory. The **upgrades** comboboxes indicate how many times that part can be upgraded in the game. You can see a projected **max value** of each stat using the base value and applying the upgrades of your choice.

Next to each part’s HP value there’s a combobox that allows changing the **type** of each of these parts between “**unused**”, “**robot**” and “**ship**”.

Unused disables the part. An example of this can be seen in the Apsalus II:

		Base value	Max value
Robot ▾	BODY HP	10000	17500
Robot ▾	HEAD HP	6600	11550
Unused ▾	-----	-1	-1
Unused ▾	-----	-1	-1

Robot is the default one.

Ship changes the description of the part in-game:

Ship ▾	BODY HP
Ship ▾	CONTROLS HP
Ship ▾	WEAPONS HP
Ship ▾	ENGINES HP

Unit abilities panel

Unit abilities

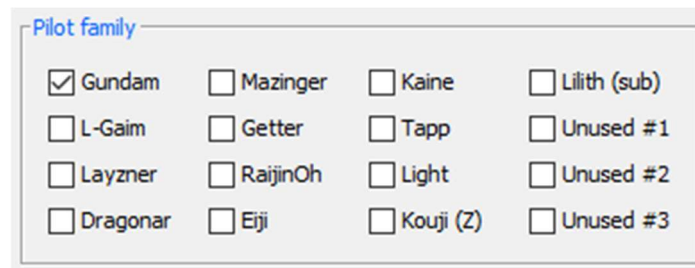
<input type="checkbox"/> Transform	<input type="checkbox"/> Double Image	<input type="checkbox"/> I-Field	<input type="checkbox"/> Mazin Power
<input type="checkbox"/> Combine	<input type="checkbox"/> Neo Getter Vision	<input type="checkbox"/> HP Regen S	<input type="checkbox"/> EWAC
<input type="checkbox"/> Separate	<input type="checkbox"/> Shin Mach Special	<input type="checkbox"/> HP Regen M	<input type="checkbox"/> V-MAX
<input type="checkbox"/> Repair	<input type="checkbox"/> Jammer	<input type="checkbox"/> HP Regen L	<input type="checkbox"/> V-MAX Red P.
<input type="checkbox"/> Supply	<input type="checkbox"/> Beam Coat S	<input type="checkbox"/> EN Regen S	<input type="checkbox"/> V-MAXIMUM
<input checked="" type="checkbox"/> Boarding	<input type="checkbox"/> Beam Coat M	<input type="checkbox"/> EN Regen M	
<input checked="" type="checkbox"/> Capture	<input type="checkbox"/> Beam Coat L	<input type="checkbox"/> EN Regen L	<input type="checkbox"/> Unit w/ Shield

Here, what you see is what you get. Some notes, though.

Even though we can activate / deactivate the **Transform**, **Combine** and **Separate** abilities, I haven't found what determines the units involved in these processes in the data, so... better not to touch these three.

The **Capture** ability is just for show. A unit with Capture can't capture anything. In order to capture another unit, your unit must have the **Boarding** ability (no need for Capture). This is most probably lazy programming and it pisses me off that I can't make a Zaku capture other unit without becoming a carrier. Oh well...

Pilot family panel



The image shows a window titled "Pilot family" with a grid of checkboxes. The first checkbox, "Gundam", is checked. The other checkboxes are: "Mazinger", "Kaine", "Lilith (sub)", "L-Gaim", "Getter", "Tapp", "Unused #1", "Layzner", "RaijinOh", "Light", "Unused #2", "Dragonar", "Eiji", "Kouji (Z)", and "Unused #3".

Pilot family			
<input checked="" type="checkbox"/> Gundam	<input type="checkbox"/> Mazinger	<input type="checkbox"/> Kaine	<input type="checkbox"/> Lilith (sub)
<input type="checkbox"/> L-Gaim	<input type="checkbox"/> Getter	<input type="checkbox"/> Tapp	<input type="checkbox"/> Unused #1
<input type="checkbox"/> Layzner	<input type="checkbox"/> RaijinOh	<input type="checkbox"/> Light	<input type="checkbox"/> Unused #2
<input type="checkbox"/> Dragonar	<input type="checkbox"/> Eiji	<input type="checkbox"/> Kouji (Z)	<input type="checkbox"/> Unused #3

These bits are meant for when you're reassigning pilots in the game. Both pilots and units have this configuration.

On the Units side, checking one of the families means that all pilots with the same check can board them. Basic example, Gundam units have a check for Gundam family and Gundam pilots also have it, so they can be assigned to them.

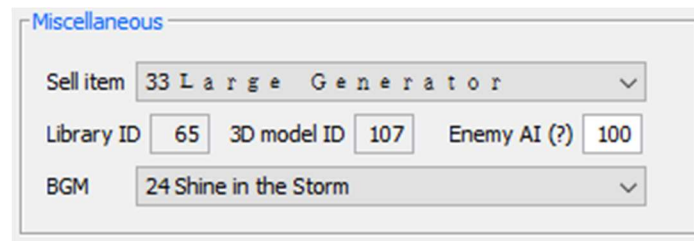
On the Characters side (in the Characters tab), it's a bit more tricky. The first 7 (Gundam to RaijinOh) work like explained above, but the next 6 (Eiji to Lilith) have special properties.

- **Eiji** gives access to units with the Eiji family AND Layzner family units.
- **Kaine**, **Tapp** and **Light** do the same, but give also access to Dragonar family units.
- Likewise, **Kouji (Z)** gives additional access to Mazinkaiser family units.
- **Lilith (sub)** gives access to L-Gaim family units **as a subpilot**. However, the L-Gaim family is just the default value. You can combine this family with any other family to turn a character into a subpilot for the second family.

For example, a character with Lilith and Gundam will be assignable as a subpilot for units of the Gundam family. Notice that the character can only be used as a subpilot then.

The last three families are unused because no unit or character uses them, but they work. You could, say, make all red units exclusive to Quattro using these unused families. Be creative with them.

Miscellaneous panel



Miscellaneous

Sell item 33 Large Generator

Library ID 65 3D model ID 107 Enemy AI (?) 100

BGM 24 Shine in the Storm

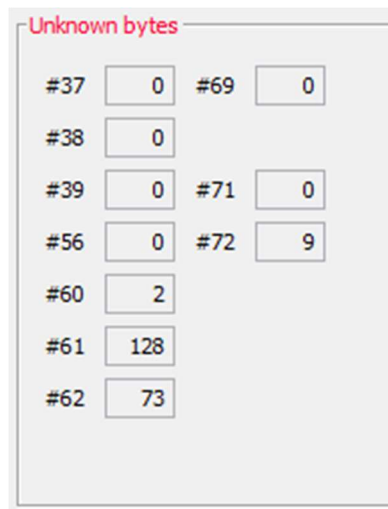
In this panel you can see a mix of several things for the unit.

- The **item** you'll get in the shop when scraping a unit.
- The **ID of the robot library entry** you'll get when you press the "Info" button in the unit stats screen.
- The **3D model** used for the unit. Funny thing, several units with several copies (like the Allones or the Dragons) use different 3D models.
- The **Enemy AI (?)** is another unclear field. All units that can be piloted by enemies (except duplicates of these units) have a value here from 50 to 250. 50 is typical for low-tier grunt units and 250 is typical for super-boss units. The values rise depending of how tough a unit is supposed to be. I think this is what determines how often an enemy will use MAP attacks or what kind of units they'll target.

A similar value can be found in the Characters tab. My theory is that the one in the Units tab is a *ceiling* value for the AI of the pilot. This limit can be seen for example in Haman (240) and the Qubeley (200).

- **BGM** lets you choose the music that will play when this unit is in battle. You can use any music from the game here, even the Level Up jingle. Do you want to make the GM and Ball use Meguriai as their BGM? Go ahead.

Unknown bytes panel



Unknown bytes	
#37	0
#38	0
#39	0
#56	0
#60	2
#61	128
#62	73
#69	0
#71	0
#72	9

These are bytes of the unit structure that I couldn't figure out completely. All of them are behind the safety lock for this reason.

- Bytes **#37**, **#38**, **#39** and **#69** are always 0, so no big deal there.
- Byte **#56** is set to 1 for several enemy grunt units that can be captured and the Elmeth. It *maybe* indicates that a unit can be piloted by a mid-boss?
- Byte **#60** has values from 1 to 10 and I have no idea what those values mean.
- Byte **#61** seems to indicate that a unit is the "normal" version (128, or 0x80 in hex) or an "alternate" version (192, or 0xC0). Alternates are duplicates of the same unit, which can go from being copies of a unit (Dragoons), player-versions of units (Qubeley Mk-IIs), enemy-versions of units (Mazinkaiser) or units that need an alternative model for some attack (Hyaku Shiki + Mega Bazooka Launcher).

In fact, it's most probably bit based. A value of 0x80 (1000 0000) could indicate that a unit has visible stats during a mission, whereas the 0x40 bit (0100 0000) could be the "alternate" indicator. The second nibble of the byte (---- XXXX) is likely used to extend the ID that can be seen in byte #62, but is never used because #62 never goes past 255.

- Byte **#62** is the same as the Library ID... until the Zeta Gundam, at which point the Library ID skips one number (turns into 72, whereas this byte gets value 71). This happens again for the Texas Mack (180 and 178). No idea what this is used for.
- Bytes **#71** and **#72** are an in-series ID. All units within a series are numbered in order here (Gundam is 0, Guncannon is 1, Guntank is 2, etc). Not sure what this is for exactly, maybe order in the library? Or is it for some sorting option in the Units list?

Weapons panel

The weapons panel has two sections:

The screenshot shows a 'Weapons' panel with the following fields and options:

- Upgrades:** 15 (dropdown)
- Inflation:** 2 (input)
- increases upgrade cost by:** 4000 (input)
- Weapon ID:** 841 (text)
- Weapon Name:** Tiger Boomerang (text)
- Animation ID:** 1 (input)
- BGM override:** 00 (no BGM) (dropdown)
- Base Power:** 3600 (input)
- Max Power:** 7300 (input)
- Weapon range:** 1 ~ 5 (inputs)
- Hit bonus:** 40 (input)
- Crit bonus:** 10 (input)
- Ammo:** -1 (input)
- EN cost:** 0 (input)
- Will req.:** 0 (input)
- Mounted on:** ARMS (dropdown)
- Combo ID:** -1 (input)
- Terrain ratings:**
 - Space: A (dropdown)
 - Water: B (dropdown)
 - Land: A (dropdown)
 - Air: A (dropdown)
- Unknown bytes:**
 - #01: 0 (input)
 - #10: 0 (input)
 - #21: 0 (input)
 - #23: 0 (input)
 - #24: 0 (input)
- Weapon type:**
 - ☒ Infight
 - ☒ Post-move
 - ☐ Gunfight
 - ☐ Beam
 - ☐ ???
 - ☐ Missile
- Other properties:**
 - ☐ NT Lv1
 - ☐ NT Lv5
 - ☐ Lv10
 - ☐ ???
 - ☐ ???
 - ☐ ???
 - ☐ Growth 2
 - ☒ Growth 3
 - ☐ Growth M
 - ☐ Event lock
 - ☐ Can't hit S
 - ☒ Hit building

The **header**, where you can set properties that apply to all weapons. These properties are just the **number of upgrades** available (which can go up to 15 without crashing the game) and the **inflation of upgrade cost**. Inflation is very simple, it adds $2000 * \text{inflation number}$ to the cost of each upgrade for weapons.

The **weapons list**, which is scrolled panel displaying all weapons assigned to the unit you're currently viewing. Each weapon is displayed in an individual panel that looks like the following one:

Starting from the top, you can see the **weapon ID** and the **name** of the weapon, which are purely informative.

The **Animation ID** is the number of the attack animation the robot will use. Animations are tied to the 3D models used by the unit, so you can't use an animation from a different robot. If you assign an incorrect animation for an attack (one that the unit's 3D model doesn't have), the battle sequence for that unit will be a black screen when it uses the attack, but the game won't crash (or it didn't for me). I don't recommend changing this.

BGM override is exactly that. You can set a BGM specifically for one attack. Examples of this are RaijinOh's and the Banpresto Original's finishers. Do you think Borot Rolling Crash should have a finisher BGM? Make it so.

Base power is the attack power (max 65535). **Max power** is the expected max value based on the number of upgrades and the growth type of the weapon (more on that later).

Weapon ranges work exactly as you'd expect. Minimum to the left and maximum to the right. Don't use a minimum higher than the maximum if you don't want to freeze your game. A high enough maximum range could potentially crash the game as well.

Hit bonus and **Crit bonus** can be negative. These values go from -128 to +127, but the game will display only between -99 and +99.

Ammo is the amount of ammunition the attack has. An ammo value of -1 means the weapon uses no ammunition. Don't try other negative numbers.

EN cost and **Will req.** are pretty self-explanatory. Notice that the max energy cost you can set to a weapon is actually 65535, which is a pretty absurd amount. Max will requirement you can input is 255, but obviously you can't reach anything past 150 in this game, so don't bother.

The **Mounted on** combobox makes the weapon dependant on the unit having the corresponding part not destroyed (otherwise the weapon can't be used). I haven't tested assigning a weapon to a part that is unused (say, Apsalus' arms), but I imagine that will make the weapon unusable.

The **Combo ID** is a field you should not touch. It's an identifier for a different section inside add02dat.bin where the configuration of combination attacks is held. The section contains the number of units needed for a combination attack to activate, a common ID for recognizing weapons of the same type and a part that *should* describe which specific pilots are needed for the attack. I never figured out the last part, so I didn't include this section in the tool, sorry.

The image shows two sections of a configuration tool. The 'Terrain ratings' section has four dropdown menus labeled 'Space', 'Water', 'Land', and 'Air', each with a value of 'A'. The 'Unknown bytes' section has five input fields labeled '#01', '#10', '#21', '#23', and '#24', each containing the value '0'.

Terrain ratings are the same as ever. The **unknown bytes** in weapons are always 0, so you shouldn't bother with those.

The image shows the 'Weapon type' section of a configuration tool. It contains eight checkboxes arranged in two columns. The first column has 'Infight' (checked), 'Gunfight' (unchecked), '???' (unchecked), and '???' (unchecked). The second column has 'Post-move' (checked), 'Beam' (unchecked), 'Missile' (unchecked), and 'MAP' (unchecked).

About **weapon types**, they're pretty well-known.

- **Infight** makes the weapon use the Melee stat of the pilot for its damage calculation.
- **Gunfight** makes the weapon use the Ranged stat. I think it's possible to have a weapon be both Infight and Gunfight types, but I'm not sure how the damage calculations would work.
- **Post-move** allows the weapon to be used after movement.
- **Beam** makes the weapon susceptible of being affected by beam coats / I-fields.
- **Missile** makes the weapon vulnerable to Jammer.

- **MAP** turns the weapon into a MAP weapon. Checking this option will make your weapon panel turn yellow-ish (for spotting MAP weapons more easily).

The screenshot shows the SRW Advance weapon editor for the 'Arounzer Inferno' weapon (ID: 894). The interface is divided into several sections:

- Weapon ID:** 894, Name: Arounzer Inferno
- Animation ID:** 3, **BGM override:** 00 (no BGM)
- Power:** Base 3500, Max 5750
- Weapon range:** 1 ~ 5
- Hit bonus:** 10
- Crit bonus:** 0
- Ammo:** -1
- EN cost:** 50
- Will req.:** 0
- Mounted on:** ARMS
- Combo ID:** -1
- Terrain ratings:** Space (A), Water (A), Land (A), Air (A)
- Unknown bytes:** #01 (0), #10 (0), #21 (0), #23 (0), #24 (0)
- Weapon type:**
 - ☐ Infight
 - ☐ Post-move
 - ☒ Gunfight
 - ☐ Beam
- Other properties:**
 - ☐ NT Lv1
 - ☐ NT Lv5
 - ☐ Lv10
 - ☐ Lv15
 - ☐ Growth 1
 - ☐ Growth 2
 - ☐ Growth 3
 - ☐ Growth M
 - ☐ Growth 1
 - ☐ Combo
 - ☐ Event lock
 - ☐ Can't hit S
 - ☒ Hit building

I never found how to change the type of MAP weapon used, so I'd stick to the originals here unless you know what you're doing.

There's a couple of bits between the types that are never used and I have no idea of what they do. Feel free to experiment.

This is a close-up of the 'Other properties' section from the weapon editor. It contains the following options:

- ☐ NT Lv1
- ☐ NT Lv5
- ☐ Lv10
- ☐ Lv15
- ☐ Growth 1
- ☐ Growth 2
- ☒ Growth 3
- ☐ Growth M
- ☐ Growth 1
- ☐ Combo
- ☐ Event lock
- ☐ Can't hit S
- ☒ Hit building

Finally, in the **Other properties** tab we have a mixed bag:

- **NT Lv1** and **NT Lv5** gives the weapon the requirement of the corresponding Newtype levels in order to be used.
- **Lv10** and **Lv15** are pilot level requirements, just like some attacks in SRW Advance. They never got used for any weapon in the game.
- **Growth 1, 2, 3, and M**, are the different growths a weapon can have through its upgrades. You can consider 1 to 3 as S, M and L, meaning that growth 3 will grow the most. Growth M is used for MAP weapons and has the smallest growth of all.
- **Combo** will make the weapon usable only when the conditions set by the Combo ID are met (enough adjacent units with the required weapon) or at least the weapon will be hidden until then.

I haven't tested this, but it's possible that if you uncheck this option and leave the Combo ID with a non-negative value, the weapon will be visible all the time but you won't be able to use it until it's "legally" possible to do so. Feel free to test that.

Also, attacks marked as Combo will be highlighted in a darker gray in the UI, to help identify them quickly:

Weapon ID: 698 Final Dynamic Special

Animation ID: 4 BGM override: 00 (no BGM)

	Base	Max	Weapon range	Hit bonus	Crit bonus
Power	6500	7650	1 ~ 1	30	0

	Ammo	EN cost	Will req.	Mounted on	Combo ID
	-1	100	120	BODY	28

Terrain ratings

Space	Water	Land	Air
A	A	A	A

Unknown bytes

#01	#10	#21	#23	#24
0	0	0	0	0

Weapon type

☒ Infight ☒ Post-move

☐ Gunfight ☐ Beam

Other properties

☐ NT Lv1 ☐ ??? ☐ Growth 2 ☐ Event lock

☐ NT Lv5 ☐ ??? ☐ Growth 3 ☐ Can't hit S

- **Event lock** means the weapon is not available until certain story events have taken place in the game. This is another thing that is not available to edit here, so I wouldn't start locking weapons behind events for now.

Weapons behind an event lock will be highlighted in blue-ish for easy spotting:

Weapon ID: 697 Getter Final Crash

Animation ID: 5 BGM override: 00 (no BGM)

	Base	Max	Weapon range	Hit bonus	Crit bonus
Power	5000	6150	1 ~ 1	30	0

	Ammo	EN cost	Will req.	Mounted on	Combo ID
	-1	80	120	ARMS	-1

Terrain ratings

Space	Water	Land	Air
A	A	A	A

Unknown bytes

#01	#10	#21	#23	#24
0	0	0	0	0

Weapon type

☒ Infight ☒ Post-move

☐ Gunfight ☐ Beam

Other properties

☐ NT Lv1 ☐ ??? ☐ Growth 2 ☒ Event lock

☐ NT Lv5 ☐ ??? ☐ Growth 3 ☐ Can't hit S

Notice that the Event lock color overrides other colors, such as MAP or Combo.

- **Can't hit S** means the weapon can't hit units of size S. Notice that, if what the description given by the game is true, no weapon can hit a unit of size SS. Unfortunately you can't make weapons unable to hit other sizes, but I guess that would be a bit ridiculous.
- **Hit building** means the weapon can be used to attack units with the "Is a building" flag.

There are 4 bits in the properties that I couldn't figure out what their effect was. You can make your own tests.

Characters tab

The characters' tab is a bit less complicated than the previous one, but still needs a bit of explaining.

The screenshot shows the 'Characters' tab interface. At the top, there are dropdowns for 'Character' (000 Shinobu Fujiwara), 'Personality' (01 Super strong), 'Robot series' (00 Dancouga - Super Beast Machine God), and 'Library ID' (0). Below these are 'Ally / Enemy' (1 Ally), 'Enemy AI (?)' (0), and 'Portrait / Battle lines ID' (290). Further down are 'Skill parts' (2), 'Skill Aces' (3 Melee + Ranged), and 'Stat growth' (2 Melee + Defense). The 'Stats' section shows a table with Base and Max values for Melee, Defense, Accuracy, Ranged, Skill, and Evasion. The 'Skill levels' section shows a table with levels from Lv1 to Lv9 for (Cyber) NT, Potential, Support, and Command. The 'Skills list' section has checkboxes for various skills like 'NOT a Cyber NT', 'Support ATK', 'Counter', etc. The 'Personality will modifiers' section shows modifiers for Hit target, Damage taken, Miss attack, Enemy shot down, Evade, and Ally shot down. The 'Spirit commands' section shows a table with Command, SP cost, and @ Lv. The 'Unknown Bytes' section shows a table with hex values and their corresponding decimal values. The 'Pilot family' section has checkboxes for various pilot families like Gundam, Mazinger, Kaine, etc.

Starting from the top left, we get the some basic character information that I couldn't group anywhere else:

This close-up screenshot shows the top part of the 'Characters' tab interface. It includes the 'Character' dropdown (0059 R a m b a R a l), 'Personality' dropdown (06 Mid-boss), 'Robot series' dropdown (04 Mobile Suit Gundam), 'Library ID' (53), 'Ally / Enemy' dropdown (2 Enemy), 'Enemy AI (?)' (170), 'Portrait / Battle lines ID' (54), 'Skill parts' dropdown (4), 'Skill Aces' dropdown (3 Melee + Ranged), and 'Stat growth' dropdown (0 Mediocre).

The **Character combobox**, much like the Unit combobox in the other panel, allows you to navigate through the different characters.

Robot series is the robot series that will be displayed in the character's library entry.

The **Ally / Enemy combobox** gives you the choice between (nothing), Ally, Enemy or Both. Characters that are both allies and enemies are characters that can be recruited to your side, like Aina. Notice that Norris has the same classification, but the event to make him join was not implemented. It doesn't seem to serve any purpose ingame, so it might have been used for some quick classification of characters for whatever tools the developers used when making the game.

The **Enemy AI (?)** field was explained already in the Miscellaneous panel of the Units tab.

The **Skill parts** are the number of skill parts that can be equipped by the character.

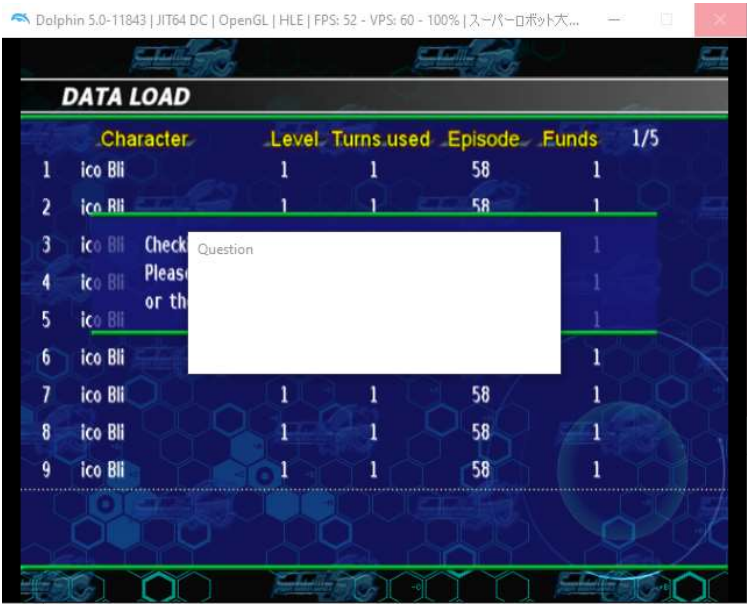
The **Skill Aces combobox** shows which attack aces are available for the character: Melee, Ranged, both or none. All characters have both by default and only a few ones have the Melee one disabled, like your ship captains and Bowie.

The **Personality combobox** allows you to select the personality type of the character. The personality of choice can have several effects. The most obvious one are the will gains depending on your actions during an scenario map. These are displayed next to it for your convenience:



This applies to personalities 0 to 7, however the next ones (8 to 10) have the added effect of hiding stats in the character stats screen (well, at least it does for 9 and 10).

The last two personalities, *09 Subpilot* and *10 NPC* count as character categories. These are apparently used by the game to build internal lists of characters. I tested turning one subpilot into a regular pilot and two NPCs into subpilots and the result was a crash in the screen for loading a saved game.



Be extra careful if you try changing these categories.

The **Library ID** works exactly like the one for Units, but this is used for the Character Library.

The **Portrait / Battle lines ID** indicates the ID used for the files inside the file bpilot.pak. This is where the portraits and battle script files are located. You shouldn't need to change this.

The **Stat growth** affects how the character stats will grow when gaining levels. This will be reflected in the Stats panel depending on the growth schema chosen:

Stat growth: 2 Melee + Defense

Stats

	Base	Max		Base	Max		Base	Max			
Melee	153	300	S	Defense	110	241	A	Accuracy	110	241	A
Ranged	148	246	B	Skill	170	268	B	Evasion	88	186	B

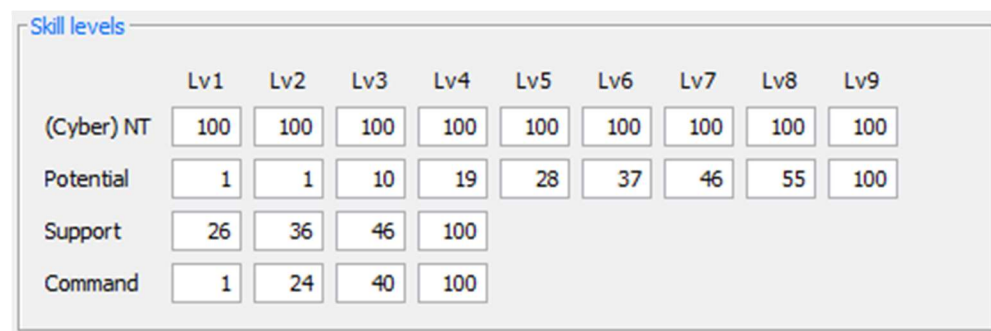
Possible stat growths: B gives +1 per level, A gives +3 every 2 levels (one level +1, next level +2) and S gives +2 per level.

Base stats need no explanation, but it's important to note that the maximum value for them is 255.

To illustrate the growths, Base stats have a Max value next to them showing what will they look like at level 99 depending on the stat growth schema.

Like with the personality, there's a special stat growth schema (8) that is given only to NPCs. It's unclear if this affects anything in the game.

Skill levels panel



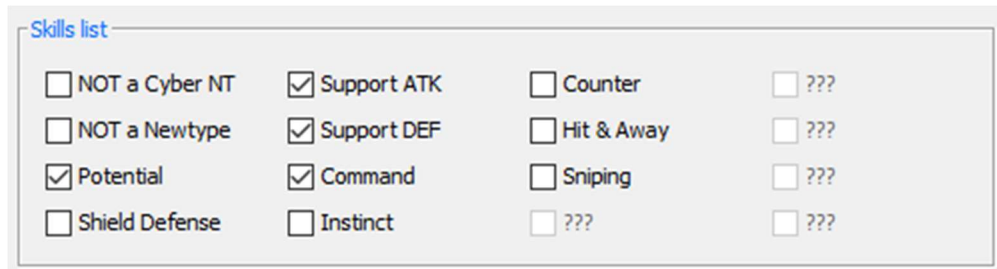
	Lv1	Lv2	Lv3	Lv4	Lv5	Lv6	Lv7	Lv8	Lv9
(Cyber) NT	100	100	100	100	100	100	100	100	100
Potential	1	1	10	19	28	37	46	55	100
Support	26	36	46	100					
Command	1	24	40	100					

The Skill levels panel shows the level at which a character can learn each level of the leveled skills in the game, those being Newtype / Cyber Newtype, Potential, Support and Command.

A skill level set to be learned at character level 100 means the character won't ever learn that skill level, since the maximum level a character can achieve legally is 99.

Setting up these levels is not enough for the character to gain access to the skills though. All of these skills require marking options in the next panel.

Skills list panel



The screenshot shows a panel titled "Skills list" with a grid of checkboxes for various skills. The skills are arranged in four columns and four rows. The first two columns contain skills that are either checked or unchecked, while the last two columns contain skills that are currently unchecked and marked with "???" indicating they are unknown or unavailable.

Skills list			
<input type="checkbox"/> NOT a Cyber NT	<input checked="" type="checkbox"/> Support ATK	<input type="checkbox"/> Counter	<input type="checkbox"/> ???
<input type="checkbox"/> NOT a Newtype	<input checked="" type="checkbox"/> Support DEF	<input type="checkbox"/> Hit & Away	<input type="checkbox"/> ???
<input checked="" type="checkbox"/> Potential	<input checked="" type="checkbox"/> Command	<input type="checkbox"/> Sniping	<input type="checkbox"/> ???
<input type="checkbox"/> Shield Defense	<input type="checkbox"/> Instinct	<input type="checkbox"/> ???	<input type="checkbox"/> ???

Some of these bits are obvious, but a couple of them require an explanation.

The bits for enabling Newtype / Cyber NT abilities work backwards from what anyone would expect. Instead of enabling Newtype or Cyber NT, the bits disable these abilities. A character without the “**NOT a Cyber NT**” and “**NOT a Newtype**” checks marked would be both a Newtype and a Cyber NT if we gave him a skill level in that.

The **Support ATK** and **Support DEF** work like you would expect. The weird thing is that this game only has one support value (traditionally it’s two separate ones). To circumvent this difference, the game gives the character their levels in support at the beginning of each phase, only if they have the corresponding bit, depending on if they’re attacking or defending.

Playable characters all have both ATK and DEF, but many enemies (bosses) don’t have Support DEF, meaning they won’t defend their allies when you attack, but will give them cover fire.

The rest of the bits in this panel work as you would expect. Five skills are unknown, as nobody has them, but I haven’t tested all of them so maybe there’s some hidden skill there.

Spirit commands panel

Spirit commands

	Base	Max	
SP	50	148	B

	Command	SP cost	@ Lv.
#1	03 Flash	10	1
#2	05 Focus	10	1
#3	06 Direct ...	25	5
#4	17 Hot Blood	40	25
#5	21 Awaken	55	44
#6	24 Soul	55	49

This one's pretty straightforward. Worth of note is that, even though base stats have a max value of 255, max SP is 65535 (just in case you need SP). Growth for SP is always B.

The **@ Lv.** column indicates the character level at which the SP command of its row is learned.

Pilot family panel

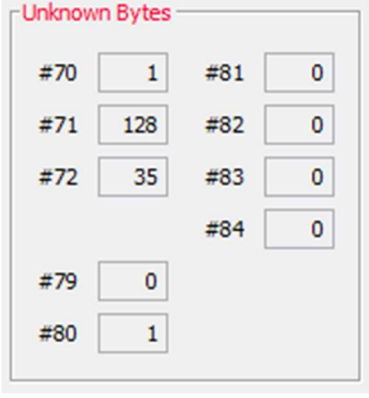
Pilot family

<input checked="" type="checkbox"/> Gundam	<input type="checkbox"/> Mazinger	<input type="checkbox"/> Kaine	<input type="checkbox"/> Lilith (sub)
<input type="checkbox"/> L-Gaim	<input type="checkbox"/> Getter	<input type="checkbox"/> Tapp	<input type="checkbox"/> Unused #1
<input type="checkbox"/> Layzner	<input type="checkbox"/> RaijinOh	<input type="checkbox"/> Light	<input type="checkbox"/> Unused #2
<input type="checkbox"/> Dragonar	<input type="checkbox"/> Eiji	<input type="checkbox"/> Kouji (Z)	<input type="checkbox"/> Unused #3

Already covered in the Unit tab. Reminder that some of these have special properties: the ones named after characters give additional access to their series and the one for Lilith (sub) can be combined with any other family to make a character a subpilot of said family (that can be assigned just like Lilith).

Also keep in mind that the Unused family bits are usable.

Unknown bytes panel



The 'Unknown Bytes' panel displays a grid of 14 byte values, organized into two columns. The left column contains bytes #70 through #80, and the right column contains bytes #81 through #84. Each byte is represented by a label (e.g., #70) followed by a text input field containing its value.

Byte	Value
#70	1
#71	128
#72	35
#79	0
#80	1
#81	0
#82	0
#83	0
#84	0

Byte #70 has values from 1 to 10. I have no idea of what this does.

Bytes #71 and **#72** work like bytes #61 and #62 in the Units tab.

#71 indicates if a character is “normal” or an “alternate”, probably indicating if they have a library entry enabled (since grunts have this value set to 0). There are some exceptions like the enemy version of Gascon.

#72 (and the lower half of #71) gives us the same value that can be found in the Library ID field, and this time it’s always the same... until we reach Fairey and this value gets 2 positions behind. Another difference with the Library ID is that characters with no entry have a 0 here (the Library ID for those would be -1).

Bytes #79 and **#80** form a pair, #79 being always 0 and #80 going from 0 to 3. All characters who don't join you have a 0. The rest have either 1 or 2. Fairey has a 3.

Interesting of note: some characters who don't actually join you have a value different than 0, like Watkein or Rowan, and even some enemies like Ramba Ral and Norris are like this. It seems like a pattern, but still unclear.

Bytes #81 and **#82** also form a pair, again #81 being always 0. They work like bytes #71 and #72 of the Units tab, being an in-series ID for the characters. Alternates and characters with no library entry set this value to 0, which could be confused with the first character of the series, but since the actual first character has other bytes with values (#71 and #72), I guess it doesn't matter.

I imagine these values are used for some kind of sorting, but not sure where.

Bytes #83 and **#84** are always 0.

Closing

That was longer than I expected this document to be, sorry about that. Hopefully I gave more answers than questions here.

If you find some absurd error, find out what some of the unknown bytes do or discover some effect I was not aware of, let me know and I'll update the tool accordingly.

Happy editing and have fun.