CTB by Charlie Lee v2.13 - Instructions

Table of contents.

Demo Instructions.

Run into the frogs for a battle.

Advanced Targetting.

Use Shift to switch between the party and the monsters when using Reflect and other "any scope" skills.
Use Shift to switch between single and "all" target when using Double Slash and other "single/all scope" skills.

Configuration of Individual Battle Commands.

Let us assume that you want to create "Black Magic" and "White Magic". Well, create two elements named "CMD White Magic" and "CMD Black Magic", and add the proper element to all the skills in your database that belong to either of the two "categories".
Finished. Yes. Simple, isn't it?
Remember, if "Name" is you category, use an element called "CMD Name". "Name" will end up in the command window of the actors, and it will be displayed only if that actor has at least one skill of that category.

Configuration of SuperArts.

In order to use superarts you need two things: an element "Superart" and a second element "CMD Superart". Every skill that has to be a superart needs to be "tagged" with these two elements. It will appear under the Superart command and will be available only when the bar is filled.

So far, the following fill modes have been implemented.

Rage: gain Superart points by inflicting damage.
Pain: gain Superart points by suffering damage.
Emphaty: gain Superart points by damage inflicted to allies.
Category: gain Superart points when skills of determined categories are used
Critical: gain Superart points when with low HP.

You can switch between the modes from the Status Scene, by pressing SHIFT, but in order to see that, you need a custom Status Scene script.

Configuration of Summons.

There are little differences between the configuration of the different types of summons.

Type 1 and 2 summons.

STEP 1
Type-1 summons do not remove the party (think about Warcraft), while type-2 do (think about FFX). In order to configure a summon, go in the common events tab of the database and create a new common event, call it as you like, and put a call script instruction like this:

```
"$game_party.summon(<type>,["actor1 name","actor2 name",...])",
```

where <type> is 1 or 2.

Examples:

```
$game_party.summon(1,["Kerya"])          # Type 1, summons Kerya
$game_party.summon(2,["Kerya","Adel"])    # Type 2, summons Kerya and Adel
```

You can put the command in more than one line:

```
$game_party.summon(2,["Kerya",
"Adel"])
```

VERY IMPORTANT: do not break the lines in the middle of an actor's name:

```
$game_party.summon(2,["Black
Night"])
```

is WRONG and will not work.

STEP 2
Then you must create a skill in the skills tab of the database, which your characters will use to summon, and make the common event field of this skill point to the common event you have just created.
All the summon skills must be tagged with an element whose name matches the name configured in the script as SUMMON_NAME, the default is "Summon". This name will not appear in the battle command window, where you will see the category name of the summon skills.

You will also need a skill to make your "Aeons" retire.

1. Copy the common event "Retire" from the demo;
2. create a skill "Retire" whose common event field points to the "Retire" common event;
3. make sure that all the "Aeons" have access to the skill "Retire". I recommend to create a category with the element "CMD Retire" and tag the skill with it. This will show a command Retire apart from the other skills.

STEP 3
Finally, in order to correctly use your "Aeons", visualize their stats and making them learn skills (in the proper Scene), you need to add them to your party. Do it by using event script commands like the following:

```
$game_party.aeons.
push($game_actors[13])
$game_party.aeons.
push($game_actors[14])
```

NOTE: this step is not necessary if you have `AUTO_ADD_ACTORS = true`

Type-3 summons.

STEP 1
Type-3 summons are used for transformations. Transformed actors will be removed and their new form will appear on the battlefield. Their form will return normal only after the end of the battle. In order to configure a type-3 summon, go in the common events tab of the database and create a new common event, call it as you like, and put a call script instruction like this:

Example:
```
$game_party.summon(3,["Ice Golem"])         # Type 3, transform into Ice Golem
```

You must specify type 3 and put only one actor in the list.

STEP 2
Then you must create a skill in the skills tab of the database, which your characters will use to transform, and make the common event field of this skill point to the common event you have just created.
Also type-3 summon skills must be tagged with an element whose name matches the name configured in the script as SUMMON_NAME, the default is "Summon". This name will not appear in the battle command window, where you will see the category name of the summon skills.

Type-3 summons do not use a "Retire" skill.

STEP 3
Finally, in order to correctly use your "Transformations" and visualize their stats (in the proper Scene), you need to add them to your party. Do it by using event script commands like the following:

```
$game_party.transformations.
push($game_actors[13])
$game_party.transformations.
push($game_actors[14])
```

NOTE: this step is not necessary if you have `AUTO_ADD_ACTORS = true`

Switching Party Members.

In order to use the switching feature you need a common event and a skill.
Create the common event, call it as you like, and put the following call script instruction in it:

```
$scene.switch()
```

Afterwards, create the new skill, and configure it so that it calls the common event you just created. Don't bother configuring the other parameters as they will not be used. Only note down the ID of this skill. Do not forget to assign it a category (i use CMD Switch) and to make the actors have it in their list of available skills.

Finally go in the configuration page of the script and locate this section:

```
#------------------------------------------------------------------------------
# PARTY ACTORS SWITCHING                                                       #
#------------------------------------------------------------------------------
SWITCHING=true
SWITCHED_ACTOR_ANIMATION_ID=2
SWITCHED_ACTORS_EXP_MULTIPLIER=0.75 # Applied to actors who are
# in the backup party as the battle ends
SWITCH_SKILL_ID=125
```

Put the ID of the skill in the last line. Configure the other parameters as you like.

In order to add/remove backup actors use a script command like this within some event:
`$game_party.add_backup_actor(ID)`
`$game_party.remove_backup_actor(ID)`
where ID is the id of the actor in the database.
If you try to add a backup actor that is already in the backup party, nothing will happen.
If you try to remove a backup actor that is not in the backup party, nothing will happen.

Steal and Thievery.

The user can define the objects that can be stolen from monsters. For each monster a common item and a rarity can be configured. Rarities can be stolen only when the character which uses the Steal skill has the Thievery state. A single item can be stolen each time and there is no limit to the number of times the user can steal from a monster. The success is random and depends on the comparison between the agilities of the character and the enemy.
You will find a hash definition like this one in the configuration:

```
ENEMY_OBJECTS={
35 => [1,19],
34 => [4,15]
} # {enemy_id => [item_id, rarity_id]}
```

which I think is self-explanatory. If an enemy does not figure in the hash the user won't steal anything.
You always have to specify an item_id, but if you don't want a rarity use "nil" in the place of the rarity_id, example:

```
34 => [4,nil]
```

Flying Enemies.

Flying enemies cannot be hit by physical attacks unless the attacker uses a ranged weapon (a bow, a gun, etc). Put the IDs of flying enemies in the `FLYING_ENEMIES` array in the configuration.

Graphics.

There are lots of possibilities for those who want to customize the graphic layout of the BS.

Skills and Items windows.

```
#-------------------------------------------------------------------------------
# SKILL and ITEM WINDOW                                                        #
#-------------------------------------------------------------------------------
SKILL_ITEM_WINDOW_WIDTH=480
SKILL_ITEM_WINDOW_HEIGHT=24*6+32      # 112
SKILL_ITEM_WINDOW_X=32                # 160
SKILL_ITEM_WINDOW_Y=96                # 368
SKILL_ITEM_WINDOW_ROWSIZE=24          # 16
SKILL_ITEM_WINDOW_WINDOWSKIN_OPACITY=160
SKILL_ITEM_WINDOW_TEXT_RGB=[0,0,0]
SKILL_ITEM_WINDOW_SKIN="window_gris"             # Put "" if unused
```

You can personalize their size and position, as well the text size, the opacity, the text color (use the RGB values) and the windowskin name.

Switch-like windows.

```
#-------------------------------------------------------------------------------
# SWITCH WINDOW                                                                #
#-------------------------------------------------------------------------------
SWITCH_WINDOW_WIDTH=480
SWITCH_WINDOW_HEIGHT=112
SWITCH_WINDOW_X=160
SWITCH_WINDOW_Y=368
SWITCH_WINDOW_ROWSIZE=16
SWITCH_WINDOW_WINDOWSKIN_OPACITY=255
SWITCH_HELP_WINDOW_WINDOWSKIN_OPACITY=200
# The following categories of skills will use the settings for the switch window
# in order to not cover the help window
SWITCH_LIKE_CATEGORY_NAMES=["Summon","Transform"]
```

The term refers to the Switch window and the skills windows used for certain categories of skills as specified in

```
SWITCH_LIKE_CATEGORY_NAMES=["Summon","Transform"]
```

This windows have their own settings (size and position, text size, opacity) because when used, a help window is typically shown with additional information (stats of the summons, or of the backup party members).
You can set the opacity of the help window as well.

Actor Command window.

```
#-------------------------------------------------------------------------------
```

```
# ACTOR COMMAND WINDOW                                                          #
#------------------------------------------------------------------------------
ACTOR_COMMAND_WINDOWSKIN_OPACITY=200
LEFT_ACTOR_COMMAND_WINDOWSKIN_OPACITY=220
ACTOR_COMMAND_BG_PICTURE_NAME="actor_command_bg_2_bw2" # Put "" if unused
ACTOR_COMMAND_WINDOW_WIDTH=160 # 160, 128 recommended with status window style=2
ACTOR_COMMAND_CURSOR_MEMORY=true
ACTOR_COMMAND_WINDOW_TEXT_RGB=[255,255,255]
ACTOR_COMMAND_WINDOW_TEXT_B_RGB=[0,0,0]
```

You can decide its opacity, as well as the opacity of the left command window (the one showing Defend, Skip, Escape and so on).
The command window may use a background image as well.
You can decide the width, text color and text border color (RGB values).
Finally, you can set the window to use cursor memory (on a per-character basis).

Face of the active battler.

```
#------------------------------------------------------------------------------
# BATTLER WINDOW                                                               #
#------------------------------------------------------------------------------
BATTLER_FACE_VISIBLE=true
BATTLER_FACE_SMALL=true
BATTLER_WINDOW_Y=301
BATTLER_WINDOW_X=91
BATTLER_FACE_SIZE=44
BATTLER_FACE_WINDOWSKIN_OPACITY=0
```

The image displaying the active battler's face uses the picture Graphics/Faces/<character_name.png> or Graphics/Faces/<character_name-small.png>, depending on the value of `BATTLER_FACE_SMALL`.

It can be enabled/disabled, moved and resized.
You can also set the windowskin opacity.

Faces of the actors in the battle status.
The same picture or a smaller one (if available) called Graphics/Faces/<character_name-small.png> is used in the status. It can be disabled, with

```
STATUS_FACE_VISIBLE=false
```

in the configuration.

Turns window.

```
#------------------------------------------------------------------------------
# TURNS WINDOW                                                                 #
#------------------------------------------------------------------------------
# Configuration of the turns window
BATTLERS_QUEUE_SIZE=15#12          # Number of displayed turns
```

```
TURN_HEIGHT=15#20                  # Height of each box
TURN_USE_PICTURES=true          # Use pictures for the party actors
TURN_USE_PICTURES_ENEMIES=true  # Use pictures for the enemies
UPDATE_ONLY_CHANGES=true
TURNS_WINDOW_Y=80               # Y-coordinate for the turns window
TURNS_WINDOW_X=526              # X-coordinate for the turns window
TURNS_BG_NAME="turns_bg"
TURNS_WINDOW_VERTICAL=true
# The following values are used when TURNS_WINDOW_VERTICAL is false
TURN_WIDTH=40                     # Width of each box
TURNS_WINDOW_H_Y=64               # Y-coordinate for the turns window
TURNS_WINDOW_H_X=0                # X-coordinate for the turns window
TURNS_WINDOW_HEIGHT=64
```

The turn bars use Graphics/Pictures/Turns/<character_name.png> and can be disabled with

```
TURN_USE_PICTURES=false.
```

In that case a block with the character's name is displayed automatically. Of course, if you want personalized graphics, you should turn on the pictures and put yours in Graphics/Pictures/Turns/.
Note that you can also personalize the block pictures shown when you use text instead of pictures: they are Graphics/Pictures/Turn-1.png , Turn-2.png and Turn-3.png.
Pictures can be used for the enemies as well. The name of the picture must match the name of the enemy, if a picture cannot be found, the old plain block with the monster's name is shown. Use

```
TURN_USE_PICTURES_ENEMIES=false
```

to disable pictures for the enemies.
You will obtain better results if you use images with the exact size they will appear on the screen, so that RMXP will not perform any resize. The default size for the turn images is 84x20.

You can customize the position of the turn bars window and the number of turns displayed.
Use TURNS_WINDOW_VERTICAL to decide whether the turns will be aligned in a vertical or horizontal fashion.

If you want to get rid of the whole window. Put the following values in the configuration.

```
BATTLERS_QUEUE_SIZE=1
TURNS_WINDOW_Y=480
TURNS_WINDOW_X=0
TURNS_WINDOW_VERTICAL=true
```

The turns window may use a background image, put it in the Graphics/Pictures/ folder and configure it in this way:

```
TURNS_BG_NAME="turns_bg".
```

Report window.

```
#-------------------------------------------------------------------------------
# REPORT WINDOW                                                                #
#-------------------------------------------------------------------------------
BATTLE_REPORT_ROW_SIZE=60
```

```
BATTLE_REPORT_WINDOWSKIN_NAME="001-Black01.Red"
BATTLE_REPORT_WINDOW_WINDOWSKIN_OPACITY=0
BATTLE_REPORT_BG_PICTURE_NAME="report"
BATTLE_REPORT_BG_PICTURE_OPACITY = 128
BATTLE_REPORT_BACKGROUND_NAME="report_bg"
BATTLE_REPORT_BACKGROUND_OPACITY=200
```

The battle report window can show each character informations in a custom block picture. You can change the windowskin's name and opacity, the block picture's name and opacity, the size of each block.
You can use a background image and decide its opacity.

Battle Formulas Basics.

Physical Attacks

The result will depend linearly on the attack value of the used weapon, and on one parameter of the user: the strenght for close combat weapons, and the dexterity for ranged weapons. In formulas:

attack=ATK*STR or *attack*=ATK*DEX.

At this point, the algorithm uses the value configured as ATTACK_DAMAGE_RANGE_CORRECTION which gets multiplied to the attack power, obtaining the new attack power:

attack=*attack**ATTACK_DAMAGE_RANGE_CORRECTION

The target's pdef value will be substracted from the attack to get the damage:

damage=*attack*-pdef.

Skills

The power of the skill is used in combination with the caster's parameters, using the -F modifiers (ATK-F, STR-F, INT-F, DEX-F, AGI-F) in the database skills tab.
For example, for a skill with power=40, ATK-F=35, INT-F=80, the attack power will be:

attack=40*(0.35*caster's ATK value + 0.80*caster's INT value).

If all the modifiers are null, the skill's power directly represents the attack power:

attack=40.

At this point, the algorithm uses the value configured as SKILLS_DAMAGE_RANGE_CORRECTION which gets multiplied to the attack power, obtaining the new attack power:

attack=*attack**SKILLS_DAMAGE_RANGE_CORRECTION

The PDEF-F and MDEF-F values in the database are used to decide how much of the skill's damage can be absorbed with physical and/or magical defense.
If a skills deals a damage of 500 and you have PDEF-F=30 and MDEF-F=25, then up to 150 points can be absorbed by the physical defense, and up to 125 points by the magical defense of the target.

Beyond skills' power limits.

The ExponentialDamage element can be used to change the meaning of the power value of a skill. For a skill tagged with this element the actual power will be e^(power/10.0).
Here is a reference table:
10 --> 2
15 --> 4
20 --> 7
25 --> 12
30 --> 20
35 --> 33
40 --> 54

45 --> 90
50 --> 148
60 --> 403
70 --> 1096
80 --> 2980
90 --> 8103
100 --> 22026
110 --> 59874
120 --> 162754

As a rule of thumb, each 1 point increment means a 10% increment in the damage (70-->1096, 71-->1211).

The DamageX10 element can be used to multiply the skill's power by ten.

Blue Magic.

An element name can be defined with

```
BLUE_MAGIC_NAME="Blue Magic"
```

that allows skills tagged with that element to be learned from enemies, using the skill named as specified in

```
DRAKOKEN_NAME="Drakoken"
```