# **Default Battle System (DBS) Guide** for **RPG Maker 2**

---

_

Table of Contents:

---
_

## I. Version History

**Version 0.4**: Completed 4-23-06. Finished the guide up to a good temporary stopping point - would like feedback from you guys!

**Final Version**: Completed 10-28-06. I didn't actually add anything. I did go through and revise it though. It just took me too long to do what I've already done in it, and now I'm quite certain that I won't ever make any more progress with it. If anyone ever feels like they are capable of continuing where I left off and **actually wants to,** please do so. I think what I've done here should be able to "get the ball rolling" for you though. ;)

---
_

## II. Legal Information

---
_

## III. Preface

I've written this guide because I believe it will help a great many people working with RPG Maker 2. It is designed to show and teach you exactly how to do all the simple, and many complex, things with RPG Maker 2's (RPGM2) Default Battle System (DBS) including overhauling the DBS's current formulas, adding an accuracy/evasion formula, eliminating the default stats and making your own, making perfect Abilities/Items/Direct Effects/Scripts/Events to do all sorts of basic and complex things, making failing to flee consume a turn, using items directly from the bag in the DBS, Limit Breaks (from Final Fantasy 7 and 8) with various initiating requirements, and a pseudo Chrono Cross/Tactics Ogre magic system where MP gradually increases as battle progresses. This should not only help you to better understand RPGM2's DBS, but also teach you how to more or less make your own battle system out of it.

If you begin to have trouble on Step #1, #2, or #3, I suggest you visit Neo Samurai/Doshi Makesuke's 'Beginner RPGM2 FAQ' and Doan the Nado's 'World Creation Guide' at http://www.doanthenado.com/guides.htm and doyleman's 'Script Tutorial' at http://www.geocities.com/nate_dog7_7/ST.htm . I know many newbies are likely to look to my Guide as the be-all, end-all solution to all of RPGM2's difficulties when, in fact, it's not. I do not assume much of you, but I do assume that you have learned the common, basic uses of Scripts and Events (like basic townspeople) as well as a basic understanding of RPGs in general (stats, items, equipment, abilities, etc.). In my opinion, Neo Samurai/Doshi Makesuke's 'Beginner RPGM2 FAQ', Doan's 'World Creation Guide', and doyleman's 'Script Tutorial' are the most thorough and straight-forward Guides for beginning RPGM2, and I'm quite sure that after you do what's in their Guides you will have a much easier time doing what's in my Guide.

Lastly, if you would like further help with RPG Maker 2, or actually anything game design/program related, there's really only one place you should go:
**DOAN'S DOMAIN - BY FAR THE BEST PLACE ON THE NET!!!!!!!**

**Doan's Domain**: http://doansdomain.proboards27.com/

---
_

**IV. Terminology**

In this Guide and at **Doan's Domain**, you are bound to hear many of the following terms, and in order to understand this Guide, you will have to know what they mean. Fortunately for you, I've made a section specifically for the meaning of these terms so that if you find an unfamiliar term you can quickly and easily learn what it means from referring back to this section.

RPGM2 - Short for RPG Maker 2.
DBS - Short for Default Battle System.
Stat/Statistic - A numerical representation of something about a party member, enemy, class, or equipment that, in RPGs, influences some aspect of gameplay. Examples: STR (Physical Attack), DEF (Physical Defense), INT (Magical Attack), MDEF (Magical Defense), ACU (Accuracy), EVA (Evasion), and the default 'AGI' (Turn Order).
Formula/Battle Formula - A mathematical formula that produces a result based on the stats that are in the formula. Example: Physical Damage = Instigator's STR - (Target's DEF/2).
Instigator - The party member or enemy that is performing the current action (action = attack, ability, item, etc.).
Target - The party member or enemy that is being targeted by the current action.
Active Character - This is the party member or enemy that will currently be affected by any Script Commands you execute. This is the most important one to know in this whole list.

Single Targeting - When one opponent or ally is being targeted.
All Targeting - When all opponents or all allies are being targeted.
Self Targeting - When the Instigator is automatically targeted by the ability or item he is using.
Party Member/Member - A party member.
Enemy - An enemy.
Every other term used in this Guide will be explained thoroughly when it is first needed.

---

_

## V. Step #1: Necessary Planning

### A. Battle Stats, Battle Formulas, and Party Member Stats

I'm sure by now you've theorized a slew of awesome gameplay features that will set your game apart from the rest, but for now I want you to only focus on what is necessary to plan before going to work with RPGM2. Don't fret if you don't know what is necessary to plan before working with RPGM2, because I am going to tell you now.

1. Battle Stats

First, you must decide what your battle's stats and formulas will be, as well as your party members' stats. A stat (short for statistic) is a numerical representation of something about a party member, enemy, class, or equipment that, in RPGs, influences some aspect of gameplay. For example, STR is usually the stat of party members and enemies that determines how much damage a Normal or Physical Attack will do to its Target, while DEF is usually the stat of party members and enemies that determines how little damage a Normal or Physical Attack will do to its Target. You may not realize it, but this is what comprises of a battle formula: the Instigator's (person doing attack) STR and the Target's DEF are being compared somehow to produce a value for the damage the attack will inflict to the Target.

In RPGM2, there are 5 default stats for party members, enemies, classes, and equipment (discluding HP, Max HP, MP, and Max MP for now) of which you can use to make your battle formulas. Unfortunately, one of these stats (default 'AGI') also affects Turn Order (the order in which party members and enemies will take their turn in battle). The real unfortunate thing is that the Turn Order is controlled internally by RPGM2 and we have no way of altering it. The default Turn Order formula is as follows:
Whoever has the highest AGI stat will go first, then the next highest, and so on till everyone has gone that round.
All and Any Class, Equipment, and Indirect Effect (status effects) Bonuses are NOT factored - only the AGI stat designated to said member or enemy in the Party Member or Enemies Database is factored in determining turn order
If a party member and an enemy are tied in AGI, the enemy will act first.
If a party member and a party member are tied in AGI, the party member furthest to the left of the screen will act first.
If an enemy and an enemy are tied in AGI, the enemy furthest to the left of the screen will act first.

This, unfortunately, narrows us down to 4 default stats to work with. To explain, balance between the stats and how effective they are in battle will in turn determine how balanced your party members, enemies, items, equipment, abilities, and classes can be. The more balanced a game, the more likely the player will enjoy it. With the AGI stat already fully determining turn order, it should not be used to do anything else - turn order is already VERY important as is.

To explain how Balance = Fun, the basic principle to making a game fun is allowing the player to do lots of different things to best the opponent. Think of a first-person-shooter (Golden Eye, Halo, etc.) - if you couldn't move forward/backward, strafe side-to-side, turn horizontally, aim vertically/horizontally, etc.; the game would be very boring. All you would do is press the fire button, and whoever pressed it faster would win. Pretty lame, huh? You see, what makes it fun is being able to do all sorts of different things to win. Well, just like in a first-person-shooter, the more things the player can do to win in an RPG, the more fun the game will likely be. This is why it's very important to have lots of different types of party members, equipment, items, abilities, etc. - which I will walk you through making those in this Guide.

The other thing to beware of is if one thing the player can do completely over-rides everything else the player can do, there by eliminating his other methods to win. I'm sure you've played some game sometime when you thought something was 'cheap' or 'unfair.' Well, that's because it was unbalanced with the other things you or your opponent could do; it over-rode you and your opponents other methods of winning. And, just like in that game, you need your game to be well-balanced for it to be fun. This is why your stats must be balanced - so that your party members, classes, abilities, equipment, etc. (which are all dependent on your stats) can be more balanced.

As far as I know, the only good ways I've thought of using the default stats are as follows:
Physical Attack, Physical Defense, Magical Attack, and Magical Defense (no Accuracy/Evasion Formula)
Physical Attack, Magical Attack, Accuracy, Evasion (no variance in Physical and Magical Defenses among members and enemies)
Physical Attack/Magical Attack, Physical Defense/Magical Defense, Accuracy, Evasion (fighters can only attack physically, mages can only attack magically, fighters only gain defense to physical attacks, and mages only gain defense to magical attacks)

This is why I recommend you make your own stats to work with: so that you are less limited in what stats you give to the party members, enemies, equipment, etc. Don't worry at all, for I will thoroughly walk you through making your own stats. The core stats that I recommend you use are:
Physical Attack, Physical Defense, Magical Attack, Magical Defense, Accuracy, Evasion, and the Default AGI (Turn Order)
You may also add your own stats, such as LUCK for determining Critical Chance, but keep in mind that doing this will make it MUCH harder to balance your party members, enemies, equipment, etc.

## 2. Battle Formulas

Now that you've got your stats planned, it's time to move on to your battle formulas. Basically, your battle formulas will determine what your battle stats do. For example, if you wanted STR to affect Physical Attacks, you'd need to make a battle formula that somehow produces a damage

value based upon the Instigator’s (person doing attack) STR stat. The common, basic formulas are as follows:
Physical Damage = Instigator’s STR - (Target’s DEF/2)
Magical Damage = Instigator’s INT - (Target’s MDEF/2)
Success Formula = (100 * Instigator’s ACU) / Target’s EVA
*note: The first 100 in the Success Formula can be changed. This number is the percent chance of success if the Instigator’s ACU and the Target’s EVA are equal. The Success Formula gives you a percent value, so having 100 there will make you have 100% success chance when the Instigator’s ACU and the Target’s EVA are equal, which you may or may not want.*

The important thing as far as battle formulas go is that every stat gets equal influence throughout the formulas. In the above examples, STR gets 1 point because it entirely determines Physical Attack, DEF gets 1 point because it entirely determines Physical Defense, etc. Every stat gets exactly 1 point of influence throughout the formulas (1 point = completely influencing a whole aspect of gameplay). That is why the formulas above are balanced.

Now, the next factor is how will the power of Physical and Magical Attacking Abilities affect your formulas. To explain, each Physical Attacking Ability and Magical Attacking Ability should not be identical, and the easiest way to differentiate them is to make some do more damage than others. In RPGM2, each Ability gets two stats unique to it that you may use in your formulas. The two stats are called Rate and Hit. The ways I have come up with using them are as follows:
Rate = multiplied with total damage, Hit = percent multiplied with total success chance
Rate = multiplied with Instigator’s STR or INT, Hit = percent multiplied with total success chance
Rate = base damage added on to total damage, Hit = percent multiplied with total success chance
The Rate is useful for making different Abilities do different amounts of damage, while the Hit is useful for making different Abilities have different success chances. For example, an Ability that puts an enemy to Sleep will use the Hit stat but not the Rate stat.

3. Party Member Stats

How your items, equipment, classes, etc. stats affect your gameplay is very important, but you do not necessarily have to have any of them in your game. What you must have in your game, however, are party members. It is important to plan your party members’ stats before working with RPGM2 because we will need a few party members with varying stats to test our Abilities with. And, now that you have your stats and formulas planned, you know exactly what stats will affect what in battle, so you can better make your party members’ stats represent their fighting style, personality, etc.

## B. Group Targeting, or No Group Targeting

Single Targeting targets one opponent or one ally. All Targeting targets all opponents or all allies. Self Targeting targets the Instigator himself (person doing Ability). Group targeting, however, is more complicated.

To explain what Group Targeting is: in RPGM1, an Ability that targeted a Group targeted all the enemies of the same type. So, say you were fighting two Goblins and a Skeleton, an Ability that targeted a Group could either hit both Goblins or the one Skeleton.

Unfortunately, Group Targeting is handled differently in RPGM2. To explain, when you're choosing a set of enemies to appear in battle (like two Goblins and a Skeleton), you can make the similar enemies (the two Goblins) a Group. However, each enemy in a Group cannot be targeted individually. Instead, you may only target the first enemy, and only after he dies can you target the second enemy. This is an obvious downside to using Groups. The advantage is being able to make Abilities with Group Targeting.

The other option you have is to not use Groups at all. Unfortunately, this limits you to only having four enemies (or less) in every battle. I personally recommend doing this though because, due to the sizes of enemies, you can barely ever even have more than four enemies in a battle. To explain, every enemy has a size (usually between 20 and 35), and you can only place enemies in battle until their cumulative size reaches 75. This, therefore, usually limits you to four or less enemies per battle anyways, so it really comes down to Group Targeting vs. being able to target each enemy individually. You MUST decide before making your Abilities (which we will be doing very soon in this Guide).
_______________________________________________________________________________
_

**VI. Step #2: Important Script Commands**

**A. Flags and Variables**

We've already touched upon Variables without you even knowing it. A Variable is a numerical value that can be used to remember a value, or to later determine a result. Sound familiar? Yes, the stats are the Variables we will be working with the most. For example, a party member's STR stat will remember their STR stat, and later determine the damage inflicted through the Physical Damage Formula. Variables can be used for all sorts of things ranging from remembering a Multiple Choice a player made to remembering how many battles the player's won to remembering a party member's STR stat and so on. The possibilities really are endless.

Flags are very similar, but inferior, to Variables. You see, a Variable will remember a value from -99999999 to 99999999, while a Flag will just remember On or Off. In this way, a Flag is an inferior version of a Variable that can only hold two values: On (0) or Off (1).

Now, I know there are over 350 default Flags and Variables and that you're quite baffled as to what they already do, and what you can do with them. Well, trying to explain all of them to you now wouldn't get much done, so I will gradually teach you what they do and what they can do as we come across their uses/potential uses in making our Battle System.

There are also over 600 Flags and 600 Variables that are NOT used by RPGM2 already, and you can use however you'd like. To change the names of these Flags and Variables, go to Game Settings from the Main Menu. These are what we call 'Custom Flags' and 'Custom Variables.' Also, note that many of the 'Sample Flags' are already used by RPGM2. Fortunately, Dungeon Warden's AWESOME 'RPG Maker 2 Advanced FAQ'at http://www.doanthenado.com/guides.htm covers these completely, so if you use one and mess up the game, his Guide will show you what to do to fix it. His Guide also has all the default Scripts, so if I say to use one and you've deleted it (unlikely, but bound to happen to someone), go find it at his Guide and re-make it in your game.

Lastly, to change a Flag's or Variable's initial value (the value it will hold when you begin playing your game), go to General Settings from the Main Menu.

**B. Info: Loads**

There are numerous Script Commands in RPGM2 that Load default flags' or variables' values. The various ones of these will load different information about different things. For the specific Flags and Variables they load, refer to Dungeon Warden's 'RPG Maker 2 Advanced FAQ' at http://www.doanthenado.com/guides.htm , or simply press start over the Info: Load command you wish to learn about. Some are also not called 'Info: Load', such as the Battle: Substitute Target Attribute for Variable, which will load the current 'Active Character''s stats.

So, now you should know how to Info: Load something you wish to know a value of, such as an Item or Party Member, but there's still one lesson left. To explain, every Info: Load command will load different info depending on what certain default Variables and/or Flags are set to. For example, Var86 [Member Number] is the database number of the current member, and depending on the value of this, the Member Info: Load command will load different info. This gives us even greater customizability, because we can set Var86 [Member Info] to anything we want, including the 'Active Character', and then use the Member Info: Load to learn info about them. To quickly see what Flags and Variables determine the outcome of each Info: Load command, and what Flags and Variables are loaded by those Info: Load commands, press start on them.

You'll notice that any Info: Load commands that load a member's stats are dependent on flag??[Stat Check Type]. When it is On, the game will load the member's stats with equipment and class factored in (so, say he has a weapon that increases STR by 20, it will then load his full STR instead of his STR minus the weapon's STR). As you can probably tell, you MUST have this flag On at all times, or at least should (otherwise, no equipment or classes). To change this Flag's initial value, go change it under General Settings from the Main Menu. It is also a good idea to place a Data: Flag: flag??[Stat Check Type] On before any Info: Load that may involve a member's stats (to be safe).

**C. Info: Saves**

This is used much less than the Info: Loads, but is still important to understand nonetheless. Basically, in order for the game to remember the new value you have designated for a Variable, you must save it. Just like saving a video game file will make the system remember the progress you've made, saving a Variable's value will make RPGM2 remember the change you've made to it. The most common use of this we will be covering is saving a member/enemy's most current HP value, like after being attacked or healed. And, just like the Info: Loads, visit Dungeon Warden's 'Advanced RPG Maker 2' Guide or press start on the Info: Saves to learn which Flags and Variables they will save the contents of.

**D. Altering Flags and Variables**

Well, I already mentioned naming the Custom Flags and Variables under Game Settings from the Main Menu, but that's not what I will be covering here.

So, you already know how to load and save information, now there's one step missing. What good would loading something and saving it do if we didn't alter its contents in between the loading and the saving? Nothing. Thus, it is time to learn how to alter a Flag's or a Variable's contents.

First, Flags are very simple. You have three choices of how to alter a Flag: you can turn it On, turn it Off, or Reverse it. The first two are self-explanatory, and the third option will change it to On if it's Off and change it to Off if it's On. The Script Command to do this is Data: Flag.

Variables are a little more tricky, however. With Variables, you can execute mathematical equations as well as simply over-writing the Variable's value. The Script Command to do these is Data: Variable. The various options are:
(n is the Variable in which we are altering)
Change a Variable's value to a set number (n = 100 + 0)
Add a set number to the Variable's value (n = n + 10)
Subtract a set number from the Variable's value (n = n - 5)
Multiply the Variable's value by a set number (n = n * 2)
Divide the Variable's value by a set number (n = n / 4)
Randomize a Variable's value between two set numbers (n = 1 ? 100) (this will give the Variable a random value between 1 and 100 -> a percent)
*Note that the 'set numbers' can also be different Variables, such as (we touched upon it earlier) a party member's STR being multiplied by an Ability's Rate to determine damage.*

These are the commands from which we will make your battle formulas with. These commands can be combined to create bigger, more complex calculations, just like the math commands you've learned in school can.

It's also important to note that RPGM2 will cut off any remainders (in other words, round down to the nearest whole number). This is important to keep in mind when doing calculations, such as the Success Formula: the (Inst ACU/Targ EVA) would most likely result in a number between 0 and 1, and RPGM2 would round it to 0. This is why we multiply the Inst ACU by 100 before dividing it by the Targ EVA.

**E. Inputs**

Inputs are real easy, especially if you understood the above sections about Flags and Variables. Inputs are the exact same as Flags and Variables, but instead of remembering On/Off or a numerical value, they remember a series' of letters (like a word or phrase). For example, 'Doan', 'Domain', and 'Rocks' are all series' of letters that an Input can remember.

And, just like Flags and Variables, there are default Inputs that can have their contents loaded and saved through Info: Load and Info: Save commands. This is how the game loads the party leader's name and displays it in a message window, and how the player can input a name for the main character and RPGM2 will remember it (save the input by the player). Pretty simple, right?

And, just like the Flags and Variables, go to Game Settings to change the Inputs' names and General settings to change their initial contents. There are also about 75 Custom Inputs for you to use in your game.

**F. Conditions**

Well, we've covered how to 1) load a Flag/Variable/Input's contents, 2) alter said Flag/Variable/Input's contents, and 3) save a Flag/Variable/Input's contents, which is all fine and dandy, but there's still one lesson left: Conditions.

You see, after loading a Flag/Variable/Input's contents (and if you wish altering it), you can use various Script Commands to check if it meets certain conditions. For example, when trying to buy an item, RPGM2 will check to see if your money (a numerical value: Variable) is equal to or greater (Condition: is it equal to or greater than?) than the cost of the item. Well, just like RPGM2 does that, you can check to see if a Flag/Variable/Input's contents meet a certain requirement, and have a result for when it does meet the requirement (buying the item) and a result for when it does NOT meet the requirement (failing to buy the item).

There are three important types of Condition Script Commands to do this with. The first, and most simple, is the Script Branch: Condition, which will simply check a Flag/Variable/Input's contents and do something if it meets a certain condition:

```
Script Branch: Condition: Var86 [Member Number] = 5
Text: Display Message Window: 'Yay!'
Condition End
```

In the above script, the first line will check to see if Var86 [Member Number] = 5, and, if it is, the game will go ahead and execute the Script Commands in between it and the Condition End. So, if [Member Number] = 5, a message window saying 'Yay!' will be displayed (requirement is met), and if [Member Number] =/= 5, nothing will happen (requirement is not met). The Condition End will automatically appear when you make a Script Branch: Condition.
*Note that the ? is supposed to be a >. This is just an error in RPGM2.*

The second type is a Script Branch: Sort, which can do the exact same thing as the Script Branch: Condition, but can also do more. Basically, with a Script Branch: Sort, you lose out on being able to use greater than or less then conditions, but you can make conditions for more than one value, such as If Var86 [Member Number] = 5 display message 'Yay!' AND If Var86 [Member Number] = 9 display message 'I did it!'.

```
Script Branch: Sort: Var86 [Member Number]
Apply If: 5
Text: Display Message Window: 'Yay!'
To End
Apply If: 9
Text: Display Message Window: 'I did it!'
To End
Branch End
```

In the above script, the first line will set-up the Sort. The Apply Ifs and To Ends are added separately, but can only be used after a Script Branch: Sort command. The Apply If: 5, Text: Display Message Window: 'Yay!', and To End do the exact same thing as a Script Branch: Condition, and each of those three lines does the same thing as each of the three lines in the

Script Branch: Condition script above. Also, the Branch End will appear automatically when you make a Script Branch: Sort.

The last type of Condition Script Commands is the Script Branch: Repeat. The Script Branch: Repeat is the exact same as a Script Branch: Condition, but it will repeatedly do the commands in between the Script Branch: Repeat and Branch End until the condition is no longer met.

```
Script Branch: Repeat: Var86 [Member Number] = 5
Text: Display Message Window: 'Yay!'
Data: Variable [Member Number] = 7
Branch End
```

In the above example, the Script Branch: Repeat will act the exact same way as the earlier example for Script Branch: Condition. It will check to see if [Member Number] is equal to 5, and if it is execute the commands in between the Script Branch: Repeat and the Branch End. The Data: Variable [Member Number] = 7 is to make it so that the loop (loop = Script Branch: Repeat) does not continue for ever and never stop. A good practice exercise for you now would be to try to make a Script Branch: Repeat that does something 5 times, and then stops. In order for it to stop, you must somehow make the requirement (Var86 [Member Number] = 7 in our example) no longer be met after the fifth repetition.

### G. Call: Scripts

Call: Scripts are very simple: they allow you to insert a Script into another Script. Use these to keep your Scripts from getting messy. Basically, A Call: Script command will call another Script and execute its contents. These are EXTREMELY helpful because a) you can reuse scripts over and over in multiple places and b) using these will keep your scripts easy to read for later editing.

### H. Battle: Make Active Character

This command is very simple. Through RPG Maker 2's internal programming, the Instigator begins as the Active Character. This will switch the Active Character (Active Character = party member or enemy that will currently be affected by any Script Commands you execute) between the Instigator and the Target. So, for example, we would use this to switch to the Target, then we'd do something to them (subtract HP, etc.), then we would use this to switch back to the Instigator. Pretty simple, right?

### I. Abbreviations

I will not always write out the full script command such as Script Branch: Sort, and will often use abbreviations such as 'Sort'. They should all be pretty easy to figure out, but nonetheless, if you have a question about one, feel free to ask at **Doan's Domain.** These particularly, the abbreviations, are something that every active member of **Doan's Domain** will be able to help you with.

___________________________________________________________________________
_

## VII. Step #3: Building the System

### A. Scripts "HP-" and "HP+"

```
Script "HP-"
-----------------------
Text: Clear Message
[call a default script called "Instigator Name"]
Data: Load Direct Effect Info
Data: Substitute Attribute for Variable
var97 'Hit Points' = var97 'Hit Points' - var35 'Damage Total'
display a message for damage taken using var35 'Damage Total'
Data: Substitute Variable for Attribute
```

The “Instigator Name” Script displays the Active Character’s name. Data: Substitute Attribute for Variable loads the Active Character’s stats, then we alter the Active Character’s current HP, then the Data: Substitute Variable for Attribute will save the new value of var97 ‘Hit Points’. I’m sure you can imagine how much this Script will be used.

```
Script "HP+"
-----------------------
Text: Clear Message
[call a default script called "Instigator Name"]
Data: Load Direct Effect Info
Data: Substitute Attribute for Variablevar97 'Hit Points' = var97 'Hit Points' + var35 'Damage
Total'
display a message for damage taken using var35 'Damage Total'
Data: Substitute Variable for Attribute
```

This Script is the exact same as the ‘HP-‘, but instead of subtracting from the Active Character’s HP it adds to it (notice the + sign in the Script?). This is used for healing a member or enemy.

**B. Physical Damage Formula**

By now you should know what you want your Physical Damage Formula to be - if you don’t, re-read through Step #1: Necessary Planning.

To make this easier, go under Game Settings > Menu Text and change ATK, GRD, INT, and LUCK to STR, DEF, INT, and MDEF (or whatever you want to call them).

Now, it’s entirely up to you how to make your game. Do you want it to be like old RPGs where ‘fighters’ only do Normal Attacks and ‘mages’ only use Magic? Or, do you want it to be different? Neither is undoubtably better, but I recommend NOT doing it the old-fashioned way because (yes, I know you know) Balance = Fun, and ‘fighters’ having infinite equal-power attacks and ‘mages’ having limited equal-power attacks will definitely make balancing the members and classes much more difficult. This is why I recommend you allow your ‘fighters’ (and some ‘mages’?) to have Physical Attack Abilities - Abilities that require MP (MP can be re-named under Game Settings, as can all the default stats) to be performed and do greater damage than a Normal Attack. If you decide to do it the old-fashioned way, disregard the Rate and Hit parts of this formula. There are of course other ways to balance 'fighters' and 'mages', but these are the two I will cover.

This is the basic formula used for calculating Physical Damage (Physical Damage = Instigator's STR - (Target's DEF/2)) that you will build upon:

* = A script I supply.

** = This is where you can customize the formulas. I will also supply the Scripts that would go here for the 'basic formula' mentioned above, but this is where you get to customize stuff!

Script "Phys Main"
--------------------------
Battle: Make Active Character
**[call a script to load target's DEF]
Battle: Make Active Character

**[call a script to load inst's STR]
**[call a script that sets var35 'Damage Total' to something resultant of comparing the Instigator's STR and the Target's DEF]**modify damage however (for example: Critical Chance, Elemental Boosts/Resistances)
Battle: Make Active Character

*[call HP-]

Battle: Make Active Character

The first Battle: Make Active Character switches to the Target, then your Script loads his DEF. The second Battle: Make Active Character switches back to the Instigator, then your Script loads his STR. You then Call the Script that produces the Damage that will be inflicted, then after you know the Damage it will inflict, you can edit it however you want. I already supplied the 'HP -' Script for you.

'Load Target DEF'
-------------------------
Data: Substitute Attribute for Variable (load Active Character's stats)
tempvariable0 = var100 'Defend' + 0 (load Active Character's DEF into a tempvariable)
tempvariable0 = tempvariable0 / 2 (divide Active Character's DEF by 2)

'Load Instigator STR'
-------------------------
Data: Substitute Attribute for Variable (load Active Character's stats)
tempvariable1 = var99 'Strength' + 0 (load Active Character's STR into a tempvariable)

'Phys Damage Total'
-------------------------
Var35 [Damage Total] = tempvariable1 - tempvariable0 (configure 'Damage Total' based on the Instigator's STR and the Target's DEF)

'Critical Chance' (Optional)
-------------------------

tempvariable2 = 1 ? 100 (randomize a tempvariable between 1 and 100 -> percent)
Condition: tempvariable2 =< 5 (if variable is under 5, do the following -> 5% chance of a Critical)
Var35 [Damage Total] = [Damage Total] * 140 (these two lines are making it do an extra 40% of damage)
Var35 [Damage Total] = [Damage Total] / 100
Condition End

*Note: The 5 in the second line can change - it is the percent chance of a Critical occuring. The 140 and 100 in lines 3 and 4 can change too - this is the percent of the normal attack it will do (140/100 = 140%).*

*Note: This Script will work perfectly with your Magic Damage Formula later.*

‘Elemental Boosts/Resistances’ (Optional)
-------------------------Data: Load Direct Effect Info

Condition: var163 [Direct Effect] = (number of your Normal Attack Direct Effect from VI. Step #3: Building the System E. Making Your Normal Attack)
Tempvariable3 = 100 - var29 'Target Resist W'
Condition End
Condition: var163 [Direct Effect] =/= (number of your Normal Attack Direct Effect from VI. Step #3: Building the System E. Making Your Normal Attack)
Tempvariable3 = 100 - var30 'Target Resist M'
Condition End
Var35 'Damage Total' = 'Damage Total' * Tempvariable3
Var35 'Damage Total' = 'Damage Total' / 100

For sorting out whether to use var29 'Target Resist W' (Weapon Resist) or var 30 'Target Resist M' (Magic Resist), I Load: Direct Effect Info and sort the 'Direct Effect' variable to find whether the action ensuing was a 'Normal Attack' Direct Effect (Weapon Resist) or -not- a 'Normal Attack' Direct Effect (Magic Resist). What others do here can vary, and your values for the 'Direct Effect' variable will vary from mine, but at first thought this was my solution, and I have tested it and know that it works perfectly. Don’t forget to come back and put the database number of your Normal Attack Direct Effect in this script later.

*Note: The last line's '100' can be altered to allow for greater variances in Property-Damage-increasing. For example, changing this 100 to 50 would make a value of -100 in the Property Resist section of a Party Member or Enemy increase damage to 400% instead of 200%.*

*Note: This Script will work perfectly with you Magic Damage Formula later.*

Lastly, to load the Direct Effect’s Rate and Hit variables, use a Data: Load Direct Effect Info, and edit tempvariable1 or var35 [Damage Total] however you like with the Rate variable. The Hit variable will be used later in the Success Chance Formula. A percent is done just like lines 3 and 4 in the Critical Chance Script above.

*Note: The Hit Direct Effect variable is actually var165 ‘Direct: Success#’. In the Direct Effect database it is called Hit, though.*

**C. Magic Damage Formula**

By now you should know what you want your Magic Damage Formula to be - if you don't, re-read through Step #1: Necessary Planning.

The Magic Formula is JUST LIKE the Physical Damage Formula, but with INT instead of STR and MDEF instead of DEF. Copy your 'Phys Main', 'Load Target DEF', 'Load Instigator STR', and 'Phys Damage Total' Scripts, and Paste duplicates of them. Go through the duplicates, and EVERYWHERE it says STR or Strength, change it to INT or Intelligence, and EVERYWHERE it says DEF or Defense, change it to MDEF or Luck (The default 'Luck' is our Magic Defense stat). The 'Critical Chance' and 'Elemental Boosts/Resistances' Scripts will be the EXACT same as before, so don't do anything to those. You can either choose for Magic Attacks to hit 100% of the time with no Critical Chance, or for them to NOT hit 100% of the time but WITH a Critical Chance. The first choice is done by not putting a Success Chance Formula in the Magic Attack Abilities' Direct Effect's 'Success' slot, which we will get to later, and the second choice is done by making your Magic Attack Abilities' Direct Effects exactly like your Physical Attack Abilities' Direct Effects, but with the Magic Damage Formula instead of the Physical Damage Formula.

Lastly, to load the Direct Effect's Rate and Hit variables, use a Data: Load Direct Effect Info, and edit tempvariable1 or var35 [Damage Total] however you like. A percent is done just like lines 3 and 4 in the Critical Chance Script above.

**D. Magic Heal Formula**

Now, I know I haven't covered the Magic Heal Formula yet. The reason why is simple: it ruins my theory about each stat getting 1 point of influence throughout the formulas (that I said before) making the stats and formulas balanced. Well, unless you plan on making a new stat specifically for the Magic Heal Formula, plan on not having any Magic Attack Abilities, or plan on not having any Magic Heal Abilities; you're stuck using INT for both.

The trick to balancing the game then, is to not allow any party member to have both Magic Attack Abilities and Magic Heal Abilities at the same time without a significant INT hindrance. This is already done in most games any ways. The follow-up trick to NOT making the Magic Healer boring and ineffective at anything but healing is to give him/her (and other members) Status Effect Abilities, and other 'Assist' (Assist = not damaging or healing anyone) moves. This way, when he/she's not healing the other party members, he/she can at least contribute to defeating the enemy - but without said party member becoming better than fighters (Physical Attacks/Assist Moves) or offensive mages (Magical Attacks/Assist Moves).

Anyways, I'm done with my long rant. Here's the Magic Heal Formula Script:

```
Script "Heal Main"
--------------------------
**[call a script to load inst's INT]
**[call a script that sets var35 'Damage Total' to something resultant of the Instigator's INT]
**modify damage however (for example: Critical Chance, Elemental Boosts/Resistances)
```

Battle: Make Active Character
*[call HP+]
Battle: Make Active Character
First, the Script loads the Instigator's INT. You then Call the Script that produces the HP value (stored in var35 'Total Damage') that will be healed, then after you know how much it will heal, you can edit it however you want (such as factoring in Rate so that the amount healed is multiplied by a percent value). I already supplied the 'HP +' Script for you above.

'Load Instigator INT'
-------------------------Data: Substitute Attribute for Variable (load Active Character's stats)
tempvariable1 = var101 'Intelligence' + 0 (load Active Character's INT into a tempvariable)

'Heal Damage Total'
-------------------------
Var35 [Damage Total] = tempvariable1 + 0 (configure 'Damage Total' based on the Instigator's INT)

*Note: Also, the Critical Chance and (if you want to?) Elemental Boosts/Resistances Scripts will work with your Magic Heal Formula too.*

**E. Making Your Basic Normal Attack**

Now, we've made all your formulas except for your Success Chance one (we will get to it soon, it's more complicated to apply into RPGM2), but we now need to insert them into actual Abilities to use in RPGM2.

Abilities, in RPGM2, are much like Events - they are only the capsules which hold the Scripts and other things (character model, object model, etc.) together. The Abilities hold some basic information about themselves (MP Cost, Name, Elemental Property), but the Direct Effects of Abilities is where all the technical stuff happens (like Scripts compared to Events). Each Ability can have a Direct Effect attached to it, and the Direct Effect will tell the system what to do (such as configure var35 'Total Damage' from stats and subtract/add it to the Target's HP) when the Ability they're of is used.

Now, I know what you're thinking: A Normal Attack is an Ability - but it's not selected from the 'Magic' or 'Skills' like in fu-ma? Well, you're right in that a Normal Attack is not an Ability in RPGM2. It is, however, controlled by a Direct Effect (like Abilities are) and works just like one.

Now, in order to make this the Normal Attack for your party members, you must put this Direct Effect in all your Party Members' 'Attack Direct Effect' slots, and to make this the Normal Attack for your enemies, you must put this Direct Effect in all your Enemies' 'Attack Direct Effect' slots. For now, just work with two party members and two enemies (not in the same Group).

Now, for making this Direct Effect, and FOR MAKING ALL YOUR DIRECT EFFECTS, copy another Direct Effect and edit it to your liking. For now, copy the default 'Normal Attack' Direct Effect to work with. Basically, copying a default one to edit will make all the parts of it we don't edit work properly.

Also, ALL your Events containing the Scripts we've made that are to be used here MUST be set to 'System' in the Event Database.

Next, simply place your 'Phys Main' Script/Event in the Normal Attack Direct Effect's 'Success' slot. To explain, the 'Success' slot of a Direct Effect is what happens when the Direct Effect is successful, which in this case is damaging the Target physically.

Go back to your Elemental Boosts/Resistances Script and put this Direct Effect's database number in it.

*note: Now, you may have noticed how the enemies are only attacking the party leader. This is because your Normal Attack Direct Effect is set to attack 'Single Party Member - Single Enemy' under Targeting in the Direct Effect database. Well, simply change that to 'Single Random Member - Single Random Enemy'. As you see, you'll need one set to 'Single Party Member - Single Enemy' for the party members and one set to 'Single Random Member - Single Random Enemy' for the enemies. Simply put, the Abilities' Direct Effects control who the Ability will target. This is also how you change an Ability to do All or Group Targeting (we will get to Self Targeting later). Note that when it says 'Unit' it is referring to a 'Group'.*

## F. Making Your Basic Physical Attack Ability, Making Your Basic Magical Attack Ability, and Making Your Basic Magical Heal Ability

This should be easy if you understood the last part. This time, instead of making a Direct Effect and attaching it to the party members' and enemies' Normal Attacks, we will be making a Direct Effect and attaching it to your first Physical Attack Ability, then to your first Magical Attack Ability.

To explain, the Physical Attack and Magical Attack Abilities and Direct Effects are the EXACT same except for where you will place either your 'Phys Main' or 'Magic Main' Script/Event. Because there are no default Physical Attack Abilities (I think, it doesn't matter anyways), I want you to copy and paste a copy of the default #0 Direct Effect called 'Fire' (a basic Magic Attack spell). After that, place your 'Phys Main' Script/Event in the 'Success' slot of the copy you made (sound familiar?).

Now, go to Abilities, make a new Ability, and place this Direct Effect in the Ability's Direct Effect slot. Edit the Ability options however you like; they are all very straightforward and simple, such as MP Cost, Name, Elemental Property, etc. This is now your Basic Physical Attack Ability that, later, you will want to copy and paste to make other Physical Attack Abilities out of. You will also copy the Direct Effect we made to make Direct Effects for your other Physical Attack Abilities later as well.

Like I said before, the only difference between your Physical Attack and Magical Attack Abilities and Direct Effects are where you will place either your 'Phys Main' or 'Magic Main' Script/Event. So, as you've probably guessed, copy and paste duplicates of both your Basic Physical Attack's Ability and Direct Effect. Place your 'Magic Main' Script/Event in the 'Success' slot of the Direct Effect, and place your new Direct Effect in the Direct Effect slot of your new Ability. Pretty simple, huh? This is all it will take you to make new Abilities of the same type (type = Physical Attack, Magical Attack) later.

And, you guessed it, do the same thing you did in the paragraph before, except this time copy and paste the default #58 ‘Heal’ Direct Effect to work from. As you can probably see, there’s so little variation in the actual ‘Ability’ part that it doesn’t really matter which you copy and paste. And again, put your ‘Heal Main’ Script/Event in the Direct Effect’s ‘Success’ slot.

## G. Making Your Healing Item Formula, and Making Your Basic Healing Item

This is done the EXACT same way your Abilities were made, only instead of placing the Direct Effect in the Ability’s Direct Effect slot, you will be placing it in the Item’s Direct Effect Slot. For healing items, the only formula I’ve seen before is a percent of the target’s Max HP, which I will post a Script for, and the basic flat value HP Heal.

This is for the percent of max HP value heal:

```
Script "Item Heal Main"
--------------------------
Battle: Make Active Character
**[call a script to load target's Max HP]
**[call a script that sets var35 'Damage Total' to something resultant of the Target’s Max HP]
**modify damage however (for example: Critical Chance)
*[call HP+]
Battle: Make Active Character
```

First, the Script switches the Active Character to the Target, then load’s the Target’s Max HP. The Max HP is then used with the Item’s Direct Effect’s Rate variable to produce a value for var35 ‘Total Damage’. I already supplied the ‘HP +‘ Script for you.

```
‘Load Target Max HP’
-------------------------
Data: Substitute Attribute for Variable (load Active Character’s stats)
tempvariable4 = var104 'Max Hit Points' + 0 (load Active Character’s Max HP into a tempvariable)
```

```
‘Heal Damage Total’
-------------------------
Load Direct Effect Info (Load Direct Effect’s Rate)
tempvariable5 = var164[Direct: Rate] * 100 (these four do our calculations)
tempvariable5 =  tempvariable5 / tempvariable4
Var35 [Damage Total] = tempvariable5 + 0
```

And here is the flat HP Value Heal Script:

```
Script "Item Heal Main"
--------------------------
Battle: Make Active Character
**[call a script that sets var35 'Damage Total' to the Direct Effect’s Rate]
**modify damage however (for example: Critical Chance)
*[call HP+]
Battle: Make Active Character
```

First, the Script switches the Active Character to the Target, then sets var35 ‘Damage Total’ to the Direct Effect’s Rate, then heals the Target by that much. I already supplied the ‘HP +‘ Script for you.

```
‘Heal Damage Total’
-------------------------
Load Direct Effect Info (Load Direct Effect’s Rate)
tempvariable5 = var164[Direct: Rate] + 0 (these two do our calculations)
Var35 [Damage Total] = tempvariable5 + 0
```

*Note: The Elemental Boosts/Resistances Script will NOT work with these formula. This is because the Element is only chosen for either a Weapon (under Item) or an Ability (under Ability). RPGM2 will only let you choose an Element for the item if it is a Weapon to be equipped, then attacked with through a Normal Attack.*

**H. Success Chance Formula (optional)**

You know what the ‘Success’ slot of a Direct Effect is used for, right? It’s what will happen when the Direct Effect is successful. Well, there are numerous other slots in Direct Effects that you can customize for other things. For what each of them is used for, and can be used for, check out Dungeon Warden’s ‘Advanced RPGM2 FAQ’ at http://www.doanthenado.com/guides.htm . Pressing start would normally work too, but there are a few slots where the in-game descriptions of them are off.

The possibilities of what you can do with these slots are near-endless, but I’m going to focus on just one slot for now: the ‘Success Check’ slot. The Script/Event that is placed here will determine whether the Direct Effect is successful or not based on whether Flag94 [Inst Success] is On or Off at the end of the Script. Yeah, you guessed it, this is where we can add our Success Chance Formula.

Before I have you do any work for this, I must tell you that this is optional. You see, every Direct Effect has a Rate and Hit variable designated to it. Defaultly, Success Chance is determined by the Direct Effect’s Hit variable, where the value of it is a percent chance of success. Feel free to skip the rest of this part and move on if that’s how you want it to be.

Alright, for this, we will temporarily use the STR/Strength stat for ACU/Accuracy and the DEF/Defense stat for EVA/Evasion. You see, in order for us to have all 7 of the basic stats (Physical Attack, Physical Defense, Magical Attack, Magical Defense, Accuracy, Evasion, and the Default AGI (Turn Order)), we will need to make our own custom stats for our party members, but we can’t quite do that yet, so we’ll temporarily use the default STR and DEF stats as our ACU and EVA stats, then later come back and change them to our custom ACU and EVA stats.

```
Script "Success Chance Main"
--------------------------
Battle: Make Active Character
**[call a script to load target's EVA]
Battle: Make Active Character
```

**[call a script to load inst's ACU]
**[call a script that sets Custom Variable [Success Chance] to something resultant of comparing the Instigator's ACU and the Target's EVA]
**modify [Success Chance] however (example: factor Direct Effect's Hit)
tempvariable2 = 1 ? 100 (set-up percent chance)
Condition: tempvariable2 =< [Success Chance] + 0
Data: Flag 94 [Inst Success] On
Condition End
Condition: tempvariable2 > [Success Chance] + 0
Data: Flag 94 [Inst Success] Off
Condition End
The first Battle: Make Active Character switches to the Target, then your Script loads his EVA (DEF). The second Battle: Make Active Character switches back to the Instigator, then your Script loads his ACU (STR). You then Call the Script that produces the Success Chance that will be used, then after you know the Success Chance (a percent), you randomize a tempvariable from 1 to 100 and check to see if the randomized tempvariable is less than or equal to [Success Chance]. Lastly, if the Condition is met, default Flag 94 [Inst Success] is turned On, and if it isn't, the flag is turned Off.

'Load Target EVA'
-------------------------
Data: Substitute Attribute for Variable (load Active Character's stats)
tempvariable0 = var100 'Defense' + 0 (load Active Character's EVA into a tempvariable)

'Load Instigator ACU'
-------------------------
Data: Substitute Attribute for Variable (load Active Character's stats)
tempvariable1 = var99 'Strength' + 0 (load Active Character's ACU into a tempvariable)

'Compare ACU/EVA'
-------------------------
Custom Var [Success Chance] = 0 + 0 (reset [Success Chance] variable to 0)
Custom Var [Success Chance] = tempvariable1 * 100 (note: this 100 is the 100 that [Success Chance] will equal if the Inst ACU and Targ EVA are the same)
Custom Var [Success Chance] = [Success Chance] / tempvariable0 (note: we did the * 100 first because RPGM2 cuts off fractions (rounds down to the nearest whole number))

'Factor in Direct Effect Hit'
-------------------------
Data: Load Direct Effect Info
tempvariable3 = var?? [Direct: Hit] + 0 (load Direct Effect's Hit variable into a tempvariable)
Custom Var [Success Chance] = [Success Chance] * tempvariable3
Custom Var [Success Chance] = [Success Chance] / 100 (Hit = a percent modifier, for example: 200 Hit makes it multiply the Success Chance by 2 (and 2 = 200%))

## I. Making the default stats editable by status effects, Making your own custom stats (optional), and Making your party members' stats

I'm sure you've played some rpg with a temporary status effect that increased or decreased a stat of the afflicted party member or enemy. Well, there's something you must do if you want those to be possible in your rpg.

To explain, if a party member's STR is 100, and he gets afflicted with a STR+40% status effect he'll have 140 STR (40% of 100 = 40). Later, when it calculates how much to reduce his STR (when the status effect is wearing off), he'll go from 140 STR to 84 STR (40% of 140 = 56). Obviously, this is bad. You don't want temporary status effects permanently changing party members' stats.

First, there is one very easy way to avoid this: have your STR Up status effect add a flat value instead of a percent value. The other way, which I will show and teach you now, is to load the party members' stats into tempvariables that will be used for that one particular battle. For example, we would take a party member's STR at the beginning of battle and load it into a tempvariable. Then, when we alter the tempvariable, we won't permanently change the party member's STR stat.

Conveniently, every party member and enemy gets 10 customizable variables that can be used in battle, and we will use those as our tempvariables. They are called a party member's or enemy's 'Battle Variables.' The only downside is that we can't rename them, so you will HAVE TO REMEMBER (write them down and keep the list with you) which numbered 'Battle Variables' are which stats. I recommend you organize them in a logical order such as: Physical Attack, Physical Defense, Magical Attack, Magical Defense, Accuracy, and Evasion.

*note: Battle Variable 0 and Normal Variable 0 do not work properly with Party Member #0, thus I recommend you use Battle Variables 1 through 6.*

Now, I know what you're thinking. If we were to make our STR UP status effect in that method, we'd still be permanently changing the party member or enemy's STR Battle Variable. Well, you're right; it's not perfect yet. I will show and teach you how to fix that when we get to making our Indirect Effects (status effects). For now, let's concentrate on using the Battle Variables instead of STR, INT, etc.

Our first step is to load each party member's and enemy's STR, INT, DEF, and M.DEF stats into their Battle Variables. This is really quite simple. All the default party members have a default script called 'Member Start' run at the beginning of every battle. The default script called 'Enemy Appear' is the same, but for enemies. Go to the 'Enemy Appear' and add a Call: Script [Member Start] command in it to make your 'Member Start' script affect enemies too. Just like with your Abilities and Direct Effects, you MUST copy and paste default party members and enemies to work with (for the same reason, so that what we don't mess with will work properly). When the default 'Member Start' script runs, it has already selected said member or enemy as the Active Character, so any commands in the 'Member Start' script will run once for every member and enemy with said member or enemy as the Active Character. So, put these script commands in it (the 'Member Start' script) to load the party members' stats into their respective Battle Variables:
Battle: Substitute Target Attribute for Variable (load all their stats, including STR, INT, DEF, and M.DEF)
Data: default var 'BattleVariable1' = var99'Strength' + 0

Data: default var 'BattleVariable2' = var100'Defend' + 0
Data: default var 'BattleVariable3' = var101'Intelligence' + 0
Data: default var 'BattleVariable4' = var103'Luck' + 0 (load STR, INT, DEF, and M.DEF into BattleVariables1 through 4)
Battle: Substitute Variable for Target Attribute (save all their stats, including the Battle Variables)

As you may have guessed, the next step is to go through all your Formula Scripts and change 'Strength' to 'BattleVariable1', 'Defense' to 'BattleVariable2', 'Intelligence' to 'BattleVariable3', and 'Luck' to 'Battle Variable4'. This is so that your Formulas are now affected by their Battle Variables instead of their permanent stats.

Now, making your own custom stats is similar to what we just did. We made it so that the Battle Variables are used in battle and that the default permanent stats of STR, INT, DEF, and M.DEF are only used outside of battle. The next step in making our own custom stats is to make the default permanent stats of STR, INT, DEF, and M.DEF NOT do anything outside of battle, and to make Custom Variables for every party member's stats for out of battle, then set the respective members' Battle Variables equal to their custom stats (called custom stats because they are stats made of Custom Variables) in the default 'Member Start' script.

Now, if you chose not to make custom stats, just set the party members' stats in the Party Member database.

If you chose to make custom stats, make Custom Variables for every stat for every member under Game Settings. Then, go set their initial values under General Settings. Set a value for the default 'AGI' (turn order) in the Party Member database. What values STR, INT, DEF, and M.DEF have no longer matter. To not confuse the player, I recommend putting 0 for the four stats we aren't using (that way the player will know those stats aren't used for anything).

You may have guessed that our next step is to set the Battle Variables of each member to the respective member's custom stats. You may also be wondering what will happen to the enemies when this script runs: they don't have any custom stats. Well, one useful default Flag is Flag72[Instigator Side]. If it is On, the Active Character is an enemy. If it is Off, the Active Character is a party member. So, all you have do is use a Script: Condition command to see whether the Flag is On or Off, then treat the Active Character as either a member (Off) or an enemy (On). Thus, your NEW script commands that will replace your old script commands done in this part earlier (in the default 'Member Start' script) should now be:
Condition: Flag72 [Instigator Side] Off
***Data: Member Info: Change to Member Order
***Data: var86[Member Number] = var48[Inst Member] + 0
*Call: 'SetCharStats'
*Call: 'SetBattleStats'
Condition End
Condition: Flag72 [Instigator Side] On
Battle: Substitute Target Attribute for Variable
**[edit the enemy's Battle Variables however you wish, or not]
Battle: Substitute Variable for Target Attribute
Condition End

The two *** commands are VERY helpful. This will give you the [Member Number] of the Active Character, if the Active Character's a member. This will come up again with tons of stuff that you may want to do later. Here, we are using it to find the [Member Number] of the Active Character so that we can use the right member's custom stats in the following two scripts. The 'SetCharStats' and 'SetBattleStats' that follow are very simple. The 'SetCharStats' sets 6 tempvariables equal to the party member's custom stats, and the 'SetBattleStats' sets 6 of that member's Battle Variables equal to those 6 tempvariables. I know it sounds redundant, but the tempvariables will allow us to edit the Battle Variables however we want, such as adding to the tempvariables based on what that party member's currently equipped with (which we will get to very shortly).

To explain the enemy part of the script before: the enemies only need Battle Variables because they will only be present in-battle. For this reason, you will simply set their Battle Variables to whatever you want in the Enemy Database. The commands from Condition: Flag72 [Instigator Side] On are thus unnecessary, because they already have their Battle Variables ready, but I've found it useful to edit the enemy's stats based on which enemy it is (Get Enemy Name, Sort: Input [Common Name]) and where the player is located (use a Custom Variable).

```
'SetCharStats'
--------------------------
Data: Member Info Load
Sort: var86 [Member Number]
*Apply If: (database number of a certain member)
**tempvariable1 = (that member's custom STR stat)
*To End
Branch End

* = repeat for every member
** = repeat for every stat, for every member
```

```
'SetBattleStats'
--------------------------
Battle: Substitute Target Attribute for Variable
*BattleVariable1 = tempvariable1 + 0
Battle: Substitute Variable for Target Attribute

* = repeat for every BattleVariable
```

Lastly, now that you've made your ACU and EVA stats, go back to your Success Chance Formula script and change your STR and DEF Battle Variables to your ACU and EVA Battle Variables.

---

_

## VIII. Ways to Improve the Gameplay While Using the DBS

### A. Varying Equipment/Abilities/Items

It may seem like I'm beating a dead horse, and I apologize for that, but I can't stress enough how important this is. The more the player can do to best the computer, the better - as long as no one thing completely overrides the other options.

**B. A Pseudo Chrono Cross/Tactics Ogre Magic System Where MP Gradually Increases as the Battle Progresses**

Basically, this will give mages an infinite supply of MP to cast spells with, but not all at once. This should help to better balance mages with fighters, and as I've already said Balance = Fun.

**C. Using Items Directly from the Bag in the DBS**
This is complicated, so don't worry if you don't understand it fully, just try to at least understand what the add-on parts (marked by * or **) do:

1. Make a standard item that is used and does something (heal HP, heal MP, etc.), or use the one you made earlier in the Guide. No specifics here matter. This is just the item we will be using for your test-run of this. You will be able to make more later.

2. Make an Ability like so:
Usable = Battle
MP Cost = 0
Direct Effect = Direct Effect from Step #3
Basically, instead of using the actual Item, we will teach the party members an Ability that mimics the Item's effect when they have that item in the bag, and remove the Ability when they no longer have that item in the bag. This is that Ability.

3. Make a Direct Effect like so:
Check Start = Script from Step #5
Effect Start = '____ Uses [Ability].'

This is the Direct Effect of the Ability, which as you learned earlier, contains all the technical stuff that happens when an Ability, Item, or Normal Attack is used. Don't forget to place this in the Ability you made in step #2.

4. Make 2 Custom Variables for every 1 Item called '____ (Bag)' and '____ (Used)'.
Unfortunately, the Bag is normally not accessible during battle. The trick here is to use a Variable to remember how many of that Item have been used in that battle, and how many of that Item were in the Bag before that battle started. We will teach the party members the Ability before the battle if they have that Item in the Bag, and set the Variable that remembers how many of that Item are in the Bag to the quantity of that Item in the Bag. The Variable that remembers how many of that Item have been used in that battle will be used to subtract the right number of that Item from the Bag after that battle.

5. Make a Script like so:
Script Control: Apply In Order
Data: Flag [TempFlagX] Off
Script: Sort Variable163 [Direct Effect]

```
*Apply If: (number of Direct Effect from Step #3)
*Script: Condition: Variable '____ (Bag)' = 0
*Data: Flag [TempFlagX] On
*Condition End
*Script: Condition: Variable '____ (Bag)' >= 1
*Data: Variable [____ (Bag)] = [____ (Bag)] - 1
*Data: Variable [____ (Used)] = [____ (Used)] + 1
*Condition End
*Script: Condition: Variable '____ (Bag)' = 0
**Member Info: [Member] Ability - (Ability from Step #2)
*Condition End
*To End
Script Branch: End
Data: Game Info: Save
Script Condition: [TempFlagX] Off
Data: Flag93 [No Instg Action] Off
Condition End
Script Condition: [TempFlagX] On
Data: Flag93 [No Instg Action] On
Condition End
* = repeat for every item
** = repeat for every member, for every item
```

This is the Script that checks to see if they have any of that Item left in the Bag when they try to use that Item's Ability. Don't forget to place this in the Direct Effect you made in step #3

6. Add this to the very top of the 'Enter Battle' Script:

```
*Data: Variable: Var134 [Item Number] = 0 + (database number of Item from Step #1)
*Data: Game Info: Load*Data: Variable: '___ (Bag)' = Var70 [Total Bag Items] + 0
*Script: Condition: Var '____ (Bag)' >= 1
*Member Info: [Member] Ability + (Ability from Step #2)
*Condition End
*Script: Condition: Variable [____ (Bag)] = 0
*Member Info: [Member] Ability - (Ability from Step #2)
*Condition End
Data: Game Info: Save
* = repeat for every item
```

This is the Script that will teach the party members the Ability before the battle if they have that Item in the Bag, and set the Variable that remembers how many of that Item are in the Bag to the quantity of that Item in the Bag.

7. Add this to the very top of the 'Battle Exit' Script:

```
*Script Branch: Repeat: Variable '____ (Used)' =/= 0
*Data: Variable: '____ (Used)' = '____ (Used)' - 1
*Data: Game Info: Load
*Party: Bag (Item from Step #1) - 1
*Branch End
*Data: Game Info: Save
```

* = repeat for every item

This is the Script that will use the Variable that remembers how many of that Item have been used in that battle to subtract the right number of that Item from the Bag after that battle.

**D. Limit Breaks with Various Initial Requirements**

Ah, you knew it was coming. Well, there are numerous ways to do this, but I'm going to just show and teach you one way to do each method.
A Limit Break is a special ability that a party member can only do every once in a while. Usually they attacked the enemy, but they also sometimes did beneficiary things to the party (they can do anything you can script), and they were always extra effective in comparison to the normal abilities.
Limit Breaks can only be used after certain conditions have been met. In Final Fantasy 7, the party members could perform their Limit Breaks after taking a certain amount of damage. In Final Fantasy 8, the party members could perform their Limit Breaks when their HP was under a certain percent of their Max HP. In Final Fantasy X, the party members obtained the ability to perform their Limit Breaks from many different requirements including taking a certain amount of damage, dealing a certain amount of damage, every turn of battle, and many more. The aforementioned requirements are the ones I will fully cover in this section, but it shouldn't be hard to expand on what this teaches you and make different requirements.

1. First Type of Requirement

First, there are two types of requirements: one to obtain the move and one to perform the move. For now, I'm going to focus on the requirements to perform the move. The various ideas I've come up with are: when being damaged, when damaging an enemy, when in critical condition, every turn of battle, or any combination of these. Really, anything is possible regarding this, but I'm just going to cover these ones.

Going back to what I taught you when we (maybe) made your own custom stats, this series of commands will tell the system which party member is the 'Active Character':

```
Condition: Flag72[Instigator Side] = Off
Data: Member Info: Change to Member Order
var86[Member Number] = var48[Insti Member] + 0
Condition End
```

After this, the database number of the current party member will be stored in Var86 [Member Number]. You'll then simply sort by that and increase that member's Limit Variable by however much you want. Then, simply copy and paste these lines to where you want them to be in your battle scripts.
*Note: The Condition: Flag72[Instigator Side] is there to make sure that an enemy is not the Active Character.*

Want the Limit Variable to increase when said party member gets attacked - add this to your HP-Script.
Want it to increase when said party member attacks an enemy - add this to your HP- Script with a Battle: Make Active Character command before and after it (to switch back to the instigator - the party member that's attacking).
Want the Limit Variable to increase when said party member's HP is below a certain value or

percent or every turn of battle - this gets a little more complex so hold on. Make a Script/Event that checks whatever the current member's HP is, and if it's under a certain value or percent, increase that member's Limit Variable. Put this Script/Event in the ---------- slot of a 'Custom' Indirect Effect. Then, go to every Party Member and put this Indirect Effect in their ---------- slot. After that, make a simple script like so:
r-------------------------------r
This will temporarily teach the said party member their Limit Break Ability.

Now, you'll want to remove that member's Limit Break Ability when it is performed. To do that, make a Script/Event like so:
r-------------------------------r
And put it in the --------- slot of the Limit Break Ability's Direct Effect.

2. Second Type of Requirement

Well, the Limit Break Flags were introduced in the Scripts of the other Limit Break part but never explained. Basically, the Limit Break Flags tell the system which party members have unlocked which Limit Breaks, and the system will let them use various ones depending on which conditions they meet. All you have to do to allow a party member to use another Limit Break Ability is turn on the Limit Break Flag that corresponds to that Limit Break Ability. So, in the example above, when we wanted our member to be able to use the 'Fire Storm' Ability (like, say he talked to some townsperson that said 'I'll teach you the Fire Storm Ability!'), we'd simply turn 'Fire Storm''s Limit Break Flag On.

**E. A Simple Use-Item-to-Permanently-Teach-Ability System like in Pokemon**

This is easier than Ideas #2, #3, and #4, and here's how to do it:
Data: Member Info: Change to Member Order
var86[Member Number] = var48[Insti Member] + 0 (these two commands find the current member's database number and store it in var86[Member Number])
Data: Member Info: Load
(you can add Conditions of var86[Member Number] here to make certain Abilities only teachable to certain members)
*Member Info: [Member] Ability + (Ability from Step #2) (this command teaches a member an Ability)
This obviously goes in the 'Result' slot of the Item's Direct Effect.

**F. Other ideas that I have not yet explicated in this Guide**

1. A Class System like in Final Fantasy 5.

Basically, you changed a party member to a class. Then, as you won battles as that class you leveled up in it, and each class level-up taught you a new ability. This is actually RPGM2's default Class System, so you should have no trouble at all doing it with what you've learned from this Guide.

2. A Class and Ability System like in Final Fantasy Tactics.

This is a difficult and lengthy one. Good luck. To explain, you changed a party member to a class, then gained points for that class from winning battles. You would then spend those points to learn abilities for your party members, just like how you spend money for equipment and items in other rpgs. To do this, you would need to use a Variable to keep track of each party member's current Class, you would need to make a Multiple Choice menu for changing that party member's current Class, you would need to use a Variable to keep track of each Class's Ability Points for every party member, you would need to make a Multiple Choice menu for learning the Abilities for every Class for every Party Member, and you would need to make one Flag for every Ability for every Class for every party member to remember if that Ability has been learned or not.

3. A Class System like in Brigandine.

In Brigandine, you started in a class tree that branched off in different ways upon promoting, and you gained all that class's abilities instantly upon becoming it. For example, a 'Fighter' could promote into a 'Cavalier', a 'Berserker', and a 'Samurai'. Each of the three promoted classes had their own statistical advantages (optional) as well as abilities. Then, those three (Cavalier, Berserker, and Samurai) promoted into other classes upon promotion. This isn't a very hard system to make - especially when compared to Idea #2. To do this, you would need to use a Variable to keep track of each party member's Class, and you would need to make a Multiple Choice menu for every Class change for every party member.

4. Various Ways of Obtaining Classes.

a. Final Fantasy Tactics/Brigandine - Each party member obtained new classes from gaining points/leveling-up in previous classes.
b. Final Fantasy 5 - All the members obtained new classes at key points in the story.
c. Breath of Fire 3 - All the members obtained new classes by talking to townspeople called "masters"
d. Anything you can imagine!

These obviously also work with equipment, items, magic scrolls in Idea #5, ability items in Idea #7, etc.

5. A Basic Equip-Scrolls-to-Use-Abilities System.

I've seen this done in lots of ways in numerous games including Final Fantasy 7, Final Fantasy 8, Final Fantasy 9, and Saiyuki: Journey to the West - so don't think there's only one satisfactory way to do it! If I were doing this I'd make it in the simplest way possible: equip a scroll on a member - they can use an ability while it's equipped, un-equip a scroll on a member - they can't use the ability while it's unequipped. The easiest way to do this is to re-name one of the Equipment Slots (default = Weapon, Head, Body, Shield, and Accessory) and use the technique in Idea #6.

6. Abilities from Equipment.

It does not have to be exactly like Final Fantasy 9's system! It's difficult to figure this out since there is a script that runs when something is equipped, but no script that runs when something is unequipped. My easy solution to that is making a script that runs when any/every equipment gets equipped, that 1) removes all equipment-abilities from that member 2) checks that party member's equipment by doing a plain Data: Load Member Info command and 3) teaches them all the abilities for their current equipment, in that order.

7. A Level-Up System like in Chrono Cross.

You simply leveled-up as you progressed through the game (every boss fight, but it could be other times to such as upon completing each dungeon, during certain story events, etc.). To do this, just forcefully level-up the party members whenever you want them to level-up. This Level-Up System is basically No Level-Up System, and is therefore extremely easy to do.

8. Choosing Which Stats to Increase Upon Leveling-Up like in Mario RPG.

This isn't too hard, but would take a while to make. Basically, upon leveling-up, the player would get to choose which stat(s) for the party member to increase more in. To do this, you would need to make a Multiple Choice menu to choose which stat(s) to increase more in for every party member.

9. Tools Not-In-Real-Time like in various Zelda games.

Basically, to solve puzzles and travel to distant places you needed to obtain and use various tools such as a hammer to smash rocks with or flippers to swim through water with. This really has nothing to do with battling and is completely disconnected from the DBS, except if you make those items teach abilities for in-battle. ;)

---

**IX. Frequently Asked Questions (FAQs)**

1. How come I can't change the Enemy's max HP?

I ran into this problem as well when altering my Enemies' HP in the default 'Member Start' script, and it took me forever to figure it out. RPGM2 will not allow you to alter an Enemy's max HP by doing this:
Battle: Substitute Target Attribute for Variable
Data: default var [Max Hit Points] = [Max Hit Points] + (any number above 0)
Battle: Substitute Variable for Target Attribute
But, it will allow you to alter an Enemy's max HP by doing this:
Data: Target: Attributes: Max HP+30
As a side note, I figured this out by displaying the [Max Hit Points] variable in a message window after loading it with Battle Substitute Target Attribute for Variable, and again after BOTH altering/saving it (each of the two scripts above) and re-loading it (re-load is to see if the new value really was saved). I then further tested it by attacking them and seeing if they had in fact gained the proper amount of Max HP and current HP.
The other thing to remember (may seem obvious) is that RPGM2 will not let you increase a Member's or Enemy's current HP over their Max HP, so if you want to raise both (like I did), you'll need to raise the Max HP first.

Lastly, take note that the Data: Target: Attributes command (only way to alter an Enemy's max HP) will NOT allow you to alter it by another variable, but rather only by set values.