

# 1g Extended Load Range Guide

Written by: Kyle P. Tarry

Date: 12/14/2004

The following is intended to be a guide to switching your first generation DSM's ECU code, in order to enable it to read twice as much "load" (airflow per revolution) as a stock DSM.

Primer: The stock 1g DSM ECU uses airflow per engine revolution to represent the internal load on the engine, and it uses this value to plot the current load and engine speed point on the fuel and timing maps. In stock form, the highest load level is roughly equal to 2.1 grams of airflow per engine revolution, which will correlate to *approximately* 15 psi of boost on the average 2.0 liter modified DSM. As such, in any situation where the car is receiving a true airflow signal (no airflow reduction through the means of a device like a SAFC or MAFT), the car will reach the maximum load value before your car reaches full boost. Once the vehicle reaches this value, the timing and fuel enrichment become static with regard to load changes, and will only vary with engine speed.

The effects of this can vary widely, but there are a couple primary effects that will often be found. First and foremost, this dissolves the ECU's ability to retard ignition timing at higher boost levels and on boost spikes, which can cause knock in these situations. The lack of dynamic timing and fuel enrichment changes can also cause a car to have inconsistent performance as ambient conditions change.

The only way to truly solve this problem, is to extend the range of load values that are mapped out on the internal timing and fuel enrichment maps. The ECU is capable of reading much more airflow than is given on the fuel and timing maps, but Mitsubishi chose to limit the map range to conserve space and give the best possible resolution within the useful range. This increase in range can be achieved by modifying a couple parameters in the ECU, the details of which will be explained below. However, the basic idea is as follows:

1. First, one needs to calculate and store the airflow per rev parameter to a location within the ECU, so that it can be used in the map axis calculations at a later point. There are several methods of doing this, and which method is used is at the discretion of the tuner. Each method and its reasoning can be found below.
2. Second of all, the load axis code needs to be modified, such that the map axis will now always be loaded off the value created above. This is a simple code change, which just changes the byte that the airflow per rev parameter is pulled from.
3. The fuel enrichment, timing, and octane maps must all be edited, in order to reflect the fact that their loads axis no longer are calibrated the way they were stock. This guide increases the readable load by a factor of two, and as such the map must be compressed by approximately the same factor; in other words, the 12 load levels that originally went to 2.06 grams of airflow per engine revolution, will now represent load levels up to twice that value (4.1 grams per revolution). The maps must be "compressed" to reflect this change. Several options for extended maps exist, and are included as a supplement to this guide.

One of the most common concerns of this conversion involves this change in the mapping strategy; since you are converting 12 discrete load points that previously covered 0.25 to 2.06 g/rev into 12 load points that now have to cover 0.5 to 4.1 g/rev, some resolution is lost in the process. However, rest assured that this **will not** affect the way your car runs, if it is tuned properly. In the vast majority of situations, a well tuned car will use fuel and timing maps that are fairly smooth and linearly progressive, and as such a large number of points are not necessary (a normal interpolation is sufficient). If one uses stock maps that have been rescaled but not smoothed out and tweaked, they may find slight problems will arise, due to the non-linearity of these maps. As such, I highly suggest that you use the first set of maps, which are stock-based by have some "cleanup" work done to them. Fully stock-based maps with no modifications are also included for users who feel the need to use them anyway, but it is not suggested.

**Step 1:** Creating a larger range airflow per rev value, and storing it to memory.

While the creation of a double-range load value could be a programming difficulty, it turns out that there is already a very simple method for this. When the ECU performs the check to see if it should initiate fuel cut, it actually creates a double-range load value for just this purpose. Once the fuel cut check is complete, however, this value is deleted, and its location is used for other temporary check and calculations during the routine. By storing this value to a different location, one is able to access it later, in order to use it as a map reference.

The stock code, for reference, is as follows:

```
D580: BD EB B8      JSR  LEBB8 ;routine to divide airflow per rev down
D583: D7 57        STAB L0057 ; stock airflow per rev to 0x57, to use for fuel cut check
```

The simplest method to change this is to simply store this airflow per rev value to a new location, which will not be deleted at a later time. This can be done using the following code:

```
D583: D7 54                STAB L0054 ; store to L0054 instead of L0057
...
D593: D1 54                CMPB L0054
...
D597: D6 54                LDAB L0054 ; load 54 instead of 57 for fuel cut (57 no longer exists)
```

It can be seen here that the code needs to be changed in three places; at D584, D594, and D598, the stock value of “57” needs to be changed to “54.” This is the simplest method, and it does not change any stock code strategies; it even retains the stock fuel cut “feature” (note that if you want to eliminate fuel cut, it can still be done through the normal methods).

One of the members of the DSM-ECU list, Carl Morris, has come up with an alternate method to doing this load value switch. He too proposes storing the extended load value to 0x54 (memory location 54, as seen above), but instead of using the fuel cut feature, he converts it to a “knock cut.” With this setup, the car will reach fuel cut any time the knock sum goes over a user-defined amount. Some people prefer to use this safety feature, so it is included below.

The new code, if this method is preferred, will look as follows:

```
D583: D7 54                STAB L0054 ; store to L0054 instead of L0057
D585+: no-ops
D597: D6 8B                LDAB L008B ;load knock sum to 8B (replacement of fuel cut code)
D599: C1 XX                CMPB $0020 ;fuel cut if knock sum is greater than XX (hex).
```

In case it is not clear, the code will be need to changed a bit more than in the last example. First, the value of 57 at location D584 needs to be changed to 54. Then, all of the code from location D585 to D596 needs to be changed to the no-operation command which is 01. Finally, starting at D597, the knock cut code can be placed, as seen above (D6 8B C1 XX). Note that the programmed must choose a threshold for the knock cut, and place it in the position occupied by “XX” above. As an example, if a threshold of 20 knock sum is desired, the value will need to be 14. If a threshold of 30 is desired, the value should be 1E. These “strange” values are due to the fact that the ECU code is in hexadecimal (base 16) numbers, and therefore all decimal numbers must be converted before they are put into the code.

That is it! Now we must change the location where the map’s axis is loaded, and we must replace the maps with values that have been recalibrated for the new range of loads.

**Step 2:** Changing the map axis code.

First, a quick look at the stock code is in order.

```
EC31: B6 01 5F      LDAA  L015F
EC34: 2A 04        BPL   LEC3A
EC36: 96 E2        LDAA  L00E2
EC38: 20 02        BRA   LEC3C
EC3A: 96 E3        LDAA  L00E3
```

This may look a bit confusing, but in reality it is very simple. The values *E2* and *E3* are the stock locations for the airflow per rev (load) values, and depending on current conditions, the ECU will choose one of the two to use for the map axis calculations.

Since we have stored a double-range value to memory location *54*, we want that to be loaded instead of the stock *E2/E3*. In order to achieve this, a slight rework of the code is necessary, shown below:

```
EC31: 96 54        LDAA  L0054
EC33: 20 07        BRA   EC3C
EC35: FF FF FF FF
EC39: FF FF FF
```

Once again, for simplicity: Instead of checking conditions and then loading one of the two stock airflow per rev values, we will now load *54* right off the bat. From here, we jump down to the stock axis code, which begins at EC3C. For clarity's sake, the bytes between EC34 and EC3C are represented as "*FF*," or empty space. This is not a requirement.

That's it for the map axis changes. Now, when the ECU loads what would normally be an airflow per rev value between 0.25 g/rev and 2.06 g/rev, it is actually picking up our extended value which will be between 0.5 and 4.1 g/rev. As such, and as mentioned above, the only changes that are left to be made are the changes to the fuel, timing, and "octane" maps.

This is one of the most difficult parts of the process; the load axis on the maps does not have a linear slope, nor do all the values on the extended map correlate to values on the stock-range map. As such, I have included two different sets of start-up maps with this guide, as a Microsoft Excel file. If you want to make your own maps, the following table will help you to understand how.

Load Row "Number"	1	2	3	4	5	6	7	8	9	10	11	12
Internal Raw Value	32	48	64	80	96	120	144	168	192	216	240	264
Stock Grams/Rev	0.25	0.38	0.50	0.63	0.75	0.94	1.13	1.31	1.50	1.69	1.88	2.06
Extended Grams/Rev	0.50	0.75	1.00	1.25	1.50	1.88	2.25	2.63	3.00	3.38	3.75	4.13

However, if you choose not to build your own maps from scratch (and there is no reason to, as I have provided a complete set of maps), then you can use the supplied Excel worksheet to copy and paste modified maps into your ECU code. First, a quick explanation of the attached spreadsheet, and then an overview of what is necessary to paste the maps into the ECU code:

The attached spreadsheet includes six maps, in two different categories. First, there is a set of maps that have been modified for the extended load range, and have also been modified and smoothed for optimal performance. Second, for users who want to use stock maps, is a set of the three maps, based on the stock ones, but modified for the extended range.

Which maps should you use? Really, there is no reason not to use the modified maps (the top three in the attached file). These maps are based on stock maps, and then were simply tweaked slightly based on experimental data and tuning theory to provide better performance, smoother operation, and less

detonation. These maps were specifically designed to directly replace a stock set of maps, without having an adverse effect on the engine's state of tune of tune.

If the stock maps are used, I offer no promises as to how the vehicle will react, since they were not intended to be used on a modified vehicle with extensive modifications and a modified ECU.

The process necessary to copy these files into the ECU code is very simple. The code section of each map (which is clearly labeled) needs to be selected and copied, and then the code needs to be pasted into a hex editor, starting at the location that is the start of the map. If your hex editor has options for the input format, this should probably be input as ASCII Hex..

The locations for all of the maps are as follows:

**Timing Map: Start: FD48  
Finish: FE07**

**Fuel Map: Start: FC3F  
Finish: FCE6**

**Octane Map: Start: FE08  
Finish: FE98**

*Note that a hex editor will not load the ECU file with the proper offset, so the map starts will probably be labeled as "7D48, 7C3F, and 7E08," respectively. If you are not using a 32k ROM image, then the numbers will once again be different (for a 16k rom, it will be labeled as 3D48, etc.)*