

ECUFlash ECU Definition files (.XMLs) for Mitsubishi Magna/Verada/Diamante

1 Archive contents

This archive contains work-in-progress ECUFlash compatible XML format definition files for a number of Mitsubishi 3rd generation Magna/Verada and Australian constructed USDM Diamante cars manufactured from 1997 to 2005.

The "Magna" subdirectory and its contents should be copied to the "Mitsubishi" subdirectory beneath the "rommetadata" directory in the ECUFlash installation. If you have placed XMLs from other sources for the same ECUs in this directory structure, remove them before attempting to use these definitions otherwise ECUFlash will arbitrarily decide which definition to use. To work around the access controls applied to the standard ECUFlash installation directory as of Windows Vista:

- 1) create a suitable subdirectory elsewhere on your system in a more accessible location;
- 2) copy the contents (all subdirectories and files) of the "rommetadata" subdirectory in the ECUFlash installation to the subdirectory created in 1) above;
- 3) Open ECUFlash, select the "Options" item from the "File" menu;
- 4) Select "Metadata Directory" from the option list on the "Options" dialog and then enter (or browse to using the "..." button) the path to the subdirectory created in 1) above;
- 5) close the "Options" dialog using the "X" button in the top right hand corner of the dialog then close ECUFlash;
- 6) copy the Magna definition files into the "Mitsubishi" subdirectory of the subdirectory created in 1) above.

Where known, the <ecuid> field in each definition has been populated with the ECU part number the ROM was taken from.

2 Copyright

This document is © Andrew I MacIntyre 2015-2017 and all rights to it are reserved.

The ECUFlash definition (.XML) files are released to the public domain.

3 YOU HAVE BEEN WARNED

1. You use these files at your own risk. While I have endeavoured to ensure that the information in them is correct, I only own one of the vehicles from which the base ROMs were sourced and I am in no position to verify the correctness for all ROMs through in car testing. Much of the information has however been cross-checked against several different reference sources, except for the H8 definitions (see item 7 below).
2. These definitions were developed using ECUFlash v1.44 (build 3721). Older versions (particularly pre-v1.40) may not accept these definitions. Please report any problems encountered using these definitions with v1.44 or newer; problems encountered with older ECUFlash versions are unlikely to be addressed.
3. L/W series automatic ECUs can be problematic as there are two versions of the ECU hardware in active use: the J series ECU hardware and a version otherwise identical except that it responds to the Evo 9 reflashing protocol. The same part numbers are used for both types and the same ROMs are used for each part number regardless of reflashing protocol. Early L series cars are the most likely to use J series ECU hardware and require the use of the customised "read template" file to read them as described below. This presents a problem as

ECUFlash definitions only support identifying a single reflashing protocol for flashing a ROM back to the ECU and the ROMs are indistinguishable so separate definitions cannot be created. As the majority of L/W series cars seem to have ECUs that respond to the Evo 9 reflashing protocol, the L/W series ROM definitions in this package specify this protocol. Normal practice with L/W series ECUs would be to attempt to read them first with the Evo 9 vehicle type and if that fails attempt to read with the customised read template as described above for J series ECUs. If ECUFlash refuses to write the ROM back to the ECU, you will need to edit the definition file to change the `<flashmethod>` entry from *mitsukernelocp* (Evo 9 protocol) to *mitsukernel* (Evo 7/8 protocol).

4. By default ECUFlash does not support reading/flashing the full 512kB ROM from the SH2 based ECUs in J series automatic Magnas however the full ROM can be read through the use of a customised "read template" file, available separately. There is no way to read the full ROM from these ECUs with Evoscan. As a consequence, many people read/flash only the first 256kB of the ROM (which contains the engine code/maps) using the Evo 7/Evo 8 vehicle type (in either ECUFlash or Evoscan). If you wish to use definitions in this package with such incomplete ROMs, you must edit the definition you wish to use to change the `<memmodel>` entry from *SH7055* to *SH7052*. If you don't make this change, ECUFlash will not recognise the ROM file and will trash the definition file. Note that doing this means that updating TCU information is not possible.
5. Always read the ECU before flashing and save the ROM file just read as a backup. That way if something goes wrong the previous state can be restored. A prudent person will also read the ROM again after writing it, and compare it to the ROM that was written – while ECUFlash does perform its own check after writing it doesn't hurt to be doubly sure. Ensuring that the car battery is fully charged before attempting to reflash improves the reliability of the reflashing operation and can be critical for reliably reflashing H8 based ECUs.
6. All J, L and W series manual ECUs should read using the Evo 7/8 vehicle type in either ECUFlash or Evoscan; ROM images for these ECUs are 256kB.
7. Several definitions for H8 (16bit) ECUs are included, although these are not as well developed as the SH2 definitions. In particular, please note that these definitions were developed by pattern matching maps from SH2 ROMs and considerable care should be exercised when using ROMs modified via these definitions.

H8 ECUs were used in the F & H series cars. The reflash connector under the dash is fitted to all H series cars but only late F series cars (currently understood to have build dates from May-June 1998 onwards). Reflashing these ECUs requires an OpenPort 2.0 cable; an OpenPort 1.3U cable can be used to read these ECUs with ECUFlash 1.40 or earlier but they can't be safely written with this cable. These ECUs can be read using the Evo 5/6 vehicle type.

F series cars lacking the reflash connector, as well as E series cars, have ECUs with an earlier microcontroller design which is effectively not flashable though some cars just before the reflash connector became a standard fitting may also have got reflashable H8 ECUs. Unless the largest chip (two chips in automatic ECUs) on an F series ECU circuit board are clearly marked with the code **MH7202F**, the ECU has to be assumed non-reflashable.

8. I am not a provider of tuning services. If you have questions about changing the contents of maps accessed through these definitions, other than comments made in this document, please seek an appropriate forum for your questions.
9. I do this as a hobby, with limited time availability. While I welcome problem reports and queries about functionality and will address them as I can, please understand that I can give no

guarantee about being able to respond to queries with any particular level of timeliness. That said, I am human and do screw things up from time to time so please don't be backward in reporting problems!

4 Reflashing Hardware and Software

4.1 OpenPort cables

This series of cables were purposely developed by Tactrix for logging and reflashing Subaru and Mitsubishi ECUs. There are two variants currently in widespread use: v1.3 and v2.0.

v1.3 cables are an older design that is no longer manufactured by Tactrix but is still being manufactured and sold by Evoscan in 3 variants: v1.3D (logging only), v1.3R (Evoscan only reflashing) and v1.3U (ECUFlash reflashing).

v2.0 cables are manufactured by Tactrix and are available directly from them or via resellers including Evoscan. v2.0 cables can be used for reflashing a wider variety of ECU hardware than v1.3 cables and include support for logging direct to a microSD card, however are considerably more expensive and provide somewhat lower logging rates than v1.3U cables. All v2.0 cables can be used for reflashing with ECUFlash however separate extension cables are required to connect the main part of the cable (which connects to the car's diagnostic connector) with the reflash connector.

4.2 ECUFlash

ECUFlash, developed and published by Tactrix in the USA, is a Microsoft Windows application that has two principle functions:

- reading and reflashing ECU hardware; and
- viewing and editing parametric data in ECU ROM images.

ECUFlash uses XML formatted text files known as definition files to describe where to find items of parametric data in ROM images, how to convert the bytes and words from the ROM image into a form meaningful to human interpretation (referred to as "scaling") and how to organise groups of data items into tables (or "maps") where that would be meaningful. In order to read ECUs ECUFlash uses XML formatted files known as read templates which describe the microcontroller (and thus the size of ROM image) and reflashing protocol. ECUFlash includes a base set of definition files and read templates for a variety of Subaru and Mitsubishi cars.

Tactrix website: <http://www.tactrix.com/>

4.3 Evoscan

Evoscan, developed and published by Hamish Ahern in New Zealand, is a Microsoft Windows application primarily intended for data logging of Mitsubishi cars (principally the Lancer Evolution models) with some other brands and models also supported. Evoscan includes support for reading and reflashing Lancer Evolution 7, 8 and 9 ECUs, however doesn't provide any support for viewing or editing parametric data in ROM images.

Evoscan website: <http://www.evoscan.com/>

5 Car support

5.1 Magna and Verada (Australian Market)

TE Magna + KE Verada

Not supported.

TF Magna + KF Verada

F series cars with a factory fitted reflashing connector have H8 ECUs which are reflashable, however there is an issue with the reflash connector wiring which needs modification¹ before the ECU can be successfully read or reflashed. Cars built before the reflashing connector was added to the loom have non-reflashable ECUs (though as noted above, there may be some exceptions).

This package release contains definitions for three F series ROMs:

- two 3.0/ 4 speed automatic; and
- one 3.5/ 4 speed automatic.

Reflashing the automatic transmission control function is currently not supported, as it is in a separate ROM from the engine function which is not easily accessible using the standard reflashing tools.

TH Magna + KH Verada

All H series cars have reflashable H8 ECUs and a reflashing connector. This package release contains definitions for two H series ROMs:

- one 3.0/ 4 speed automatic; and
- one 3.5/ manual.

Reflashing the automatic transmission control function is currently not supported, as it is in a separate ROM from the engine function which is not easily accessible using the standard reflashing tools.

TJ Magna + KJ Verada

All J series cars have reflashable SH2 ECUs and a reflashing connector. This package release contains definitions for twelve J series ROMs:

- one 3.0/ manual;
- one 3.0/ 4 speed automatic;
- five 3.5/ manual (including Ralliart);
- two 3.5/ 4 speed automatic;
- two 3.5/ 5 speed automatic (including Ralliart); and
- one 3.5/ 5 speed automatic AWD.

¹ Details of the loom modification required to support flashing are available from several online forums.

TL/TW Magna + KL/KW Verada

All L/W series cars have reflashable SH2 ECUs and a reflashing connector. This package release contains definitions for seven J series ROMs:

- one 3.5l manual;
- two 3.5l 4 speed automatic;
- two 3.5l 5 speed automatic; and
- two 3.5l 5 speed automatic AWD.

Please refer to item 3 in the *YOU HAVE BEEN WARNED* section concerning reflashing protocols for these ECUs.

5.2 Diamante (US Market)

The Australian made Diamante is the same basic vehicle as the Australian Verada apart from the obvious change to left hand drive. There are however important differences which result in the Diamante ROMs being noticeably different to Magna/Verada ROMs, including:

- the engine was fitted with a knock sensor so the ROMs have both high and low octane fuel and ignition maps;
- the "Federal spec" and "California spec" emissions compliance required fitting two and four oxygen sensors respectively;
- the ROMs include OBD2 protocol emulation; and
- the body electronics were built around the ETACS system rather than the BEM used in the Magna/Verada.

There may be other differences in the ECU interface electronics as well, as I have tested a Diamante ECU (MY2002) with a contemporary Magna ROM in a KF Verada and while the engine appeared to operate normally, the automatic transmission exhibited erratic and abnormal shift characteristics. Operation of this car with the same Magna ROM in a J series Magna ECU was completely normal².

The Diamante appears to have been fitted with a reflashing connector from the 1997 model year onwards. The information I have accumulated suggests model years 1997 to 2000 had H8 ECUs and model years 2001 to 2004 had SH2 ECUs. At least 1997 and 1998 model year cars appear to have had the ECU and TCU functions in separate units (both H8 based). 2000 model year cars had both ECU and TCU functions combined in a single unit with two H8 processors in the same configuration as the H series Magna/Verada. I have no information at this time about the ECU/TCU arrangement in 1999 model year cars. All H8 based TCUs are reflashable and the reflash signal is connected to a second pin in the reflashing connector. Reflashing the TCU function with OpenPort cables would be possible with the ECU and TCU reflashing pins swapped in the reflash connector, or by fitting a switch to allow the desired function to be reflashed via the ECU reflash pin only.

This package release contains definitions for four Diamante ROMs:

- four 3.5l 4 speed automatic (model years 2001, 2002, 2003 and 2004).

It is possible that the MY2004 Diamante ECUs may be subject to the same reflashing protocol issue as the L/W series Magna/Verada - please refer to item 3 in the *YOU HAVE BEEN WARNED* section concerning reflashing protocols for more information.

² This ROM had been modified to neutralise the barometric pressure CEL as well as the immobiliser.

6 General Information About Magna/Verada ECUs

6.1 Periphery bits

The Periphery bits are option "switches" which enable/disable certain functionality in the ECU code, such as the operation of the immobiliser.

There are six sets of periphery bits, known as Periphery 00, 0, 1, 2, 3, and 4. They are also known by the "address", being F9A, FAA, FBA, FCA, FDA and FEA (for SH2 ECUs) respectively.

Each set of periphery bits is organised as an array of eight 16 bit words, with the active word being identified by the value of MUT Request 34 which is more generally known as the Spec Number. In all the Magna/Verada/Diamante ROMs checked so far, the Spec Number is fixed and access to this value has been included in the definitions.

The definitions in this package provide access to periphery information in two different ways:

- as a map of bits for each periphery set; and
- as a 3D map of hexadecimal word values for all periphery sets.

To correctly change a periphery bit, the Spec Number is required to identify which value to change.

As the operation of particular periphery bits is confirmed, the descriptions for the bits will be updated. There is now also some coverage of periphery bits for the automatic transmission (TCU) code.

6.2 Identifying ROMs

Mitsubishi has maintained standardised locations in the ROMs for version identifiers within each ECU architecture, though the exact locations and form of the identifier are different for each architecture.

If the ROM you read from a car isn't one for which a definition is available here, first save the ROM without attempting to create a definition file in ECUFlash (unless of course you're familiar with doing so).

You then need to inspect the ROM file with a hex editor, such as HxD (<http://mh-nexus.de/en/hxd/>).

For an H8 based ECU (e.g. TH auto or manual) the 4 bytes starting at offset 0x21A, expressed in hexadecimal, form the ROM ID. ROMs for this type of ECU will always be 128kB in size, and the ROM ID will have 2 as the first digit. **NB:** when entered into a definition, the address of the ROM ID needs to be entered as "1021A" rather than "21A".

For SH2 based ECUs (TJ onwards auto or manual), the 4 bytes starting at offset 0xF52 form the ROM ID. ROMs from auto ECUs will be 512kB in size and have 8 as the first digit of the ID (e.g. 85490001) while ROMs from manual ECUs will be 256kB in size and have 9 as the first digit of the ID (e.g. 91760000).

Feeding the ID from an unknown (to ECUFlash) ROM into your favourite search engine may turn up someone else's attempt at a definition file.

By all means contact me for assistance with a ROM for which you can't find a definition, though please note comments above about my ability to respond.

IMPORTANT NOTE: I have encountered one example where the above approach for ECU identification has been misleading: there are two distinct but subtly different ROMs with the otherwise standard ID of 87200005, one of which is actually a variant of 87200004. The definition files for these ROMs use more extensive identification sequences in order to distinguish the ROMs.

6.3 MUT protocol logging and map adjustment

Intake Air Temperature (IAT)

There are two widely identified MUT requests for logging IAT: 11 and 3A

Request 11 (Evoscan name: "MAF Air Temp Scaled") is the IAT scaled to SI units with a nominal value range of -40°C to 215°C. The standard sensor calibration provides for an effective value range of -40°C to 120°C. This request is appropriate for logging as the basis for all IAT map values, axis values and scalars other than the IAT Sensor Calibration map.

Request 3A (Evoscan name: "Air Temperature") is the raw ADC input value and reflects a range of 0V to 5V. This request is only appropriate for logging as the basis for adjustments to the IAT Sensor Calibration map.

Coolant Temperature

There are two widely identified MUT requests for logging coolant temperature: 07 and 10.

Request 07 (Evoscan name: "Coolant Temp") is the raw ADC input value and reflects a range of 0V to 5V. This request is only appropriate for logging as the basis for adjustments to the Coolant Temperature Sensor Calibration map.

Request 10 (Evoscan name: "Coolant Temp Scaled") is the coolant temperature scaled to SI units with a nominal value range of -40°C to 215°C. The standard sensor calibration provides for an effective value range of -40°C to 120°C. This request is appropriate for logging as the basis for all coolant temperature map values, axis values and scalars other than the Coolant Temperature Sensor Calibration map.

ECU Load

ECU Load is calculated directly from the MAF signal via the Raw Airflow to Engine Load Conversion Factor scalar and is used as a means of representing demand for air flow similar to the way that manifold absolute pressure (MAP) is used in speed density based ECU systems. The values for Raw Airflow to Engine Load Conversion Factor would appear to have been chosen by Mitsubishi to produce maximum base ECU Load values of approximately 80 with stock engines (the Ralliart being the exception).

There are actually 4 variations of ECU Load used throughout the ECU code:

- the base Load value (referred to as ECU Load in Evoscan and also called Raw Load in some references, returned by MUT request 1C);
- an IAT compensated Load value (returned by MUT request 1F);
- a barometric pressure compensated Load value (not loggable with standard MUT); and
- a Load value with both intake air temperature and barometric pressure compensation applied (returned by MUT request 1D).

Most 2D and 3D maps with Load as an axis value use the base Load value, including the fuel maps, however there are some important exceptions.

In particular, ignition map Load axes use a value based on either the barometric pressure compensated Load value (above 25°C) or the intake air temperature and barometric pressure compensated Load value (up to 25°C) with some added RPM based throttle position compensation. This specialised Load value can only be logged via 2-byte logging.

6.4 2-Byte Logging

The Mitsubishi H8 and SH2 ECUs supported by these definitions store nearly all operational variable data as 16 bit words in big endian format (i.e. the most significant or "high" byte of a word is at the lowest memory address). The MUT protocol however only supports returning values as single 8 bit bytes so the MUT values are scaled down from words to bytes with a corresponding loss of precision. This loss of precision can be an impediment to maximising the effectiveness of a tune.

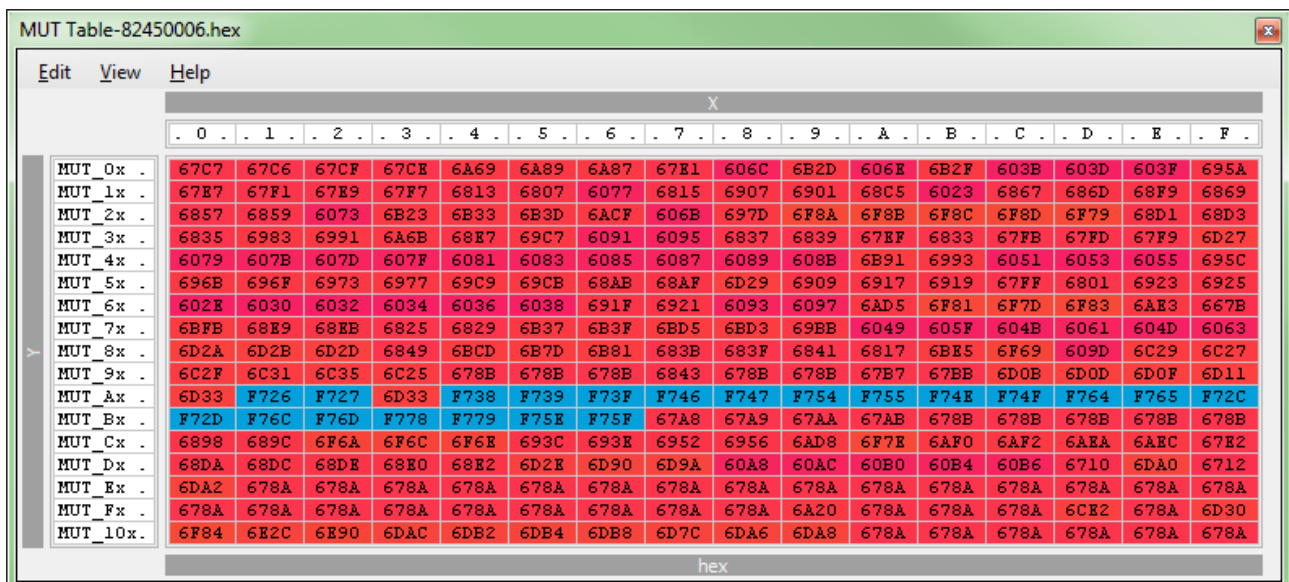
Evoscan supports a technique known as 2 byte logging by which both upper and lower bytes of a word value are retrieved via separate MUT requests and then combined to recreate the original word value. While this does allow access to the full precision of the source variables, as a result of the delay between the two individual MUT requests it is possible to have the source variable updated between the first and second requests with the consequence that the resulting value is completely out of character compared to the preceding and subsequent logged values. This effect is most noticeable whenever the source variable has raw values which are close to multiples of 256.

In addition to getting better resolution for otherwise loggable values, there are also a number of very useful variables which aren't loggable except by using this technique.

To use 2 byte logging requires modifying the Evoscan log item configuration and the MUT Table (via ECUFlash) in the ROM.

I won't address the Evoscan log item configuration in this document so if you are interested in using this technique you will need to find other resources for details about the necessary changes to Evoscan's configuration.

The MUT Table (see Figure 1) is the list of memory addresses that the ECU MUT protocol support code uses to find data to return for each different MUT request. For each value you wish to log as a 2 byte value, separate MUT requests need to be identified to be used for the high and low bytes and the address changed to point to the target variable data. Only MUT IDs from 00 through BF can be used for this purpose with IDs 00&01, 02&03 and 04&05 in particular being widely used. In general I would recommend avoiding MUT IDs used for regularly logged "standard" data values.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
MUT_0x	67C7	67C6	67CF	67CE	6A69	6A89	6A87	67E1	606C	6B2D	606E	6B2F	603B	603D	603F	695A
MUT_1x	67E7	67F1	67E9	67F7	6813	6807	6077	6815	6907	6901	68C5	6023	6867	686D	68F9	6869
MUT_2x	6857	6859	6073	6B23	6B33	6B3D	6ACF	606B	697D	6F8A	6F8E	6F8C	6F8D	6F79	68D1	68D3
MUT_3x	6835	6983	6991	6A6B	68E7	69C7	6091	6095	6837	6839	67EF	6833	67FB	67FD	67F9	6D27
MUT_4x	6079	607B	607D	607F	6081	6083	6085	6087	6089	608B	6B91	6993	6051	6053	6055	695C
MUT_5x	696B	696F	6973	6977	69C9	69CB	68AB	68AF	6D29	6909	6917	6919	67FF	6801	6923	6925
MUT_6x	602E	6030	6032	6034	6036	6038	691F	6921	6093	6097	6AD5	6F81	6F7D	6F83	6AE3	667B
MUT_7x	6BFE	68E9	68EB	6825	6829	6B37	6B3F	6BD5	6BD3	69BB	6049	605F	604B	6061	604D	6063
MUT_8x	6D2A	6D2B	6D2D	6849	6BCD	6B7D	6B81	683B	683F	6841	6817	6BE5	6F69	609D	6C29	6C27
MUT_9x	6C2F	6C31	6C35	6C25	678B	678E	678B	6843	678B	678E	67B7	67BE	6D0B	6D0D	6D0F	6D11
MUT_Ax	6D33	F726	F727	6D33	F738	F739	F73F	F746	F747	F754	F755	F74E	F74F	F764	F765	F72C
MUT_Bx	F72D	F76C	F76D	F778	F779	F75E	F75F	67A8	67A9	67AA	67AB	678B	678B	678B	678B	678B
MUT_Cx	6898	689C	6F6A	6F6C	6F6E	693C	693E	6952	6956	6AD8	6F7E	6AF0	6AF2	6AEA	6AEC	67E2
MUT_Dx	68DA	68DC	68DE	68E0	68E2	6D2E	6D90	6D9A	60A8	60AC	60B0	60B4	60B6	6710	6DA0	6712
MUT_Ex	6DA2	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A	678A
MUT_Fx	678A	678A	678A	678A	678A	678A	678A	678A	678A	6A20	678A	678A	678A	6CE2	678A	6D30
MUT_10x	6F84	6E2C	6E90	6DAC	6DB2	6DB4	6DB8	6D7C	6DA6	6DA8	678A	678A	678A	678A	678A	678A

Figure 1: MUT Table for ROM 82450006

Table 1 identifies the addresses to use for some commonly logged items for the SH2 ROMs supported in this package. Unfortunately I'm not able to provide this information for the H8 ROMs.

Table 1: 2 Byte Logging MUT Table addresses

ROM IDs	RPM	Uncomp Load	Comp Load IAT	Comp Load Baro	Comp Load IAT + Baro	Comp Load Ignition Map	Airflow Hz	
82450006, 87200002, 87200005a	684E, 684F	6870, 6871	6872, 6873	6874, 6875	6876, 6877	6A84, 6A85	68C8, 68C9	
87190003, 87200004, 87200005b	685E, 685F	6880, 6881	6882, 6883	6884, 6885	6886, 6887	6A94, 6A95	68D8, 68D9	
91760000, 98320000, 98320001	875E, 875F	8780, 8781	8782, 8783	8784, 8785	8786, 8787	8994, 8995	87D8, 87D9	
97180003, 97180004, 97190002	876E, 876F	8790, 8791	8792, 8793	8794, 8795	8796, 8797	89A4, 89A5	87E8, 87E9	
82000001, 82000002, 82030001, 82030002, 85490000, 85490001	6856, 6857	6878, 6879	687A, 687B	687C, 687D	687E, 687F	6A90, 6A91	68D0, 68D1	
87410004	6912, 6913	6934, 6935	6936, 6937	6938, 6939	693A, 693B	6B4C, 6B4D	698C, 698D	
80350003	6952, 6953	6974, 6975	6976, 6977	6978, 6979	697A, 697B	6BA8, 6BA9	69D2, 69D3	
81930003	6A20, 6A21	6A42, 6A43	6A44, 6A45	6A46, 6A47	6A48, 6A49	6CBA, 6CBB	6AA2, 6AA3	
83890006	6A62, 6A63	6A84, 6A85	6A86, 6A87	6A88, 6A89	6A8A, 6A8B	6D0E, 6D0F	6AE6, 6AE7	
91970002	8766, 8767	8788, 8789	878A, 878B	878C, 878D	878E, 878F	89A0, 89A1	87E0, 87E1	

NOTE: Addresses are in hexadecimal with the first address being for the High byte and the second address for the Low byte. To enter these values into a MUT Table cell will require using a prefix of "0x" to signify they are hexadecimal values.

6.5 Logging TCU parameters

Unfortunately I have not been able to get Evoscan to log TCU parameters from a J/L/W series (SH2) automatic transmission ECU through the TCU interface, though I have been able to do so with a simple MUT logger I wrote myself. It is however possible to log J/L/W series TCU data with Evoscan by copying the MUT request addresses from the TCU MUT table to unused entries in the engine MUT table. In my opinion this is actually more practical than logging through the TCU interface as both engine and TCU data can be logged at the same time.

The following table contains a list of TCU MUT IDs for information I have identified along with the recommended Evoscan eval formula:

TCU MUT ID	Item description	Evoscan formula (units)
11	Input shaft RPM	x*30 (RPM)
12	Output shaft RPM	x*30 (RPM)
13	AT fluid temperature	x-40 (°C)
1A	Torque converter solenoid duty cycle	x*0.392157 (%)
1B	Shift solenoid 1 duty cycle	x*0.392157 (%)
1C	Shift solenoid 5 duty cycle	x*0.392157 (%)
1D	Shift solenoid 4 duty cycle	x*0.392157 (%)
1E	Shift solenoid 2 duty cycle	x*0.392157 (%)
1F	Shift solenoid 3 duty cycle	x*0.392157 (%)
20	Engine RPM	x*30 (RPM)

Output shaft RPM can be converted to vehicle speed in km/h via the following identity:

$$Speed = \frac{Output\ shaft\ RPM * 60}{Final\ drive\ ratio * Tyre\ revs/km}$$

For example, the speed of a car with a final drive ratio of 3.684:1 running on tyres rated at 495 revs/km with the output shaft turning at 2700 RPM would be calculated as:

$$Speed = \frac{2700 * 60}{3.684 * 495} \approx 89\ km/h$$

Note that this MUT table manipulation technique for logging TCU parameters cannot be used with H8 automatic transmission ECUs even if the TCU ROM can be read and the TCU MUT table identified as the engine MUT logging has no access to the TCU RAM variables.

7 Common configuration changes

7.1 Neutralising the immobiliser

Neutralising the immobiliser is commonly done to work around issues with the immobiliser system in the car, to replace a faulty ECU for which re-coding the immobiliser is not practical or to permit a track car to operate with a minimalist wiring loom.

There are two levels of immobiliser neutralisation in Mitsubishi's H8 and SH2 ROMs:

1. the ECU requires a connection to the immobiliser system but doesn't match codes with it; and
2. the ECU doesn't require a connection to the immobiliser system.

F series Magnas and Veradas had immobiliser systems which were not coded to the ECUs and in these cars the ROMs are already in the first state, however there is other code in the ROM that performs some sort of signalling between ECU and immobiliser system so level 2 neutralisation is required to disable any form of immobiliser function in these cars.

The first level is the same state in which new replacement ECUs would be delivered to a dealer, with the expectation that when fitted the full immobiliser function would be restored by flashing the immobiliser code that matches the vehicle's factory immobiliser system. This state is achieved simply by setting the immobiliser code to FFFF (hexadecimal, entered as 0xFFFF) as illustrated in Figure 2.

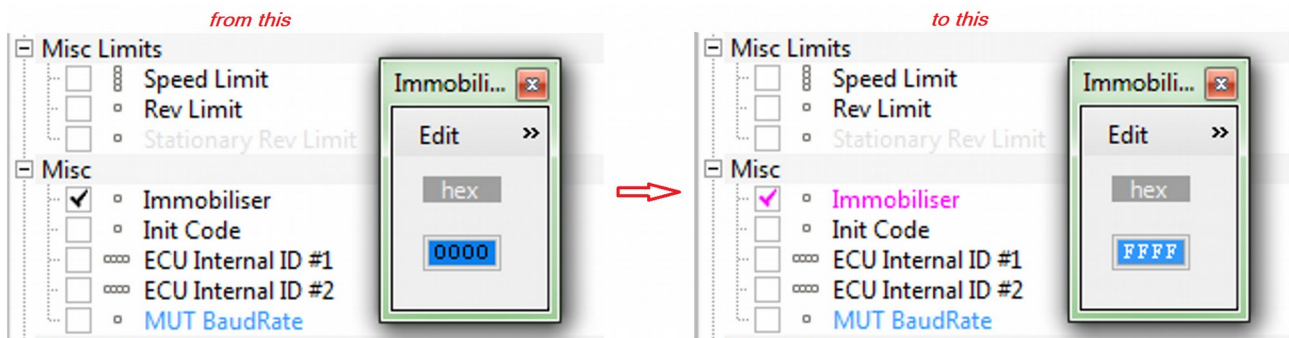


Figure 2: Immobiliser neutralised by setting factory default code

If the second level of neutralisation is required, such as for a track car with no immobiliser system at all installed, the immobiliser function needs to be disabled completely in the Periphery Bits.

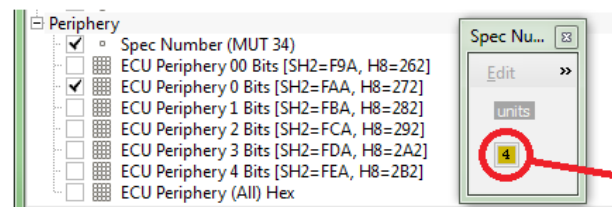
There are two alternate approaches to doing this for any given ROM:

1. disable via the specific periphery bit for the car's Spec Number; or
2. change the Spec Number to 7 or, in ROMs common to both TCL and non-TCL cars, 6.

To disable the specific periphery bit for the car:

1. open the Spec Number (MUT 34) 1D table and note the value;
2. open the ECU Periphery 0 Bits table;
3. in the Enable Immobiliser row, set the value in the column matching the Spec Number to 0.

In the example shown in Figure 3, the immobiliser is disabled for Spec Number 4. Figure 4 shows the equivalent change using the ECU Periphery (All) Hex table.



ECU Periphery 0 Bits [SH2=FAA, H8=272]-82450006.hex

Edit View Help

Spec Number (MUT 34)

	0	1	2	3	4	5	6	7
bit.15	0	0	0	0	0	0	0	0
bit.14	0	0	0	0	0	0	0	0
Enable TCL	0	1	0	1	0	0	1	0
Enable Purge Control	1	1	1	1	1	1	1	1
bit.11	0	0	0	0	0	0	0	0
bit.10	0	0	0	0	0	0	0	0
bit.9	1	1	1	1	1	1	1	1
bit.8	0	0	0	0	0	0	0	0
bit.7	0	0	0	0	0	0	0	0
bit.6	1	1	1	1	1	1	1	1
bit.5	0	0	0	0	0	0	0	0
Enable Closed Loop	1	1	1	1	1	1	1	1
Enable Immobiliser	1	1	1	1	0	0	0	0
bit.2	0	0	0	0	0	0	0	0
bit.1	0	0	0	0	0	0	0	0
bit.0	0	0	0	0	0	0	0	0

ECU Periphery 0 Bits [SH2=FAA, H8=272]

Figure 3: Disabling immobiliser for Spec Number 4

ECU Periphery (All) Hex-82450006.hex

Edit View Help

Periphery Set

	00/F9A	0/FAA	1/FBA	2/FCA	3/FDA	4/FEA
Spec Number (MUT 34) 0	0021	1258	0000	0000	0000	0022
1	0021	3258	0000	0000	0000	0023
2	0021	1258	0000	0000	0000	0000
3	0021	3258	0000	0000	0000	0024
4	0021	1258	0000	0000	0000	006C
5	0021	1250	0000	0000	0000	0000
6	0021	3250	0000	0000	0000	0000
7	0021	1250	0000	0000	0000	0000

hex

ECU Periphery (All) Hex-82450006.hex*

Edit View Help

Periphery Set

	00/F9A	0/FAA	1/FBA	2/FCA	3/FDA	4/FEA
Spec Number (MUT 34) 0	0021	1258	0000	0000	0000	0022
1	0021	3258	0000	0000	0000	0023
2	0021	1258	0000	0000	0000	0000
3	0021	3258	0000	0000	0000	0024
4	0021	1250	0000	0000	0000	006C
5	0021	1250	0000	0000	0000	0000
6	0021	3250	0000	0000	0000	0000
7	0021	1250	0000	0000	0000	0000

hex

Figure 4: Disabling immobiliser for Spec Number 4 via the periphery hexadecimal view

7.2 Copying the immobiliser coding

It is possible to copy the immobiliser coding from one ROM to a different ROM in the same family, however this sometimes doesn't work for as yet unknown reasons.

In addition to copying the immobiliser code, the lower 7 bits (bits 0 to 6) of the Periphery 4 bit set in the source ROM need to be copied to the target ROM based on the Spec Numbers in the source and target ROMs. As far as I can determine, no Magna/Verada ROM has any of bits 7-15 set in Periphery 4 so the whole 16 bits can simply be copied and pasted. The later Diamante ROMs do have some of the upper bits set in Periphery 4 so more care will be required to only copy the required bits. Figure 5 illustrates copying the Periphery 4 bits from ROM 82450006 (Spec Number 4) to ROM 87200002 (Spec Number 3).

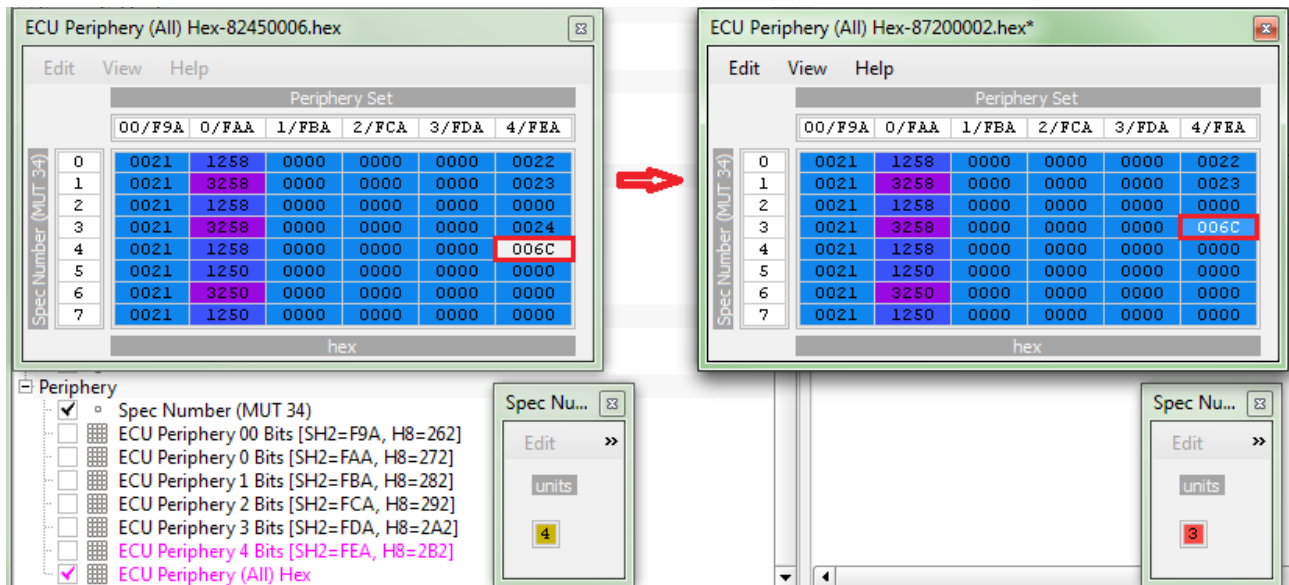


Figure 5: Copying Periphery 4 immobiliser bits between ROMs

7.3 Using a J series ECU in E or F series cars

F series cars, both manual and automatic, can operate with H and J series ECUs if the immobiliser is completely (i.e. second level) neutralised. E series manual cars may also operate with a F, H or J series ECU, however E series automatic transmissions are sufficiently different to the later cars that they don't operate properly with later ECUs (F, H or J). L/W series ECUs are best avoided for use in these cars.

The E and F series cars were equipped with MAFs that lacked a barometric pressure sensor which all H and J series ROMs expect, resulting in the ECU showing the Check Engine Light (CEL) even though the ROM uses the same default barometric pressure value as the original ECU and the engine will run as well as it would with the original ECU.

In J series ROMs, the CEL caused by the absence of the barometric pressure sensor can be neutralised by setting the Barometric Pressure Sensor Out-Of-Range CEL and Barometric Pressure Sensor Absolute Limits tables as illustrated in Figure 6.

While there is no technical reason the CEL could not also be neutralised in H series ROMs as well, I am not in a position to locate the equivalent control parameters for them and I'm not aware of any other published H8 definitions with these controls.

E & F series cars also lack both EGR and purge control functions so these should also be disabled.

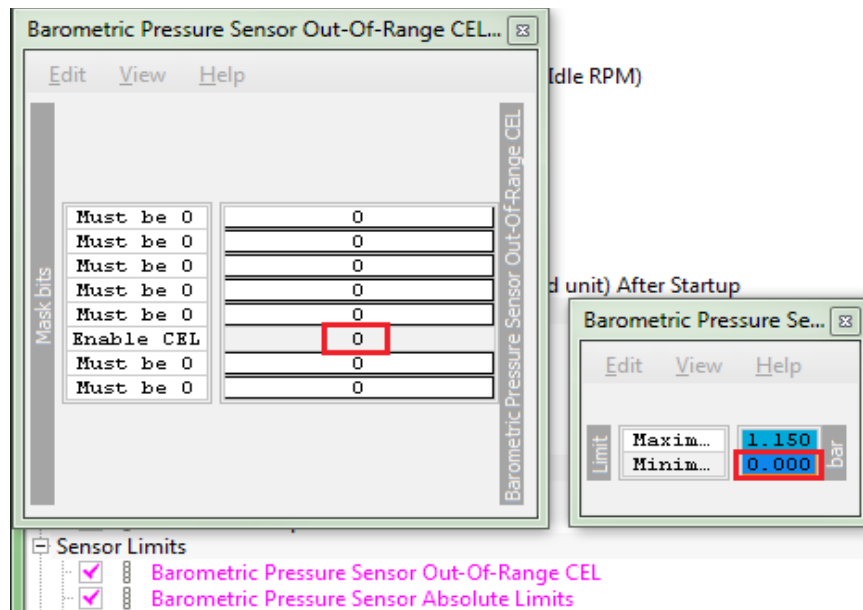


Figure 6: Neutralising Barometric Pressure Sensor CEL

7.4 Converting non-Tiptronic shift to Tiptronic

All F, H, and J series automatic transmissions are electrically the same and therefore can be operated in Tiptronic shift mode with the appropriate harness and ROM. All J series automatic ROMs are Tiptronic capable and when appropriately configured will correctly operate the transmission in this mode with the appropriate wiring harness, however it has been identified that only those ECUs delivered in cars with factory enabled Tiptronic shift have the necessary output circuitry to operate the individual gear lights in a Tiptronic instrument cluster.

The Tiptronic shift function is enabled via the Auto Trans Periphery 40010 table as illustrated in Figure 7.

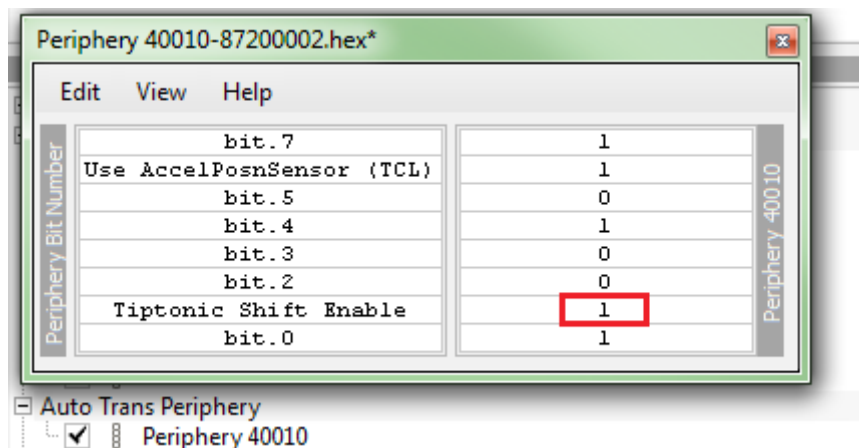


Figure 7: Enable Tiptronic Shift

It should be noted that the only 5 speed automatics without TCL from the factory were the AWDs. If converting a 4 speed car without TCL to 5 speed, the TCL function needs to be disabled in the engine Periphery 0 bits (visible in Figure 3) and the Use AccelPosnSensor (TCL) bit in Auto Trans Periphery 40010 (visible in Figure 7) needs to be set to 0. A 4 speed Tiptronic wiring loom is likely to also need additional wires from the ECU to the transmission and the instrument cluster to support the additional shift solenoid and gear lights respectively. If a TCL equipped wiring loom and throttle body are used, both TCL related options are probably best left enabled.

8 Notes on ECU map operation

8.1 Air flow limits

The Max Allowed MAF Frequency table applies a hard limit on the maximum metered amount of air that will be taken account of in the fuelling calculation. Hitting this limit will result in the air/fuel mixture leaning out.

The map units of Pulses Per Stroke (PPS) can be derived from the MAF signal frequency (usually referred to as "Airflow Hz" in Evoscan) via the following formula:

$$PPS = \text{Airflow Hz} * 20 / \text{RPM}$$

In the absence of a wideband oxygen sensor, logging the MAF signal frequency or ECU Load (a value itself directly derived from the MAF signal frequency) can be used to determine whether the airflow limit is being applied.

With the "338" MAF used in all J, L and W series cars, it has been found that ECU Load is ~20 times the PPS value for 3.5/ ROMs and ~23.6 times the PPS value for 3.0/ ROMs (changing the Raw Airflow to Engine Load Conversion Factor scalar will change these multipliers proportionally).

If the PPS as logged (calculated from either MAF signal frequency or ECU Load via the above identities) vs RPM curve coincides with the PPS vs RPM curve from the Max Allowed MAF Frequency table when plotted on the same graph, it is highly likely that the limit is being applied with consequential mixture enleanment. In this event the Max Allowed MAF Frequency table values need to be adjusted upwards until logging data shows a clear margin between the limit and the maximum metered airflow.

The maximum value that can be entered into a cell in the Max Allowed MAF Frequency table is 13.06 PPS. Hence with stock Raw Airflow to Engine Load Conversion Factor scalar values the practical limit for ECU Load is ~261 for a 3.5/ ROM and ~306 for a 3.0/ ROM. If airflow is to exceed the value limit of this table, a code patch to circumvent it's operation would be required.

Unfortunately it has become apparent that all production L/W series ROMs are afflicted with an effective load limit of around 95-97 (for the standard Raw Airflow to Engine Load Conversion Factor) unrelated to the Max Allowed MAF Frequency table. This load value will be exceeded with forced induction and can potentially be exceeded by a naturally aspirated 6G75. As yet the cause of this limit has not been identified and no neutralisation method is available. I am aware of two R&D ROMs in circulation which appear not to suffer from this particular limit: 82000001 (4 speed auto) and 85490000 (AWD).

8.2 Ignition advance limits

The minimum ignition advance (maximum ignition retard) supported by the code is 20° ATDC (-20° as a scaled value in tables). This cannot be changed without extensive code modification.

If TCL is enabled, the minimum limit is interpolated from the Ignition Advance – Minimum Limit – TCL Enabled table based on coolant temperature otherwise the Ignition Advance – Minimum Limit – TCL Disabled scalar value is used (usually 10° ATDC). While the L/W series manual and AWD ROMs include the Ignition Advance – Minimum Limit – TCL Enabled table, it won't be interpolated if TCL is enabled. This table is therefore not included in the definitions for those ROMs.

Although the scaling used for ignition advance would appear to support larger values, there appear to be calculations performed that would result in unpredictable—and potentially damaging—ignition timing being used once the advance limit (set via the Ignition Advance – Maximum Limit scalar value) is raised beyond 60° BTDC. *Caveat utilitor!*

8.3 ROMs with 3 Ignition Advance tables

Some very early J series ROMs, including both the 3.0/ manual and auto ROMs and several 3.5/ manual ROMs, have 3 ignition advance tables. While it hasn't been confirmed definitively, early investigations suggest that these ROMs use a similar ignition advance table layout to many of the Evo 9 ROMs which didn't enter service until about 5 years later.

In this scheme, table #2 is used during normal operation. During warmup coolant temperature is used to interpolate between tables #1 and #2. Table #3 appears to be used when limp mode fueling is active – either during the transition from cranking to running or when an operational malfunction is detected (such as a malfunctioning MAF).

8.4 Post-cranking and warm-up enrichment

During the starting process, once the ECU decides the engine is running the fuel management transitions from the cranking tables to the Limp Mode fuel mapping until the ECU can determine that the MAF is providing usable airflow metering. Once the MAF is determined to be functioning, the fuel management is then transitioned to the normal High/Low Octane fuel mapping.

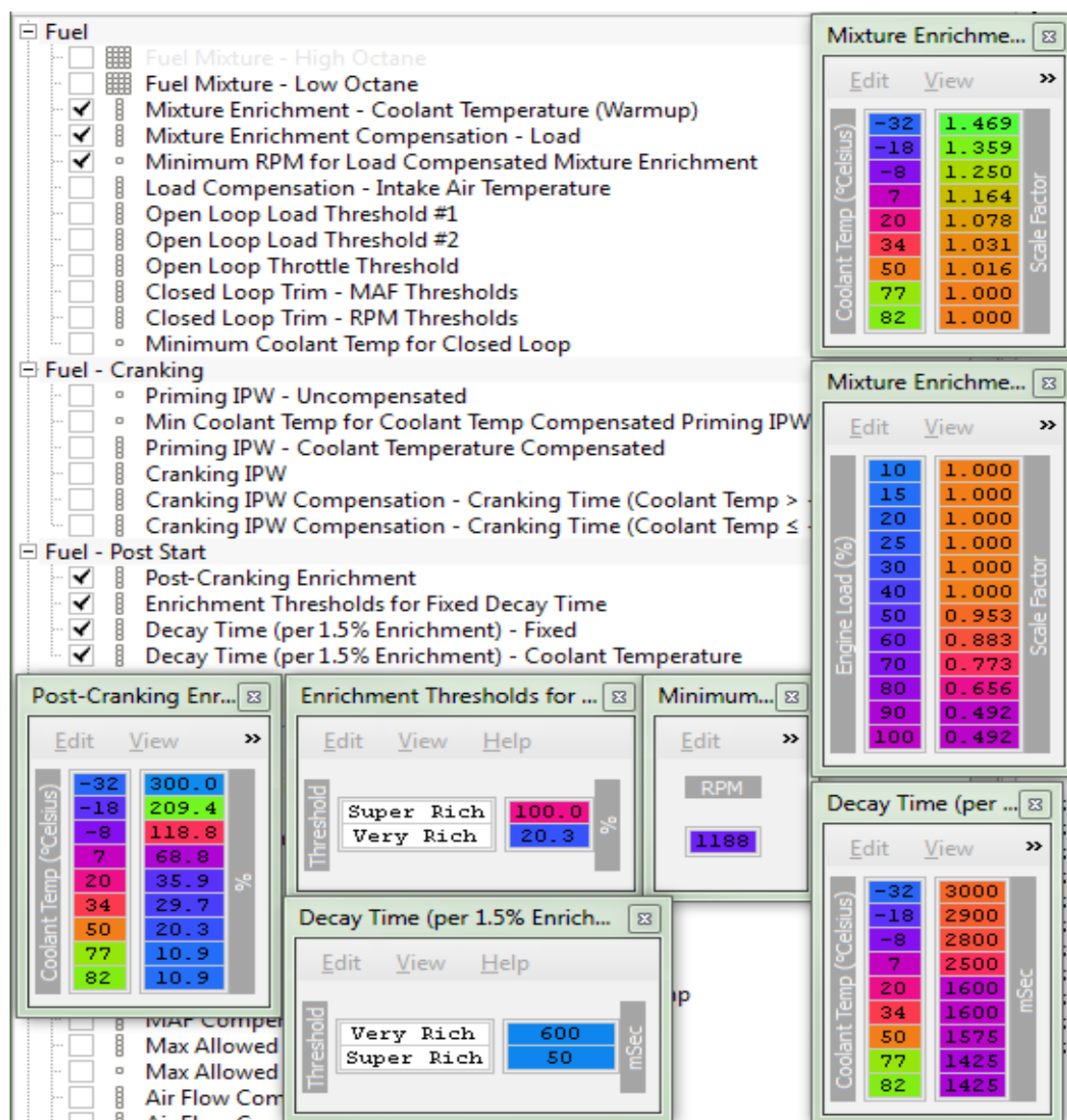


Figure 8: 6G74 post-cranking and warm-up enrichment tables

From the point that the engine is deemed to be running, there are two sets of enrichment applied

concurrently:

1. post-cranking enrichment; and
2. warm-up enrichment.

The post-cranking and warm-up enrichment tables for a stock 3.5l ROM are shown in Figure 8. Post-cranking enrichment is normally only of fairly short duration, though lower coolant temperatures naturally bring more enrichment and the enrichment lasts longer, whereas warm-up enrichment is essentially directly dependent on coolant temperature.

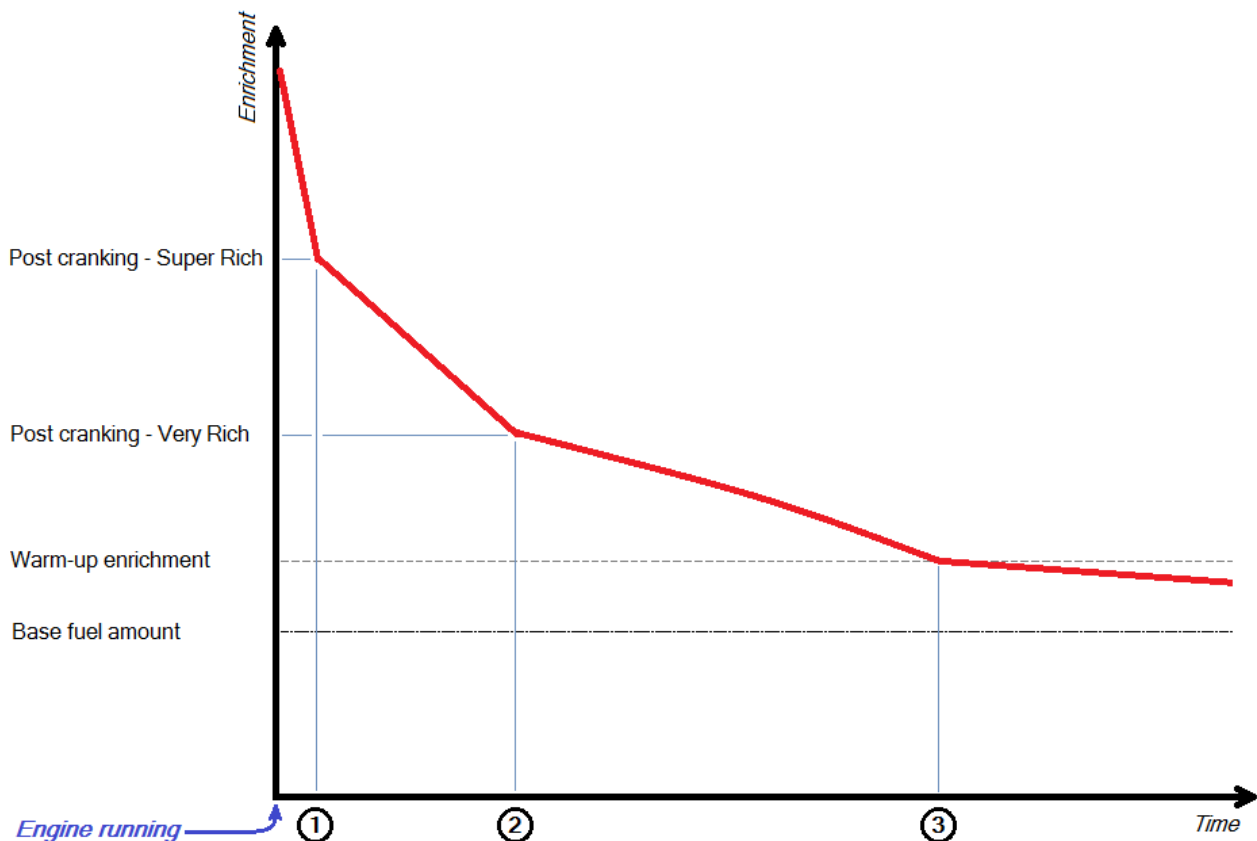


Figure 9: Enrichment vs time - post cranking and warm-up

Figure 9 is an attempt to illustrate the application of both post-cranking and warm-up enrichment commencing at the point in time that the engine is determined to be running.

The base fuel amount is increased by the warm-up enrichment factor interpolated from the Mixture Enrichment – Coolant Temperature (Warmup) table based on the coolant temperature. The amount of warm-up enrichment is adjusted as coolant temperature rises.

The post-cranking enrichment factor is applied to the combined base fuel amount and warm-up enrichment amount. The initial amount of post-cranking enrichment is interpolated from the Post-Cranking Enrichment table based on coolant temperature.

The points marked on the Time axis represent the following states:

1. post-cranking enrichment drops below the "Super Rich" threshold;
2. post-cranking enrichment drops below the "Very Rich" threshold;
3. post-cranking enrichment no longer active.

The post-cranking enrichment is reduced by ~1.5% per decay time unit. For enrichment above the "Very Rich" threshold, the duration of each decay time unit is set via the Decay Time (per 1.5% Enrichment) – Fixed table for the respective threshold. For enrichment below the "Very Rich" threshold, the duration of each decay time unit is interpolated from the Decay Time (per 1.5%

Enrichment) – Coolant Temperature table. The "Super Rich" and "Very Rich" enrichment thresholds are set via the Enrichment Thresholds for Fixed Decay Time table.

Once the post-cranking enrichment has been reduced to 0, the warm-up enrichment continues to be applied by interpolation from the Mixture Enrichment – Coolant Temperature (Warmup) table. When the engine speed is above the Minimum RPM for Load Compensated Mixture Enrichment value, the amount of warm-up enrichment is multiplied by the value interpolated from the Mixture Enrichment Compensation – Load table so that at higher loads the amount of warm-up enrichment is reduced.

8.5 Post-Cranking ISCV controls

In a similar manner to the post-cranking enrichment described in section 8.4, once the engine is deemed to be running the post-cranking ISCV demand interpolated from the ISCV Demand – Post-Cranking Adder table is added to the base ISCV demand interpolated from one of the Initial ISCV Demand tables. This extra ISCV demand then decays at a rate of 1 demand unit (approximately 0.4%) for each time period interpolated from the Idle-UP Demand Decay Time (per demand unit) After Startup table. The Idle Error Demand Adjustment tables don't appear to become effective until after the post-cranking demand component has decayed to 0.

8.6 Acceleration Enrichment and Deceleration Enleanment

Identification of the various scalars and tables for the acceleration enrichment and deceleration enleanment mapping is mostly derived from the work of EvolutionM.Net forum member MrFred, who extensively documented the operation of this functionality in Evo ROMs in his Advanced Fuel Control Options thread at <http://www.evolutionm.net/forums/ecuflash/453964-advanced-fuel-control-options.html> which should be read in order to understand the use of this information.

Note that the Magna/Verada ROMs don't contain all the items that are present in the Evo ROMs MrFred's description is based on. In particular, the Sync Load Change Idle related maps and code functionality are missing as far as I can determine. Additionally the Ceiling Load vs RPM for SyncLoadAccel/Decel Contributions 2D table present in Evo ROMs has been replaced by a scalar value in the Magna/Verada ROMs.

I have used raw airflow scalings rather than uncompensated load as used by MrFred to avoid complications with different conversion factors between ROMs. The renaming of the MAF Size scalar to Raw Airflow to Engine Load Conversion Factor was in part driven by the need to provide a reasonably straightforward method for users to visualise these values in load terms in the absence of any support in ECUFlash for scalings based on values in a ROM. Simply multiply a raw airflow value by the ROM's current Raw Airflow to Engine Load Conversion Factor to get the uncompensated load equivalent.

8.7 Strategies for E85 fuel management

I am aware of 3 basic strategies for configuring Mitsubishi ECUs to use E85 fuel:

1. Increase the Fuel Composition Flow Rate Compensation value for the active Spec Number; or
2. Reduce the Injector Size value; or
3. Reduce AFR values in the Fuel Mixture table(s).

Method 1 is the approach that Mitsubishi built into the ECU code for this purpose. This approach has the finest control resolution because of the way values are encoded and way the fuel calculations work. It is most easily understood when the fuel maps are set to use λ^3 scaling rather than AFR scaling, as Mitsubishi internally appears to consider fuelling in terms of λ rather than AFR.

Method 2 is probably the most widely used and documented method, particularly in the Evo

³ Air–fuel equivalence ratio, where $\lambda = 1.0$ is at stoichiometry, rich mixtures $\lambda < 1.0$, and lean mixtures $\lambda > 1.0$.

community.

Methods 1 and 2 are in fact different variables in the same fuel calculation, though in practice Method 1 provides for somewhat finer adjustment resolution than Method 2.

Method 3 is probably the approach self tuners using readily available consumer grade wideband oxygen sensor systems would think should be most easily understood, as it would appear that readings from an E85 calibrated AFR meter should be able to be used to directly adjust the fuel map. While this approach can be made to work reasonably well with sufficient care, in general it isn't as simple to set up as would first appear—for a variety of reasons—and suffers from potential loss of control resolution as the AFR values are reduced. While Method 3 would appear to be at odds with closed loop operation it works correctly because oxygen sensing works in terms of λ rather than AFR, provided that the AFR values entered in the parts of the map subject to closed loop operation accurately reflect the fuelling required to achieve stoichiometric mixtures.

When changing from petrol to E85 the fuel flow typically needs to be increased by around 30% in order to achieve satisfactory long term fuel trims during closed loop operation (i.e. at stoichiometric air/fuel mixtures).

Enabling and adjusting the Priming IPW – Coolant Temperature Compensated table, by lowering the Min Coolant Temp for Coolant Temp Compensated Priming IPW value to -40° C, may also prove beneficial when using E85 as the priming requirements can be more temperature sensitive than with petrol.

9 Glossary

ADC -

Analogue to Digital Converter

BEM -

Body Electronics Module

CEL -

Check Engine Light

ECU -

Engine Control Unit

ETACS -

Electronic Time and Alarm Control System

ROM -

Read Only Memory (also refers to microcontroller flash memory)

RPM -

Revolutions Per Minute

TCL -

Traction (Slip and Trace) Control system

TCU -

Transmission Control Unit

10 Changelog

20170717

- renamed the MAF Size scalar to Raw Airflow to Engine Load Conversion Factor and adjusted its scaling to reflect its actual use (which is to convert raw MAF signal values to Engine Load values)
- additional maps and scalars defined, including:
 - synchronous load acceleration enrichment and deceleration enleanment
 - asynchronous TPS based acceleration enrichment
 - coolant and intake air temperature thresholds for LTFT updates
- some corrections to map definitions and refinement of scalings
- addition of section 6.5: Logging TCU parameters
- addition of section 8.6: Acceleration Enrichment and Deceleration Enleanment
- minor editorial revisions to other text

20151230

- additional ROMs, both H8 and SH2, supported
- additional maps defined, including:
 - ROM identification codes
 - post cranking enrichment
 - ignition advance limits
 - injector flow linearisation (at short pulse durations)
 - post start idle increase decay timing
 - sensor calibration
 - barometric pressure sensor CEL control
- corrections and refinements to various scalings, including adopting MrFred's (of the EvolutionM.net forums) ISCV Demand scaling for the ISCV tables
- reorganised the grouping of information in definitions
- conversion of the README text file into a formatted PDF, with additional information about cars supported, logging, common configuration changes and some notes on map operation

20140722

- restructure use of include files for auto trans maps
- rationalise map naming so Diamante (which has Hi/Low Octane fuel/ignition maps) uses the same base definitions as the Magna/Verada (which only has Low Octane fuel/ignition maps)
- considerably expand the map coverage, including:
 - EGR solenoid duty cycle (Diamante)
 - cranking related maps
 - ISCV control related maps
 - MAF scaling related maps
- expand H8 definitions and integrate with base definition
- add several additional Diamante definitions

20111127

- fix reversed labels in the fan speed threshold tables
- bit 3 in periphery set 0/FAA: re-label to reflect function
- add EGR related maps (Magna/Verada only):
 - EGR solenoid duty cycle (category: Emissions)

- EGR enabled ignition advance (category: Timing)
- add auto mode shift point maps for all SH2 auto ROMs

20111012

- add definitions for 2001 USDM Diamante and 2002 Ralliart Magna (auto)

20110619

- add several definitions released individually
- add Periphery support
- add Idle speed Neutral/Drive maps

20110508

- initial release

11 Credits

There are many people to thank for contributions towards the information in this package, not least the creators of the OpenPort cables and the ECUFlash and Evoscan software.

I owe a lot to the efforts of quite a few people on the EvolutionM.net forums, especially Tephra, MrFred, acamus, logic (who wrote the disassembler I use, amongst other things), merlin.oz and mattjin. The information in sections 8.4 and 8.5 is an adaption and condensation of information presented in the thread at <http://www.evolutionm.net/forums/ecuflash/644259-start-up-fuel-iscv-tables.html>. Section 8.6 and the related tables and scalars in definitions were made possible by MrFred's Advanced Fuel Control options thread at <http://www.evolutionm.net/forums/ecuflash/453964-advanced-fuel-control-options.html>.

The GeekMapped forums have also been a useful source of information, however this site sadly seems to have disappeared.

There are a number of people whose knowledge, in one form or another, has been invaluable including A Graham Bell (book & other info), Matt Leicher (Hitman Tuning, Penrith, NSW), DaveTJ and Merlin.

I would particularly like to thank Merlin for supplying information about the cranking injector pulse width tables and EGR operation. Merlin's Evo Tuning Guide is a useful resource particularly regarding the process of working with Mitsubishi cars with ECUs of this type, even though some of the Evo specific details aren't applicable to the Magna/Verada.

Then there are the contributors of ROMs: TT_TJ, TUFFTR, Life, TJTime, ih8hsv, Dingers, fat35l, elvio, dsp26, bellto, dayno, rush, d1ng0d4n and others who remain anonymous.

Last but definitely not least, thanks to the admins of the ModifiedMitsubishi forums for permitting these files to be hosted on the forums and to the admins of the Aussie Magna Club forums for permitting a thread linking to separately hosted copies of these files.

12 Contact

I can be contacted as user WytWun on either the Modified Mitsubishi forums or the Aussie Magna Club forums.

Document last updated 17 July 2017.