

## How to build an EEC Definition file - Part 2 (Finding Parameters)

By Jaysen Anderson & Jason Bolger

There are a couple of ways to go about building a definition file to suit your desired EEC. This document attempts to cover the steps required in actually editing a bin.

If you do not understand the relationship between binary, hexadecimal and decimal, then we strongly recommend that you look here <http://tunerpro.markmansur.com/tutorials/UnderstandingHex.htm>. Hex really isn't hard, and as its short hand for binary, you don't really need to be fluent in binary, just have an understanding of how it all is inter-related. To summarise it: Binary is base 2, Decimal is base 10, and Hexadecimal is base 16. There is so much info on the net about this, so we need not go in to much detail.

So now you know what is needed to read your EEC and program a chip with your edited parameters. Next is the software, which to run the Aussie reader/burner will be TunerPro RT, or the Burn1 and associated components with Moates Flash & Burn. Its pretty easy to do, so we need not go in to much detail, however one thing to note is that you should set the start address to 0 and the end address to DFFF in an EECIV or 3FFFF in an EECV so that you read the whole bin. You can put your start address at 0x22000 in the EECV, but you will lose your EEC definition and bin code, and to run the xdf files that we have outlined above, you will need to repad the bin with the buffer you removed, so in summary, just start at 0.

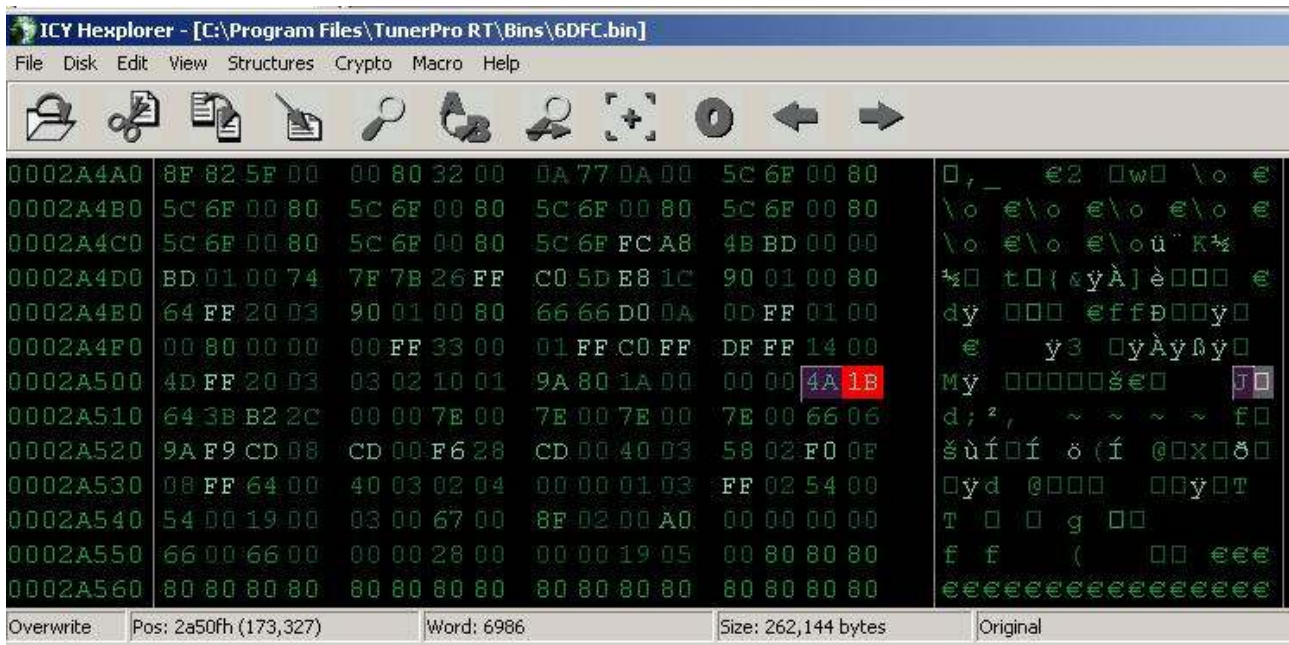
Something else to add when dumping EEC bins is that you should have the ECU powered, but not in the operational mode, ie have it installed in the car but with the key in the off position. There are several 12V+ & Ground pins on the EEC connector, but you need to run power to the KAM 12V+ pin and the ground to the ECU ground (not the case ground). so check with your workshop manual or online to find which ones are the right ones if you choose to power up the ECU on the bench.

Next you need a GUI (graphical user interface) to tune parameters. Yes we know you can change things with a hex reader, but trust us, it gets old very quick. This is where TunerPro comes in, there are several others out there, however this is our preferred GUI. Tunerpro has very detailed instructions under the help menu, so again we need not go in to much detail. If you have already got your bin defined, you need not read any further, as you will already be able to read, manipulate and burn EEC binary to the necessary hardware. If you want to chase down other parameters to define for your EEC, or you need to convert a similar bin def file to the bin you plan on using, here is how you can do it.

On the next page is a screen shot of TunerPro. The Constant circled in red is the Engine Displacement, you can simply scroll down the lists at the side and choose any of these constants & scalars you wish to change. Below that is Flags and Switches which dictate operating conditions that occur (its not hard to figure out), and below that again is Maps & Tables. The information circled in yellow is where it gets important. The info there tells you that the selected item is a constant, that its hex address in the bin file is at 901A, and that the size of the constant is 16 Bit, or a word in programming lingo. This bin file is an A9L Mustang ECU and its def file is a GUFB.

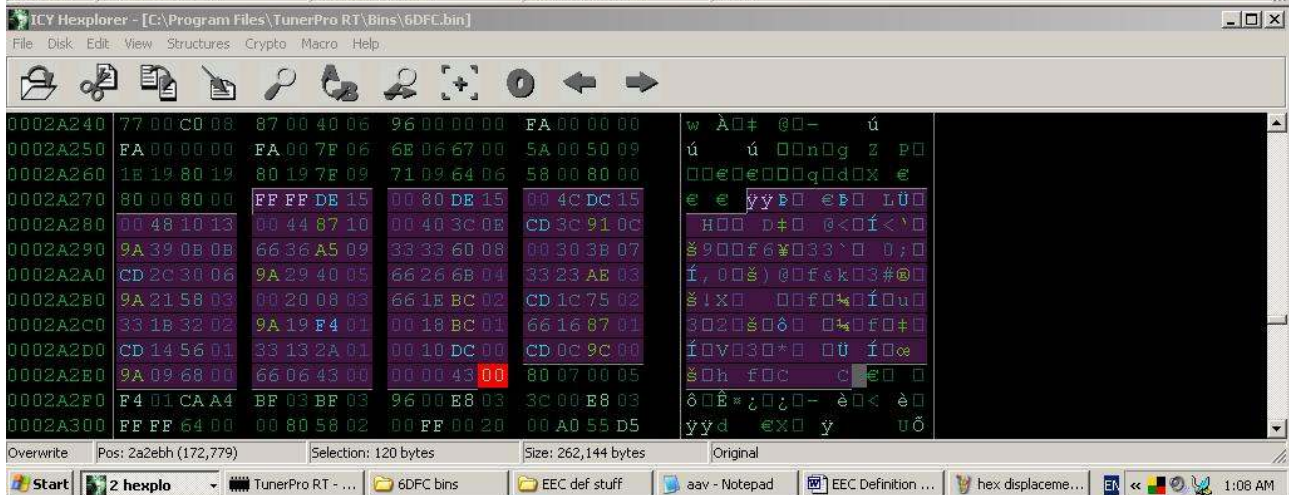
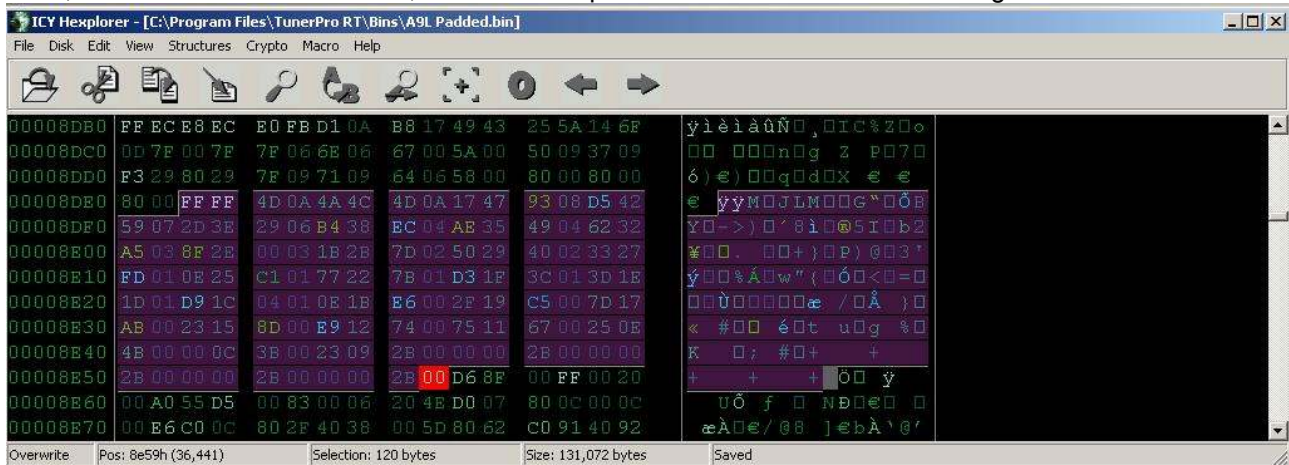
Now you open up 2 applications of your hex editor, load the defined bin in to one of them, and the bin that your searching for date in the other. A couple of things to note, hex editors normally have the offset (hex location) down the left hand side, if it doesn't, we would advise against using it. Sometimes the editor will also give you the decimal equivalent to the highlighted data, in the screen shot on the following page for instance you can see that 901A & 901B are both highlighted, and down the bottom of the screen you can see "Word: 6986", this means you have selected a word in length (2 bytes) and that is the decimal equivalent with the LSB (Least significant byte) first (numbers are read backwards from left to right, reverse to how you would normally write it). If your editor doesn't have a search function, and a jump to offset function, its going to make life harder for you, and if you can, sort your columns in to 4 banks of 4 like it is here, it makes it easier to read.





This parameter was simply found by highlighting a hex string (group of bytes) copying it and pasting it in to the find function of the hex editor. The key here is to get a string around a double word long (4 bytes) including no bytes with a value of zero (for some reason the editor will drop all data after the zero). In this example one would start with the string from 0x2A4F8 to 2A4FB, reading 01,FF,C0,FF, and chances are that you will get it nearly first time every time, but if at first you don't succeed, pick a different string and try again.

Once you have found the location, you can enter it in to the GUI, for TunerPro, you simply select the parameter, press F2 and you can then modify the xdf item. Put in the new location, apply and close the xdf editor window. Sometimes it can get tricky, for example the MAF Transfer table uses different calculations due to processor speeds, and as the A9L had the 8061, and the 6DFC had the 8065, the table multiplier is different. Back to the hex again.



Notice that both transfers start with FF,FF, that's because both have the highest voltage set to 16V, and see how the second word in on the A9L is 4D,0A, where as the second word on the 6DFC is DE,15. LSB applies here also, which means that the second word on the A9L would equal 2637, and the 6DFC would be 5598 in decimal.

Note how the transfer table is laid out, the 120 bytes is separated in to words, and each word alternates what it means. The first word is FF,FF, and when divided by 4096 it equals to 16, and to give it meaning to humans, we put a V next to that to mean that is what the MAF output in volts is. The next byte is the Mass Air Flow in KG/Hr's, so in the A9L, if you divide 2637 by 3.1562283 you will get a number for the MAF of 835.49KG/Hr. So now we know that when the MAF voltage is 16, the MAF is 835.5KG/HR, and it keeps going until it reaches zero (negative in some parameters) If you get a new MAF meter, it is advised to adjust your MAF transfer to suit, as your MAF is one of the main factors to determine load calculations. Normally reputable companies supply you with a chart of MAF vs Voltage, which will help you enter in the required numbers.

## ***Misc other stuff***

Some other stuff of interest is that the checksum is usually located @ the 10<sup>th</sup> byte after the program code starts, ie 200A in the EECIV and 2200A in the EECV 1 & 2 bank bins. The bins are usually laid out in banks, for instance the A9L is a 2 bank bin, as is the 6DFC, however the 4DDG is a 1 bank. What this means is that there is usually a bank of null data at the beginning of the bin, then the code starts. If its a single bank, there will be a very short burst of null data beteen the operating code and the parameters, where as if its a multiple bank bin, then there will be a substantial amount of null data between the 2. In the later EECV's, ie AU falcons, they are 4 bank ECU's.

There is a linked list which can be found @ 0x2020 in the A9L and @ 22060 in the 6DFC. This should give you the pattern to find it in other bins. In the A9L the word is the actual jump to address of the linked list is 0x8c00, where as in the EECV 256k 6DFC, the 2 needs to be added to the front of the list due to the larger offset. The list is LSB, so in the A9L the first jump to is, and in the 6DFC its 0x2A000. When you jump to that location you will find that it has the next jump to address found in the linked list. We havnt discovered what to do with this yet, apart from the fact that the parameters start at the first jump to address, but if you can figure out the rest of them, we would appreciate it if you could share it with the rest of us.

More will be added to this document as we discover or figure out how to explain it in an easy manner.

JA & JB