ME 7.x with Launch-Control & No-Lift-Shift

This "document" is written by someone with allmost zero knowledge in assembler and not very good insight into the ME 7.x, or rather the infineon/siemens c167 cpu and its memory setup itself..
The document is composed from my own personal notes and english is not my native language..
so try to bear with me here..

My main source of information and inspiration has been the NefMoto community, these guys are all about free information and helping eachother out with anything and everything related to tuning =)

Click the logo to visit this awesome community!

..anyway, this is what i came up with after a few days/nights of tinkering.. // **Sn00k**

Be WARNED! this function can SERIOUSLY damage your engine and exhaust components!

Be adviced! you need a coildriver of the later model to use this function, otherwise youll burn the ignition coils, or if youre lucky, just a fuse.
(this seem to work perfectly on 1.8t models year 01 and up using the 1024kb flash, also on the S4 M-box, i have yet to try it out on other cars.)

This manipulation is done in binary/hex form, and based on setzis v2 function posted in this thread:

http://nefariousmotorsports.com/forum/index.php?topic=607.msg5283#msg5283

Oh, be sure to grab his zip-file as it contains an xdf where conditions are explained.

This described is the "easy"/lazy way to implement the Launch-control & No-Lift-Shift functions into me 7.x, it is **VERY** simplified, and leaves out alot of steps needed to write and compile the actual function itself.. let alone modify it..

All credit goes to **setzi62** for writing and compiling this function.. originally i think the idea for a similar function, which also manipulates ignitioncoil dwelltime, came from Eurodyne(?).

To ensure this working 100% and not "bricking" the ecu, a full dissassembly of the image should be done, and ALL adresses that will be used to store the new function and variables should be crossreferenced to make sure them are free to use and dont interfere with the normal operations of the ecu.

..if you skip this step, and yes you can do that, be sure to have access to **bootmode** flashing, as you will really be doing some educated guessing from here on.. ;)

1. Start this adventure by finding yourself a good **HEX-editor**, i personally like the "010-editor", but there are many good ones out there.

   Some friendly advice before we begin:

   BE SURE YOU DONT INSERT VALUES AND MOVE THE CODE, MAKE SURE TO WRITE THEM OVER. THIS ALSO GOES FOR PASTING CODE, MAKE SURE IT DONT MOVE THE CODE AROUND, OR AFFECTS THE LENGTH OF THE FILE.

   (in 010-editor you had to pre-mark the field in which u wanted to paste, else it moved everything around, and this corrupts the file beyond rescue)

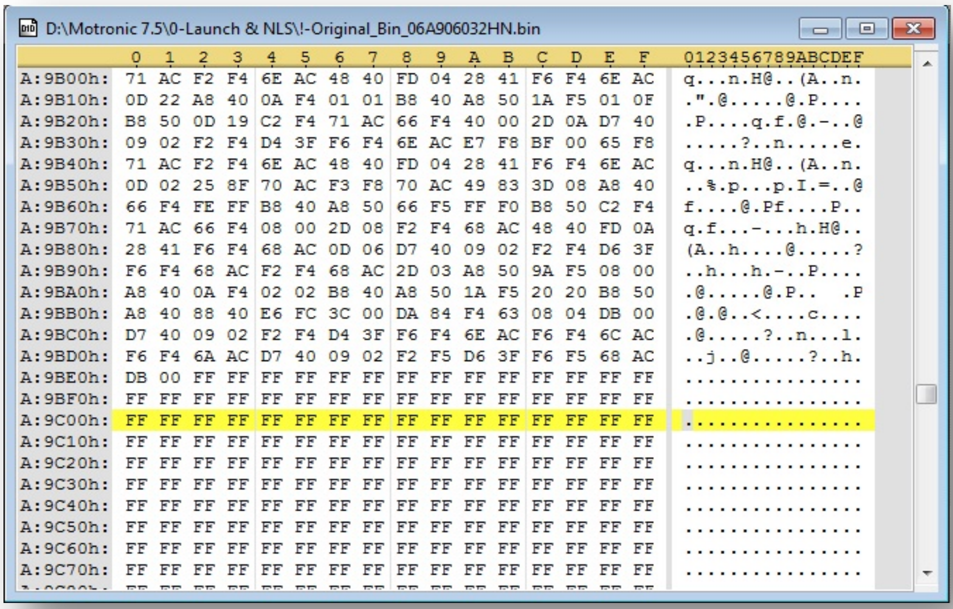
2. Download and use setzis **me7logger** tool, called "**me7info**" with these parameters:
   "me7info -n file.bin" to create an **.ecu** file containing specific adresses for **your** ecu, sorted by adress, as you will need these later on.
   (this .ecu file is placed in the "ecu" directory.)

   the logger and tool can be found here:
   http://nefariousmotorsports.com/forum/index.php?topic=837.msg7054#msg7054

3. Open your .bin file in the hex-editor.. and also an empty txt file, as you will be taking lots of notes of **adresses** during this.

4. First find some empty space after the OEM code, where you are to put the new function. search 0x75000h onwards for an empty line consisting of pure "**FF**" till you find a good even number, like: **8E800** that setzi used.
   The function currently occupies 9 rows of space.

   see screenshot for an example:



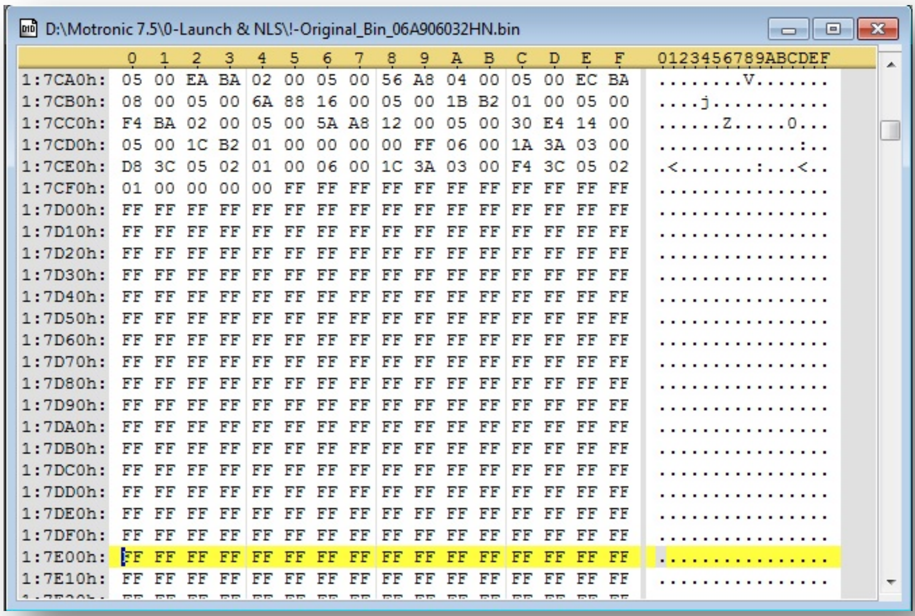
D:\Motronic 7.5\0-Launch & NLS\!-Original_Bin_06A906032HN.bin

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
A:9B00h:  71 AC F2 F4 6E AC 48 40 FD 04 28 41 F6 F4 6E AC   q...n.H@..(A..n.
A:9B10h:  0D 22 A8 40 0A F4 01 01 B8 40 A8 50 1A F5 01 0F   .".@.....@.P....
A:9B20h:  B8 50 0D 19 C2 F4 71 AC 66 F4 40 00 2D 0A D7 40   .P....q.f.@.-..@
A:9B30h:  09 02 F2 F4 D4 3F F6 F4 6E AC E7 F8 BF 00 65 F8   .....?..n.....e.
A:9B40h:  71 AC F2 F4 6E AC 48 40 FD 04 28 41 F6 F4 6E AC   q...n.H@..(A..n.
A:9B50h:  0D 02 25 8F 70 AC F3 F8 70 AC 49 83 3D 08 A8 40   ..%.p...p.I.=..@
A:9B60h:  66 F4 FE FF B8 40 A8 50 66 F5 FF F0 B8 50 C2 F4   f....@.Pf....P..
A:9B70h:  71 AC 66 F4 08 00 2D 08 F2 F4 68 AC 48 40 FD 0A   q.f...-...h.H@..
A:9B80h:  28 41 F6 F4 68 AC 0D 06 D7 40 09 02 F2 F4 D6 3F   (A..h....@.....?
A:9B90h:  F6 F4 68 AC F2 F4 68 AC 2D 03 A8 50 9A F5 08 00   ..h...h.-..P....
A:9BA0h:  A8 40 0A F4 02 02 B8 40 A8 50 1A F5 20 20 B8 50   .@.....@.P..  .P
A:9BB0h:  A8 40 88 40 E6 FC 3C 00 DA 84 F4 63 08 04 DB 00   .@.@..<....c....
A:9BC0h:  D7 40 09 02 F2 F4 D4 3F F6 F4 6E AC F6 F4 6C AC   .@.....?..n...l.
A:9BD0h:  F6 F4 6A AC D7 40 09 02 F2 F5 D6 3F F6 F5 68 AC   ..j..@.....?..h.
A:9BE0h:  DB 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9BF0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C00h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C10h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C20h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C30h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C40h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C50h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C60h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
A:9C70h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

   write down the adress.

5. Next find some empty space where to put "condition"-variables, around 0x**17E00** > (search onwards for an empty line consisting of pure "**FF**", pref on a number like "00" to keep things clean and easy to remember, setzi originally used: **17E00**)

   see screenshot:

```
D:\Motronic 7.5\0-Launch & NLS\!-Original_Bin_06A906032HN.bin

           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
1:7CA0h:  05 00 EA BA 02 00 05 00 56 A8 04 00 05 00 EC BA   ........V.......
1:7CB0h:  08 00 05 00 6A 88 16 00 05 00 1B B2 01 00 05 00   ....j...........
1:7CC0h:  F4 BA 02 00 05 00 5A A8 12 00 05 00 30 E4 14 00   ......Z.....0...
1:7CD0h:  05 00 1C B2 01 00 00 00 00 FF 06 00 1A 3A 03 00   ..............:..
1:7CE0h:  D8 3C 05 02 01 00 06 00 1C 3A 03 00 F4 3C 05 02   .<.......:...<..
1:7CF0h:  01 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D00h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D10h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D20h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D30h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D40h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D50h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D60h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D70h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D80h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7D90h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DA0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DB0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DC0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DD0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DE0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7DF0h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7E00h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
1:7E10h:  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

   write down the adress.

6. Take a look at setzis **pseudo code,** which i have attached to this document, to better get a grip on how the function works:
   setzi_v2_pseudo_code.txt

7. Fetch Setzis **compiled function** for the s4 M-box:
   setzi_v2_compiled.txt

8. You will also need **the function dissassembled** to understand how it is built:
   setzi_v2_dissassembled.txt

9. You will need to know all the **M-box adresses** to understand the code later on, these are normally fetched in dissassembly, but ill supply them for reference here:

   M-box_adresses.txt

10. Now you need to go into your created **.ecu file** with notepad or similar text-editor and fetch your adresses.. **you need the same items as in the "M-box_adresses.txt"** file, be sure to get them all, and be sure to write down the "**Bitmask**" after the values too, esp on "B_kuppl" and "B_brems".

    now the B_brems eluded me for a while, since it is placed inside a **bit-registry** along with B_kuppl and some more variables, that said i found it located in the same adress as the B_kuppl, but **-2** in the bitmask.

    that is if B_kuppl is located at 0x00FD56**.8** (the last **.8** beeing the bitmask), then B_brems should be located at 0x00FD56**.6**

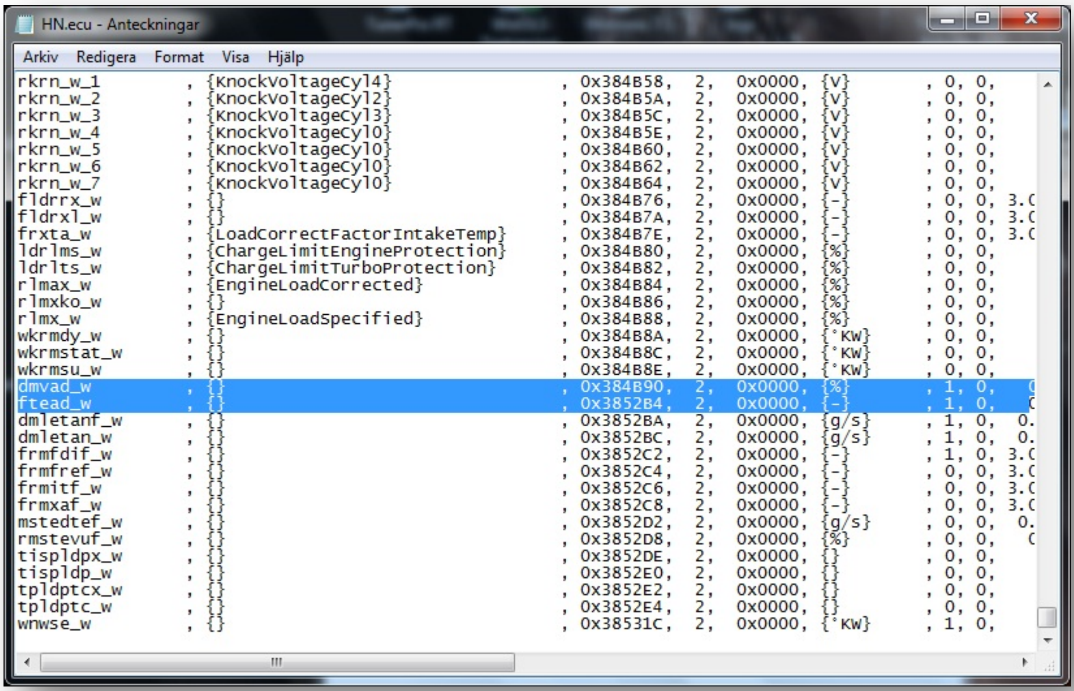
    (i have verified this to be the case in 4 different bins).
    thanks to Gremlin for pointing this out.

    more info on bit-registrys, bitmasks and how to convert them from say **0x0040** into **.6** in this attached file:

    Bit-registrys_bitmasks.txt

11. Locate emtpy **RAM adress** searching in your .ecu file, like 0x384**FF0** that setzi originally used.. find unused adress preferably after the last used 0x38**4**xxx adress.
    (this is where you will place the NLS-counter later on)

    See screenshot:



```
rkrn_w_1        , {KnockVoltageCyl4}                   , 0x384B58,  2,  0x0000, {V}       , 0, 0,
rkrn_w_2        , {KnockVoltageCyl2}                   , 0x384B5A,  2,  0x0000, {V}       , 0, 0,
rkrn_w_3        , {KnockVoltageCyl3}                   , 0x384B5C,  2,  0x0000, {V}       , 0, 0,
rkrn_w_4        , {KnockVoltageCyl0}                   , 0x384B5E,  2,  0x0000, {V}       , 0, 0,
rkrn_w_5        , {KnockVoltageCyl0}                   , 0x384B60,  2,  0x0000, {V}       , 0, 0,
rkrn_w_6        , {KnockVoltageCyl0}                   , 0x384B62,  2,  0x0000, {V}       , 0, 0,
rkrn_w_7        , {KnockVoltageCyl0}                   , 0x384B64,  2,  0x0000, {V}       , 0, 0,
fldrrx_w        , {}                                   , 0x384B76,  2,  0x0000, {-}       , 0, 0, 3.
fldrxl_w        , {}                                   , 0x384B7A,  2,  0x0000, {-}       , 0, 0, 3.
frxta_w         , {LoadCorrectFactorIntakeTemp}        , 0x384B7E,  2,  0x0000, {-}       , 0, 0, 3.
ldrlms_w        , {ChargeLimitEngineProtection}        , 0x384B80,  2,  0x0000, {%}       , 0, 0,
ldrlts_w        , {ChargeLimitTurboProtection}         , 0x384B82,  2,  0x0000, {%}       , 0, 0,
rlmax_w         , {EngineLoadCorrected}                , 0x384B84,  2,  0x0000, {%}       , 0, 0,
rlmxko_w        , {}                                   , 0x384B86,  2,  0x0000, {%}       , 0, 0,
rlmx_w          , {EngineLoadSpecified}                , 0x384B88,  2,  0x0000, {%}       , 0, 0,
wkrmdy_w        , {}                                   , 0x384B8A,  2,  0x0000, {°KW}     , 0, 0,
wkrmstat_w      , {}                                   , 0x384B8C,  2,  0x0000, {°KW}     , 0, 0,
wkrmsu_w        , {}                                   , 0x384B8E,  2,  0x0000, {°KW}     , 0, 0,
dmvad_w         , {}                                   , 0x384B90,  2,  0x0000, {%}       , 1, 0,
ftead_w         , {}                                   , 0x3852B4,  2,  0x0000, {-}       , 1, 0,
dmletanf_w      , {}                                   , 0x3852BA,  2,  0x0000, {g/s}     , 1, 0,  0.
dmletan_w       , {}                                   , 0x3852BC,  2,  0x0000, {g/s}     , 1, 0,  0.
frmfdif_w       , {}                                   , 0x3852C2,  2,  0x0000, {-}       , 1, 0, 3.
frmfref_w       , {}                                   , 0x3852C4,  2,  0x0000, {-}       , 0, 0, 3.
frmitf_w        , {}                                   , 0x3852C6,  2,  0x0000, {-}       , 0, 0, 3.
frmxaf_w        , {}                                   , 0x3852C8,  2,  0x0000, {-}       , 0, 0, 3.
mstedtef_w      , {}                                   , 0x3852D2,  2,  0x0000, {g/s}     , 0, 0,  0.
rmstevuf_w      , {}                                   , 0x3852D8,  2,  0x0000, {%}       , 0, 0,
tispldpx_w      , {}                                   , 0x3852DE,  2,  0x0000, {}        , 0, 0,
tispldp_w       , {}                                   , 0x3852E0,  2,  0x0000, {}        , 0, 0,
tpldptcx_w      , {}                                   , 0x3852E2,  2,  0x0000, {}        , 0, 0,
tpldptc_w       , {}                                   , 0x3852E4,  2,  0x0000, {}        , 0, 0,
wnwse_w         , {}                                   , 0x38531C,  2,  0x0000, {°KW}     , 1, 0,
```

write down the adress.

12. Now you have all your **adresses** needed?

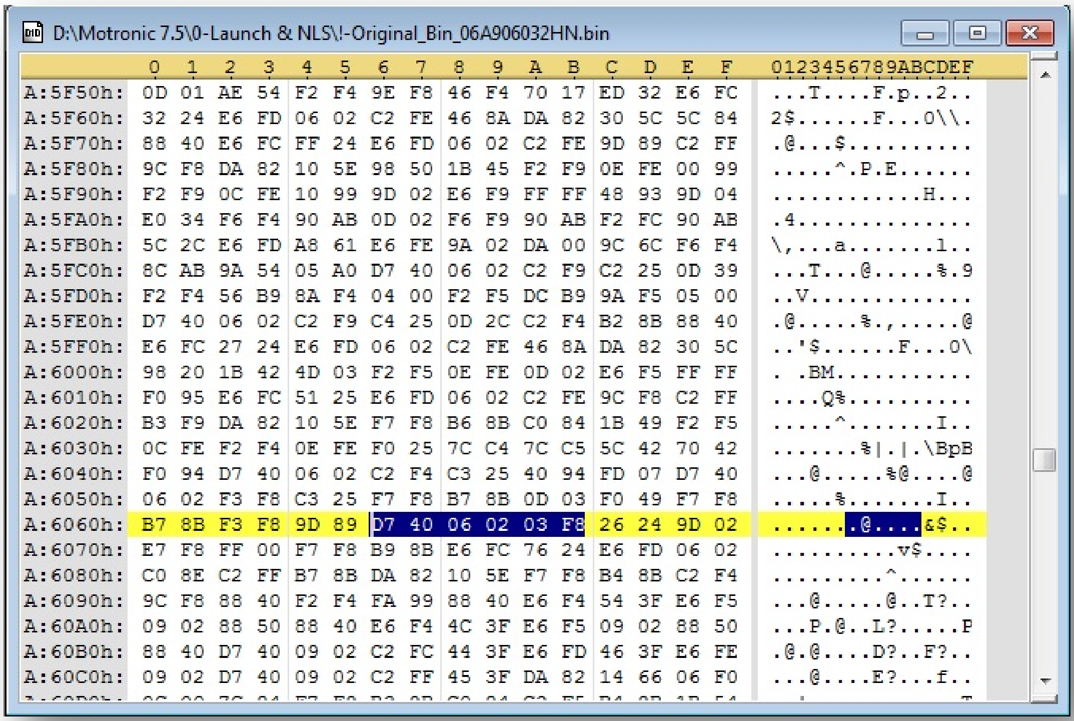
    if **yes**, now comes the tricky part, entering these directly into the allready assembled code. this took some tampering and tinkering to figure out, but it was not too hard after having compiled a "few" rows by themselves in a compiler.(read: 48hour **madness**)
    I also had alot of documentation for the c167 cpu, where i could look things up, especially how it uses adresses in rom/ram and asm commands, not to mention good advice from people here on the forum, getting me back on track after i was somewhat LOST, thanks guys!

    Go HERE for detailed info on **how to modify the function!**

13. Now to **locate** the correct **routine** and **call the new function**:

    in your hex editor, go to the end of file and search upwards for: "**D7 40 06 02 03 F8**" and you should be in the right place.

    See screenshot:



```
D:\Motronic 7.5\0-Launch & NLS\!-Original_Bin_06A906032HN.bin

           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
A:5F50h:  0D 01 AE 54 F2 F4 9E F8 46 F4 70 17 ED 32 E6 FC   ...T....F.p..2..
A:5F60h:  32 24 E6 FD 06 02 C2 FE 46 8A DA 82 30 5C 5C 84   2$......F...0\\.
A:5F70h:  88 40 E6 FC FF 24 E6 FD 06 02 C2 FE 9D 89 C2 FF   .@...$..........
A:5F80h:  9C F8 DA 82 10 5E 98 50 1B 45 F2 F9 0E FE 00 99   .....^.P.E......
A:5F90h:  F2 F9 0C FE 10 99 9D 02 E6 F9 FF FF 48 93 9D 04   ............H...
A:5FA0h:  E0 34 F6 F4 90 AB 0D 02 F6 F9 90 AB F2 FC 90 AB   .4..............
A:5FB0h:  5C 2C E6 FD A8 61 E6 FE 9A 02 DA 00 9C 6C F6 F4   \,...a.......l..
A:5FC0h:  8C AB 9A 54 05 A0 D7 40 06 02 C2 F9 C2 25 0D 39   ...T...@.....%.9
A:5FD0h:  F2 F4 56 B9 8A F4 04 00 F2 F5 DC B9 9A F5 05 00   ..V.............
A:5FE0h:  D7 40 06 02 C2 F9 C4 25 0D 2C C2 F4 B2 8B 88 40   .@.....%.,.....@
A:5FF0h:  E6 FC 27 24 E6 FD 06 02 C2 FE 46 8A DA 82 30 5C   ..'$......F...0\
A:6000h:  98 20 1B 42 4D 03 F2 F5 0E FE 0D 02 E6 F5 FF FF   . .BM...........
A:6010h:  F0 95 E6 FC 51 25 E6 FD 06 02 C2 FE 9C F8 C2 FF   ....Q%..........
A:6020h:  B3 F9 DA 82 10 5E F7 F8 B6 8B C0 84 1B 49 F2 F5   .....^.......I..
A:6030h:  0C FE F2 F4 0E FE F0 25 7C C4 7C C5 5C 42 70 42   .......%|.|.\BpB
A:6040h:  F0 94 D7 40 06 02 C2 F4 C3 25 40 94 FD 07 D7 40   ...@.....%@....@
A:6050h:  06 02 F3 F8 C3 25 F7 F8 B7 8B 0D 03 F0 49 F7 F8   .....%.......I..
A:6060h:  B7 8B F3 F8 9D 89 D7 40 06 02 03 F8 26 24 9D 02   .......@....&$..
A:6070h:  E7 F8 FF 00 F7 F8 B9 8B E6 FC 76 24 E6 FD 06 02   ..........v$....
A:6080h:  C0 8E C2 FF B7 8B DA 82 10 5E F7 F8 B4 8B C2 F4   .........^......
A:6090h:  9C F8 88 40 F2 F4 FA 99 88 40 E6 F4 54 3F E6 F5   ...@.....@..T?..
A:60A0h:  09 02 88 50 88 40 E6 F4 4C 3F E6 F5 09 02 88 50   ...P.@..L?.....P
A:60B0h:  88 40 D7 40 09 02 C2 FC 44 3F E6 FD 46 3F E6 FE   .@.@....D?..F?..
A:60C0h:  09 02 D7 40 09 02 C2 FF 45 3F DA 82 14 66 06 F0   ...@....E?...f..
```

what you want to do code-wise is:

replace: "*movb rl4, byte_xxxx*"

with: "***calls*** *aah, 0Ebbbh ; aaEbbbh*"

This is done by finding the "**F3**" value in this line(or the line above it)..
here is an example line:

"B7 8B **F3** F8 9D 89 D7 40 06 02 03 F8 26 24 9D 02"

this **F3** you will replace with "**DA**" (*calls*).
the next **three fields** will consist of **your adress where you placed the new function**.

now its getting tricky again since it is scrambled.. like this..

if i want it to call my **new function** located here: **8A9C00**
(allways starts with **8** since in rom)

this becomes in the next 3 fields: **8A 00 9C**

making **the new line** in my example look like this:

B7 8B **DA 8A 00 9C** D7 40 06 02 03 F8 26 24 9D 02

from how it **originally** looked, like this:

B7 8B **F3 F8 9D 89** D7 40 06 02 03 F8 26 24 9D 02

BE SURE YOU DONT INSERT VALUES AND MOVE THE CODE, MAKE SURE TO WRITE THEM OVER IN THE SAME LOCATION.

There, you have placed the call to your new function!

14. **Allmost done!** now we need to find **your conditions value** in the hex-editor, like setzi used **17E00** as an example, and insert **this** line to give it setzis standard example values:
(these can be changed later on in your XDF/ols)

"A6 01 50 46 0A 00 F0 55 E6 FF FF FF FF FF FF FF"

15. **Make your xdf/winols references** like this, using setzis **17E00** for example:
(enter your choosen adress for conditions instead)

SpeedThreshold:	0x17E00 size 2, 16bit LSB, conversion: X * 0.0078125	(km/h)
LaunchRPM:	0x17E02 size 2, 16bit LSB, conversion: X * 0.25	(rpm)
IgnitionCutDuration:	0x17E04 size 2, 16bit LSB, conversion: X * 20	(ms)
RPMThreshold:	0x17E06 size 2, 16bit LSB, conversion: X * 0.25	(rpm)
AccPedalThreshold:	0x17E08 size 1, 8bit LSB, conversion: X * 0.392157	(%)

FTOMN: find this adress in your damos/xdf, 8bit LSB, conversion: X / 10 (ms)

Take a look in setzis xdf for complete information on these conditions, or read the pseudo code again, things should be pretty clear by now.

16. Now you should have everything you need, and be able to add this functionality to basically any me7.x.

DONT FORGET to:

* change the **FTOMN** to "**0**" (0.00**ms**)
* **set your conditions**
* **calculate your checksums.**

Also be sure to let me know what you think of this write-up, as said, this was done with allmost no previous knowledge.. but.. i found some patterns.. and, after some trial and error, in the end i successfully compiled my own function for shooting fireballs out the "auspuff"..! =D

Visit the NefMoto community:

```
9A 2B 13 80 F2 F4 40 8E D7 00 81 00 F2 F9 00 7E
40 49 9D 0B F2 F4 7A F8 D7 00 81 00 F2 F9 02 7E
40 49 FD 03 F7 8E AC 8D 0D 2F 9A 2B 29 80 8A 2B
22 60 F2 F4 7A F8 D7 00 81 00 F2 F9 06 7E 40 49
FD 1A C2 F4 02 8B D7 00 81 00 C2 F9 08 7E 40 49
FD 12 D7 00 38 00 F2 F4 F0 4F D7 00 81 00 F2 F9
04 7E 40 49 9D 11 F7 8E AC 8D 08 41 D7 00 38 00
F7 F8 F0 4F 0D 09 D7 00 38 00 F6 8F F0 4F 0D 04
D7 00 38 00 F6 8E F0 4F F3 F8 F3 8A DB 00 FF FF
```

Ꮖ·ō2ǿŜ ƛă ăŜűΊƛă Ŏ2ΥǈƛƒŜŘ ŦdzyŎűƛ2yφ

Now **THIS** looks like a scrambled mess **at first**.. and it took me a while to figure out the patterns, but, ill try to brake it down into segments and explain as much as i can..
patching asm code is very hard.. and patching compiled code is.. even worse imo.. that said, **here goes nothing! =D**

9A <u>**2B**</u> 13 <u>**80**</u> F2 F4 40 8E D7 00 81 00 F2 F9 <u>**00 7E**</u>

Line 1, in this first row, we have in the end the **00 7E** adress, which really is the **17E00** location, that one is easy to **identify** and **modify**.

now looking in **the dissassembled function** we find that there are **two** adresses **BEFORE** this one.

(dont bother with the "loc_xxxxx" adresses in the dissassembled function as the code is position independent.)

these are **8E40**, which refers to your **vfil_w** located at **0x380E40**..
now these **0x38xxxx** adresses are remapped as 8000...BFFF, which basically means your **380E40** now is beeing called **8E40**.
see, this is the one in the dissassembled code, and it is located in bit **6-7** in the binary, and also in **reversed** order, **7-6**.

a 0x38**1E40** adress would be remapped as **9E40** (and written **"40 9E"**) since <u>**the adresses start at 8000**</u>.

now in this first line i ran into some trouble..
the **9A** in the first bit seem to suggest this is a **"jnb"** instruction, whereas if it had been an **8A**, it suggest it beeing a **"jb"**..
this isn't really important now, but we will run into this later on, just thought id mention it.

now you see that **80** value in bit **3**, the **first half of that one is the bitmask**.. so if your bitmask is **.8**, enter **80** here.. if it is **12**, enter **C0**, and so on.
(look your bitmask up in the **.ecu** file and compare it to the table i supplied on **registrys and bitmasks**)

the second bit is the tricky part, the **2B** (we cannot touch the bit holding 13 as this seem to be code), so we need to figure out how this simple **"2B"** value becomes the adress **FD56.**

documentation of the memory structure states:

*"The upper 256 Byte of the internal RAM (00'FD**00**h through 00'FD**FF**h) and the GPRs of the current bank are provided for single bit storage, and thus they are bit addressable."*

so this means the value we really need to focus on is **56**, and how this **2B** references it in the code.

these are **EVEN** bit values.. split them in 2, and the first and second part can be grabbed in the table below:

F	=	1E
E	=	1C
D	=	1A
C	=	18
B	=	16
A	=	14
9	=	12
8	=	10
7	=	E
6	=	C
5	=	A
4	=	8
3	=	6
2	=	4
1	=	2
0	=	0

if we are looking for say **FD56**, we would enter **2** for **4**(or **40** since it stands **infront** of another number),
and **B** for **16** here, making it **"2B"**(or **56**)

whilst if we were looking for **FD4E** we would enter **27** here..
FD82 would be **41**.. **FD5E** would be **2F** and so on.

i think this has to do with memory beeing accessed only at EVEN bits, but given the time is atm 04:30am atm and my brain-function = nonexistent..
..ill try not to give it any more thought.. i dissassembled this back and forth for about 2 hours and came up with this.. which is a working pattern, good enough for me.

thats it, if youve read and somewhat understood the explanations above, you should be able to enter your adresses accordingly into the first row **ABOVE** ^^.

40 49 9D 0B F2 F4 **7A F8** D7 00 81 00 F2 F9 **02 7E**

Line 2, in the end we find the **"02 7E"**, which = **7E02** (**17E02**), which is the next **"conditions"** value, easy to modify into your own value. looking into the dissassembled function we can see a **F87A** adress before this, and this is the **nmot_w** on adress **0x00F87A**, its located in bit **6-7** as **"7A F8"** which reversed is **F87A** and **0x00f87A**.. very straightforward to alter. were done with row 2.

40 49 FD 03 F7 8E **AC 8D** 0D 2F *9A* **2B** 29 **80** *8A* **2B**

UPDATE!! **Line 3**, first we have the **tsrldyn** located at bits **6-7**, as **"AC 8D"**, which reversed is **8DAC** as in **0x380DAC**, switch this for your own.

remember those **"jnb"**(**9A**) and **"jb"**(**8A**)functions **i mentioned** we would run into **earlier**, well here we have them.
you recognize the value after **9A**? yes, that is the value you calculated earlier, and the same **B_kuppl**,swap it for the same adress as you did earlier.. and **remember to also correct the bitmask** like earlier which is stored in that **80**.
now wait, there is an **8A** here.. followed by another **2B**.. this is the **B_brems** you are seeing, and as it is located **in the same registry**, swap this adress to the same as **B_kuppl**, BUT remember to enter the correct bitmask!(**see Line 4**)

22 **60** F2 F4 **7A F8** D7 00 81 00 F2 F9 **06 7E** 40 49

Line 4, if you have followed **the dissassembled function**, you se that **b_brems** is located in the same registry as **b_kuppl**, but at bitmask **.6** and not **.8** and so we begin with correcting the bitmask at bit **1** which is **60**, meaning **.6** atm.
in the end of the row we see another of the **17E0x** adresses.. the **17E06** in form of **"06 7E"**, alter it to your correct adress.
looking in **the dissassembled function** again we can see that there is an **F87A** before this, which was the **nmot_w** again at adress **0x00f87A**, located on bit **4-5** as **"7A F8"**, which reversed is **F87A.**
swap it for your own **nmot_w** adress.
row 4 done.

FD 1A C2 F4 **02 8B** D7 00 81 00 C2 F9 **08 7E** 40 49

Line 5, in the end of it we have the **17E0x** again, in form of **"08 7E"** which in reverse is **7E08**, swap it for your own.
looking in the code again, we have only one more adress in this line, called **8B02**, which translates to **0x380B02**, which would be the **wped**, we find it at bit **4-5** in form of **"02 8B"** which reversed is **8B02**, swap it for your own **wped.**
line 5 done.

FD 12 D7 00 38 00 F2 F4 **F0 4F** D7 00 81 00 F2 F9

Line 6, here we find the **NLScounter**, which setzi originally placed at **0x384FF0** in the **RAM**, remember? you should allready have found a **free adress** for this earlier looking into your **.ecu** file. so go ahead and alter this one located at bit **8-9**, in form of **"F0 4F"** which in reverse becomes **4FF0**.

UPDATE!! There is a pointer to the **RAM**, **0x38**0000,in the code here so disregard the **NLScounter** from the adress remapping, 0x38**4FF0** becomes **4FF0**.
line 6 done.

04 7E 40 49 9D 11 F7 8E **AC 8D** 08 41 D7 00 38 00

Line 7, this one starts with a known **17E0x**, in form of **"04 7E"** at bit **0-1**, which translates into **7E04** as in **17E04**, swap it for your own adress.
next adress in **the dissassembly** seems to be **8DAC** as in **tsrldyn** located at **0x380DAC**, we find it at bit **8-9** as **"AC 8D"** which reversed becomes **8DAC**, swap it for your own **tsrldyn** adress.
line 7 done.

F7 F8 **F0 4F** 0D 09 D7 00 38 00 F6 8F **F0 4F** 0D 04

Line 8, we have the **counter** at bit **2-3**, in form of **"F0 4F"** which reversed is **4FF0** as in **0x384FF0**, swap it for your counter adress. it also appears in bit **C-D**, as **"F0 4F"**, swap this one too.
line 8 done.

D7 00 38 00 F6 8E **F0 4F** F3 F8 F3 8A DB 00 FF FF

Line 9, the last line, and a bit special one too..
we find the counter again at bit **6-7** as **"F0 4F"**, swap it for yours..
now the absolute last one we need to fetch from the location where the new function is linked-in, this IS **the original command that we replaced when we made the calls to this new function**.

Go make the link-in and get back to this, be sure to copy the original row where you make the link-in, and save it in your notebook.

in the **M-box** the **routine** where you make the **"calls"** look like this originally:

8b3a0h: F0 49 F7 F8 AC 8D **F3** *F8* **F3 8A** D7 40 06 02 03 F8

the value we are looking for here are here named **"F3 8A"** which reversed is **8AF3**.. it is located in the two bits in front of the **"D7 40 06 02 03 F8"** that you searched for earlier, **BEFORE** altering this routine with your **calls**.
when making the call **the adress will be overwriting the F3 8A part..** which is **why you saved this** line earlier, right? Right!

things are a bit confusing in the M-box since this one is actually **LOCATED** at 8A**F3** and that causes the **F3** to appear two times in the same row.. but this is the **later** one, which you would be writing over with the adress in **the mbox link-in**:

```
8b3a0h: F0 49 F7 F8 AC 8D DA 8F 60 FA D7 40 06 02 03 F8
```

so onto line **9** again.. look at bit **A-B**, and pls look at the right line, **line 9**, in the text way **above** us, there you will see **"F3 8A"**, this in reverse is the **8AF3** we were seeking, swap this for the adress you overwrote when making the **link-in** to your function.
line 9 done.

copy paste all the modified lines together so them form a nice block of 9 rows as per example:

```
9A 2B 13 80 F2 F4 40 8E D7 00 81 00 F2 F9 00 7E
40 49 9D 0B F2 F4 7A F8 D7 00 81 00 F2 F9 02 7E
40 49 FD 03 F7 8E AC 8D 0D 2F 9A 2B 29 80 8A 2B
22 60 F2 F4 7A F8 D7 00 81 00 F2 F9 06 7E 40 49
FD 1A C2 F4 02 8B D7 00 81 00 C2 F9 08 7E 40 49
FD 12 D7 00 38 00 F2 F4 F0 4F D7 00 81 00 F2 F9
04 7E 40 49 9D 11 F7 8E AC 8D 08 41 D7 00 38 00
F7 F8 F0 4F 0D 09 D7 00 38 00 F6 8F F0 4F 0D 04
D7 00 38 00 F6 8E F0 4F F3 F8 F3 8A DB 00 FF FF
```

Your new function has now been created!

paste it in your location, making sure NOT to move any of the other code, or the end of file.. how to do this depends on which hex editor you are using, but im sure youll figure it out.

Go back!