Configuring MegaSquirt-II

Tuning Software

The software application you use to tune and configure your MegaSquirt-II is called **MegaTune2.25**+ by *Eric Fahlgren*.
Be sure to get the latest version of MegaTune 2.25+. Some earlier versions of MegaTune 2.25 will NOT work with the production releases of MegaSquirt-II (or MicroSquirt) - *some important specifications were changed in beta testing*. You can get the latest version (MT225) here:

http://www.megamanual.com/files/software/

You also need Windows 9X/ME/XP and a conventional serial port to communicate with MegaSquirt-II (or MicroSquirt). USB will may work, and many people have reported that they have been successful using a USB-serial adapter. Just about any computer that is capable of booting Windows 95 (or better) will be fast enough, but get the fastest laptop computer you think is reasonably priced.
To tune all the parameters of MegaSquirt-II (or MicroSquirt) so that your engine runs the best it can, you will need to do the following:

1. First, install and learn to use **MegaTune2.25** or higher,
2. Next, set the constants,
3. Get the engine started and idling,
4. Then tune the PWM criteria,
5. Then tune the cold start and warm-up enrichments,
6. Then tune the VE table, the AFR table, and the spark advance table,
7. Finally, set the acceleration enrichments,

This section of the manual covers the first two items. the remainder are covered in the Tuning MegaSquirt-II (or MicroSquirt) section. Before starting this section, be sure you have read and understood the information on the MegaSquirt-II (or MicroSquirt) Installation page.

Using MegaTune2.25+

General Advice and Operation

Some general principles to follow when setting the configuration parameters are:

- All parameters with a **red label** should be set with the **engine off**, and the power to MegaSquirt® should be cycled (**switched off and on**) before starting the engine!
- Always save a combination after you have changed things, and give it a descriptive name. that is, don't save every file as *megasquirt.msq*, you won't be able to recover if you corrupt a file. Instead, save files with names like *msii_june2605.msq*, or some other scheme that makes sense to you and lets you identify how recent a file is. Note the MegaTune will suggest a filename of the form '*datalogyyyymmddhhmm.xls*'. There is more on this in the 'File/datalogging' section.
- When editing the tables, be sure to '***Burn Table***' when you are happy with it, or the changes you make will disappear when you shut off the power to MegaSquirt® EFI Controller,
- In general, change only those items you need to at first. If you are not sure what a parameter does, or whether it applies to you, leave it at the default value.

Some Warnings

Note that it is possible to damage your engine and/or MegaSquirt® by choosing inappropriate tuning or set-up parameters in MegaTune. Be sure to read this document, and all associated tuning documents that apply to your engine, vehicle, and tuning circumstances. In particular, read the safety precautions.

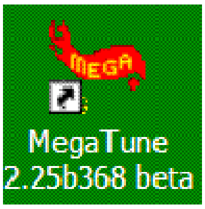
Installing MegaTune

MegaTune has an 'automated' install package, similar to most other modern Windows software. You download the set-up file (*see below*), click on it, and the package installs itself. You may have to select some options, or you may chose to select all the defaults.

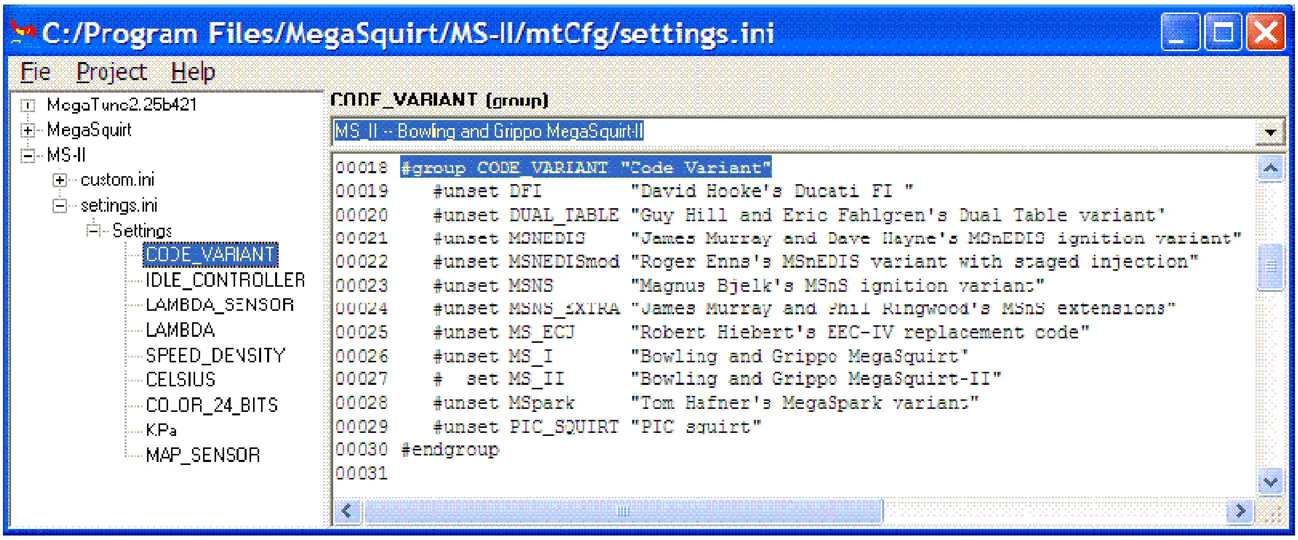
Files

Get the latest version of MegaTune from www.megamanual.com/files/software/.
To install the latest version of MegaTune on your computer:

1. Click this link to go to the release site,
2. Select the file **MegaTune225p3_setup.exe** and double click on it.
3. Choose '***Run***' or '*Run program from current location*' when prompted.
4. Choose all the defaults when installing the program, and it will create a folder called '*C:\Program files\MegaSquirt\MegaTune225*' and subfolders '*mtCfg*' and '*carMtCfg*'. The install program also creates an icon on your desktop with the name **MegaTune225...**, DON'T click it yet, you need to set up MegaTune (see '*Using MegaTune225 below*').



5. The install program will also create a folder in '*C:\Program files\MegaSquirt\*' called '*Car1*', this is where all your vehicle specific information will go.
6. Start up the MT Configurator (*mtCfg.exe*), which opens automatically during the installation process (it can be opened directly from the MegaTune folder at any later time). Open the MegaTune2.25b*xxx* tree item, and you'll see a number of megasquirt-II.ini.N.N files. Pick the one where N.N matches the version of embedded code you are running, highlight the .ini file version that you wish to use and execute File -> Activate. This will rename it to be "**megasquirt-II.ini**" (deleting the existing megasquirt-II.ini file first). It will then be used by MegaTune when it starts up. (Note that all 2.*XY* use the same 2.*X* INI file. For example, the 2.35 code uses the 2.3 INI file. This is the way the code upgrades are designed. Major revisions are incremented by +0.1 and require a new INI file, minor revisions (+0.01) use the **same** INI file.)
7. You need to tell MegaTune which version of MegaSquirt® you have (MS, MS-II, MSnS-E, etc.), as well as the EGO type, etc. To do this, use the mtCfg program, which opens automatically during the installation process (it can be opened directly from the MegaTune folder at any later time). Look for the CODE_VARIANT variable, it will be in '*Car1/settings.ini/Settings/CODE_VARIANT*' in the directory tree on the left side of the mtCfg window. Click on CODE_VARIANT. You can then use the drop box in the upper right section of the window to select your code variant.

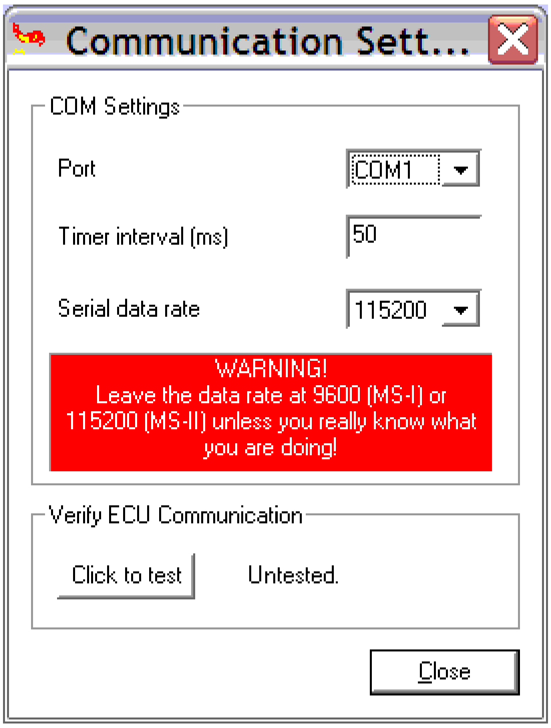


```
#group CODE_VARIANT "Code Variant"
   #unset DFI        "David Hooke's Ducati FI "
   #unset DUAL_TABLE "Guy Hill and Eric Fahlgren's Dual Table variant"
   #unset MSNEDIS    "James Murray and Dave Hayne's MSnEDIS ignition variant"
   #unset MSNEDISmod "Roger Enns's MSnEDIS variant with staged injection"
   #unset MSNS       "Magnus Bjelk's MSnS ignition variant"
   #unset MSNS_EXTRA "James Murray and Phil Ringwood's MSnS extensions"
   #unset MS_ECU     "Robert Hiebert's EEC-IV replacement code"
   #unset MS_I       "Bowling and Grippo MegaSquirt"
   #  set MS_II      "Bowling and Grippo MegaSquirt-II"
   #unset MSpark     "Tom Hafner's MegaSpark variant"
   #unset PIC_SQUIRT "PIC squirt"
#endgroup
```

While you are in the setting.ini files, you can verify or change the EGO sensor option under LAMBDA_SENSOR if necessary.

You can use mtCfg to change the color depth, temperature units (°F/C) and a large number of other parameters for MegaTune. See the documentation in the files displayed in mtCfg.
Be sure to '*Save*' the file.

8. Increase the value of the **interwrite delay** from the default of 1. An interwrite delay value of 5 seems to cure most comm issues. You will find this value in the common.ini file, which is usually in the '*mtCommon*' subfolder of the MegaSquirt® folder (i.e., *C:\Program Files\MegaSquirt\mtCommon\common.ini*). Make sure the preceeding line reads "writeBlocks = off", do not edit that line unless it says "on" (*the comment is misleading, interwrite delay is still used if writeBlocks = off*). You can open common.ini with Notepad, or copy it from this link: common.ini (right click and 'save as').
9. You should now be able to launch MegaTune. Double-click the MegaTune icon (*created above*) on the desktop. In order to communicate with MegaSquirt-II (or MicroSquirt), you may need to set the serial port configuration. To do this, got to the *Communications/Settings* dialog on the MegaTune menu.



Communication Sett...

COM Settings

Port: COM1

Timer interval (ms): 50

Serial data rate: 115200

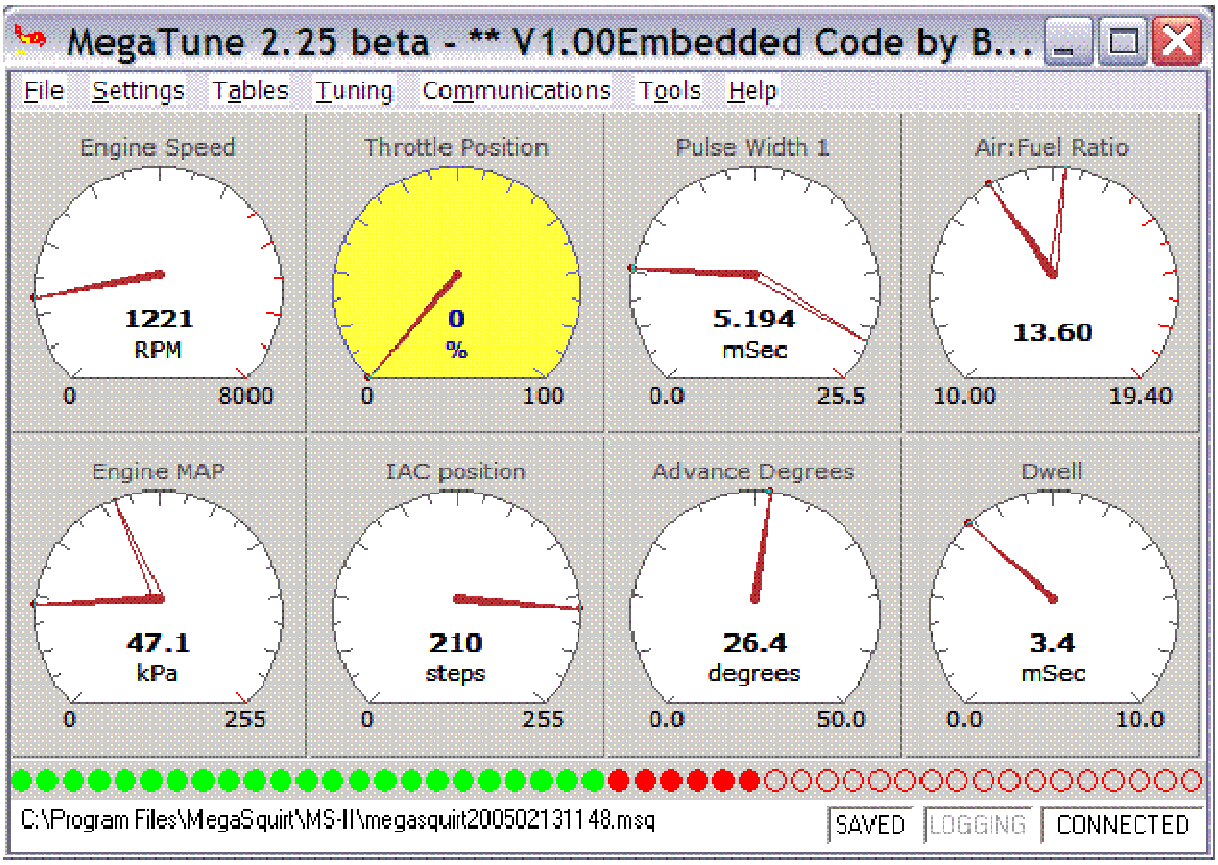
WARNING!
Leave the data rate at 9600 (MS-I) or 115200 (MS-II) unless you really know what you are doing!

Verify ECU Communication

- **Port**: The communications port number should correspond to the port to which the MegaSquirt® controller is attached.
- **Timer Interval (ms)**: The timer interval dictates how frequently the runtime and tuning displays are updated. An interrupt is generated at the specified interval, and the real time data is pulled down from the MegaSquirt® controller. Use 50-100 ms to start; you can try to smaller values (ex. 20 ms) if your computer is fast enough.
- **Serial data rate** (the variable *baud* in the code): This should be set at **115200** for MegaSquirt-II (or MicroSquirt).
- **Verify ECU Communications**: Click the button to verify communications with the MegaSquirt-II (or MicroSquirt) controller. If communications are established between MegaTune and MegaSquirt-II (or MicroSquirt), the word '**Success**' will appear when the '*Click to test*' button is clicked.

  If '**No response!**' is reported, check that you have the correct COM port selected, and that no other software is using that port (a modem or fax, for example). (*You should have already checked this and the DB9 cable configuration while building MegaSquirt.*)

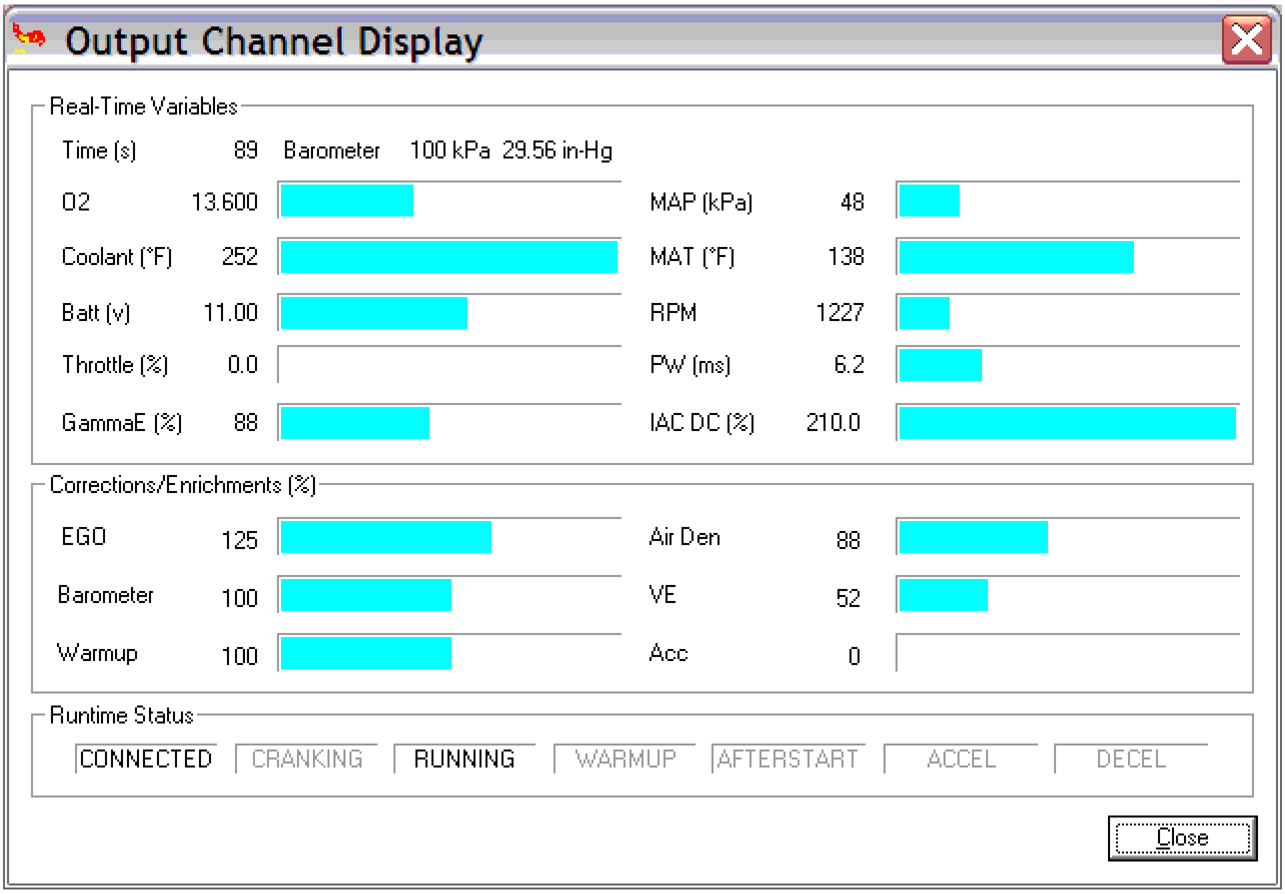
If you use a USB adapter, be SURE to set the buffer size to ZERO (0). You can change this in the Control Panel, look under 'System, Hardware, Device Manager'. The default buffer size for these adapters is typically 14 bytes. If this is the case with yours and you left the default, you would have no control over the data flow no matter what else you do.
When MegaSquirt® is loaded and communicating with MegaSquirt-II (or MicroSquirt), the first thing you will see is the front page, which shows eight of the more useful gauges:



A number of the sensor readings are displayed, as well as some outputs. For example, the injector pulse width is the measure in milliseconds of how long the injector is opened for each pulse, regardless of how many times it is opened in a cycle. Duty cycle gives the percentage of time the injector is open irrespective of individual pulse duration.

There is a bar gauge across the bottom of the window shows the oxygen sensor reading. The scale is determined by egoGauge value in the Tuning section of the **MegaTune2.25**+.ini file. This same setting controls the analog and bar gauges on the tuning page. The first value of this setting controls the lowest voltage displayed on the gauges, the second number controls the highest and the optional third value specifies the “alert” value, above which the LEDs are red. The bottom of the front page contains a status bar. The current file name (used for Save operations) is displayed in the left part of the status bar, followed by “saved” status. When the memory image has been modified since the last Open or Save operation, this entry shows “**SAVED**” in bold face.

However the front page is limited to eight gauges, which may not be enough, especially when you are learning to use MegaTune. If you go to the 'Tuning/Realtime Display', you can see many more at once:



Output Channel Display

Real-Time Variables

Time (s) 89 Barometer 100 kPa 29.56 in-Hg

O2 13.600 MAP (kPa) 48

Coolant (°F) 252 MAT (°F) 138

Batt (v) 11.00 RPM 1227

Throttle (%) 0.0 PW (ms) 6.2

GammaE (%) 88 IAC DC (%) 210.0

Corrections/Enrichments (%)

EGO 125 Air Den 88

Barometer 100 VE 52

Warmup 100 Acc 0

Runtime Status

This is a good page to monitor while experimenting with the stimulator and MegaSquirt-II (or MicroSquirt).

MegaTune2.25+ allows you to save and restore configurations as disk files (they have an **.msq** extension). Use the *Open*, *Save* and *Save As* menu items to do this.

Setting the General Parameters

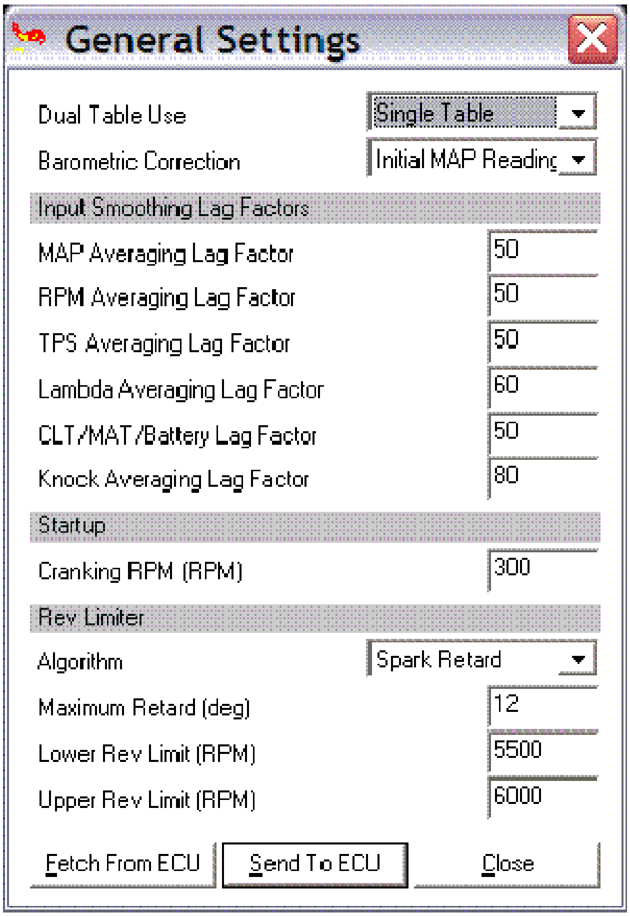
Note that as the code develops, some advanced options may not be documented here. To find information on these advanced options, see the MegaTune help file.

Some general principles to follow when setting the configuration parameters are:

- Always save a combination after you have changed things, and give it a descriptive name. that is, don't save every file as *megasquirt.msq*, you won't be able to recover if you corrupt a file. Instead, save files with names like *msii_june2605.msq*, or some other scheme that makes sense to you and lets you identify how recent a file is.
- When editing the tables, be sure to 'Burn Table' when you are happy with it, or the changes you make will disappear when you shut off the power to MegaSquirt-II (or MicroSquirt)
- In general, change only those items you need to at first. If you are not sure what a parameter does, or whether it applies to you, leave it at the default value.

Now we are ready to start configuring MegaSquirt-II (or MicroSquirt)!

On the main MegaTune2.25+ menu is an item called '*Settings/General*'.



General Settings

Dual Table Use	Single Table
Barometric Correction	Initial MAP Readinç
Input Smoothing Lag Factors	
MAP Averaging Lag Factor	50
RPM Averaging Lag Factor	50
TPS Averaging Lag Factor	50
Lambda Averaging Lag Factor	60
CLT/MAT/Battery Lag Factor	50
Knock Averaging Lag Factor	80
Startup	
Cranking RPM (RPM)	300
Rev Limiter	
Algorithm	Spark Retard
Maximum Retard (deg)	12
Lower Rev Limit (RPM)	5500
Upper Rev Limit (RPM)	6000

You can set these as follows:

- **Dual Table Use** choose whether you want to control each bank of injector independently, or to run both injector banks off one set of VE/AFR tables.
- **Barometric Correction** choose whether you want:
  - '*None*' - no barometric correction,
  - baro correction based on the initial start-up MAP reading ('*Initial MAP Reading*'), or
  - '*Two Independent Sensors*' for continuous baro correction (ONLY if you have installed a second MAP sensor.)
- **X-Tau Usage**X-Tau usage can be set to off, or accel/decel using this parameter. The X-Tau tuning values are set in the X-Tau Time Table menu item under Tables. See the X-Tau page for more information.
- **Prime, ASE, WUE Tables**. Chose whether you want two point (a high temperature and a low temperature settings) or 10 entry tables for the prime pulse, afterstart enrichment, and warmup enrichment. The tuning values are listed in under the menu item Tables.
- **Input Smoothing Lag Factors** Lag factors force the variables to change more slowly than the actual input value. Note that in all cases, 100 is no lag effect at all, and smaller numbers slow the input response speed.

  The lag factors are used as follows:

  NewValue = PreviousValue + (NewValue - PreviousValue) * (LagFactor/100%)

  In each case the PreviousValue is itself filtered.

For example, suppose your last MAP value was 70, the lag factor is **40** and the current MAP value read directly from the sensor is 90. Then MegaSquirt-II (or MicroSquirt) will calculate:

NewValue = 70 + (90 - 70) * (40/100) = 70 + 20*0.4 = 78

If the MAP then remains at 90, then next value would be:

NewValue = 78 + (90 - 78) * (40/100) = 78 + 12*0.4 = 82.8

and so on...
This has the effect of slowing MegaSquirt-II (or MicroSquirt)'s response to fluctuating input values, both smoothing them and reducing the effect of noise in the signals.
The following inputs can have lag factors applied to them:

- MAP Averaging Lag Factor,
- RPM Averaging Lag Factor,
- TPS Averaging Lag Factor,
- Lambda Averaging Lag Factor,
- CLT/MAT/Batt Averaging Lag Factor,
- Knock Averaging Lag Factor.

- **Start-Up**
  - *RPM*: This is the rpm at which MegaSquirt-II (or MicroSquirt) switches from the cranking pulse width injected at every ignition event to the pulse width calculated from the fueling equations (MAP, IAT, RPM, etc.). 300 rpm is good for most automotive engines, but a higher value may be needed for motorcycle engine or other specialized uses.
- **Rev Limiter**

  Algorithm - you can select:

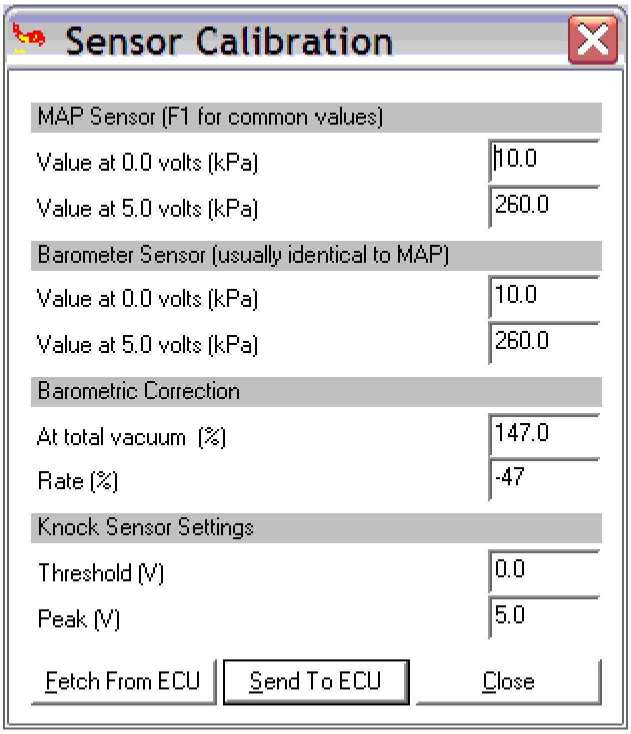
    - None - no rev limiter
    - Spark Retard - reduces revs by retarding the ignition advance,
    - Fuel Cut - reduces revs by eliminating fuel.
  - *Maximum Retard* is the MOST the spark can be retarded in spark Retard mode,
  - *Lower Rev Limit* is the level at which the the fuel is re-enabled in Fuel Cut mode, and the level at which timing is fully restored in Spark retard mode.
  - *Upper Rev Limit* is the level at which the rev limiter is initially applied.

For example, if you set the upper limit to 6000 and the lower limit to 5800, then in:

- *Spark Retard mode*:
  - Spark would be applied normally (from the advance table, etc.) until you reached 5800 rpm.
  - It would then be cut linearly as the revs increased to 6000 rpm.
  - Note that the spark is retarded with rising revs between 5800 and 6000, as well as when the revs fall.
- *Fuel Cut mode*:
  - Fuel would be applied normally (from the advance table, etc.) until you reached 6000 rpm.

- Fuel would then be cut completely, and restored ONLY when the revs dropped to 5800 rpm.
- Note that the fuel is NOT cut with rising revs between 5800 and 6000, only as the revs fall.

If you are using a knock sensor, separate baro sensor, or non-standard MAP sensor, you can configure MegaSquirt-II (or MicroSquirt) to work with them using the '*Sensor Calibration*' dialog under '*Tools*'.



Sensor Calibration

MAP Sensor (F1 for common values)

Value at 0.0 volts (kPa) 10.0

Value at 5.0 volts (kPa) 260.0

Barometer Sensor (usually identical to MAP)

Value at 0.0 volts (kPa) 10.0

Value at 5.0 volts (kPa) 260.0

Barometric Correction

At total vacuum (%) 147.0

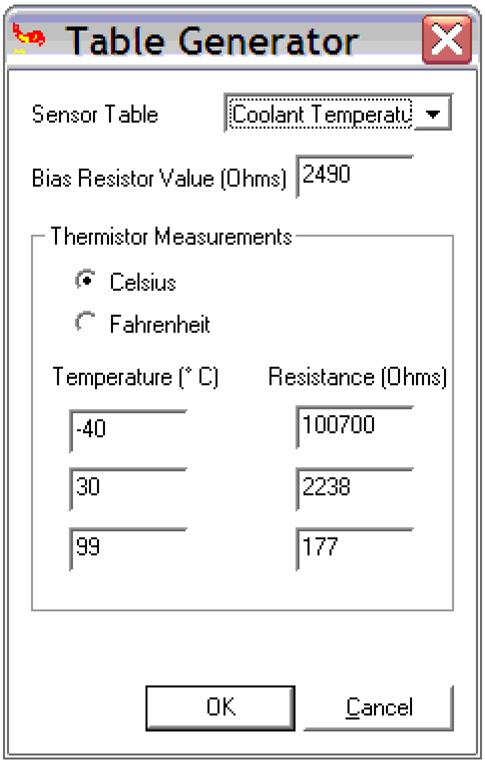
Rate (%) -47

Knock Sensor Settings

Threshold (V) 0.0

Peak (V) 5.0

The MegaSquirt-II code is set up for the standard General Motors temperature sensors. If you are using other sensors, you can use the '*Calibrate Thermistor Tables*' dialog under '*Tools*'. You enter the bias resistor value and three temperature/resistance points, and the table is created and downloaded to MegaSquirt-II (or MicroSquirt) for you.



Sensor Table: Coolant Temperatu

Bias Resistor Value (Ohms): 2490

Thermistor Measurements

● Celsius

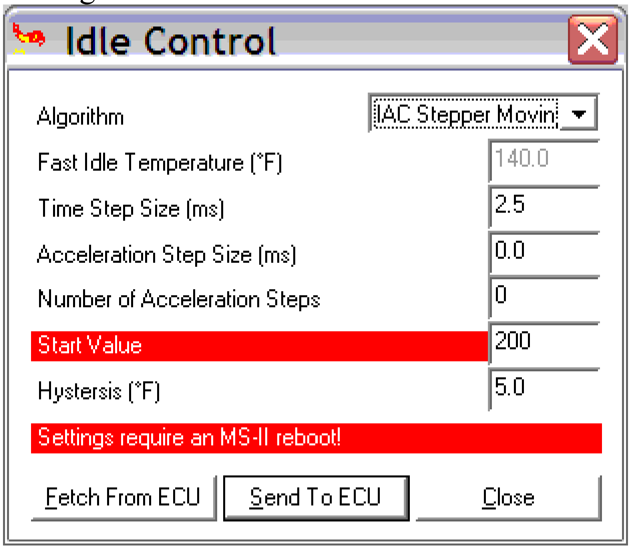
○ Fahrenheit

Temperature (° C)	Resistance (Ohms)
-40	100700
30	2238
99	177

IMPORTANT NOTE: Do NOT burn tables ('Calibrate AFR Table' or 'Calibrate Thermistor Tables') on a running engine. Even idle is NOT allowed, because these tables ONLY exist in flash, so once a table is erased, there is nothing but garbage in there until it is re-programmed, one word at a time. Until that reprogramming is complete, operating the engine is unsafe.

Idle Control Settings

The basic idea of IAC is that the motor or PWM solenoid starts out with a large opening of the air valve at cold startup, then gradually closes as the coolant temperature rises. The basic motor position at any given time is determined from the input table of step position versus coolant temperature. To this basic control algorithm, several features have been added as described below. You need to set MegaSquirt-II (or MicroSquirt) to tell it if you have a fast idle "solenoid type" valve or a stepper motor IAC, or neither. These are selected under **Settings/Idle Control** in MegaTune:



Algorithm	IAC Stepper Movin
Fast Idle Temperature (°F)	140.0
Time Step Size (ms)	2.5
Acceleration Step Size (ms)	0.0
Number of Acceleration Steps	0
Start Value	200
Hystersis (°F)	5.0

Settings require an MS-II reboot!

If you have an IAC stepper motor, you will have to choose between 'moving only' and 'always active'. If you set your stepper to 'always on' for 15 min or so and it doesn't feel too hot to you, then you can leave it that way. Apparently this is what General Motors does. But if you want to be safe you can test it on the bench for 15 min or so. It will get right warm, but it shouldn't burn your fingers just touching it. If it gets too hot, use 'moving only' instead.

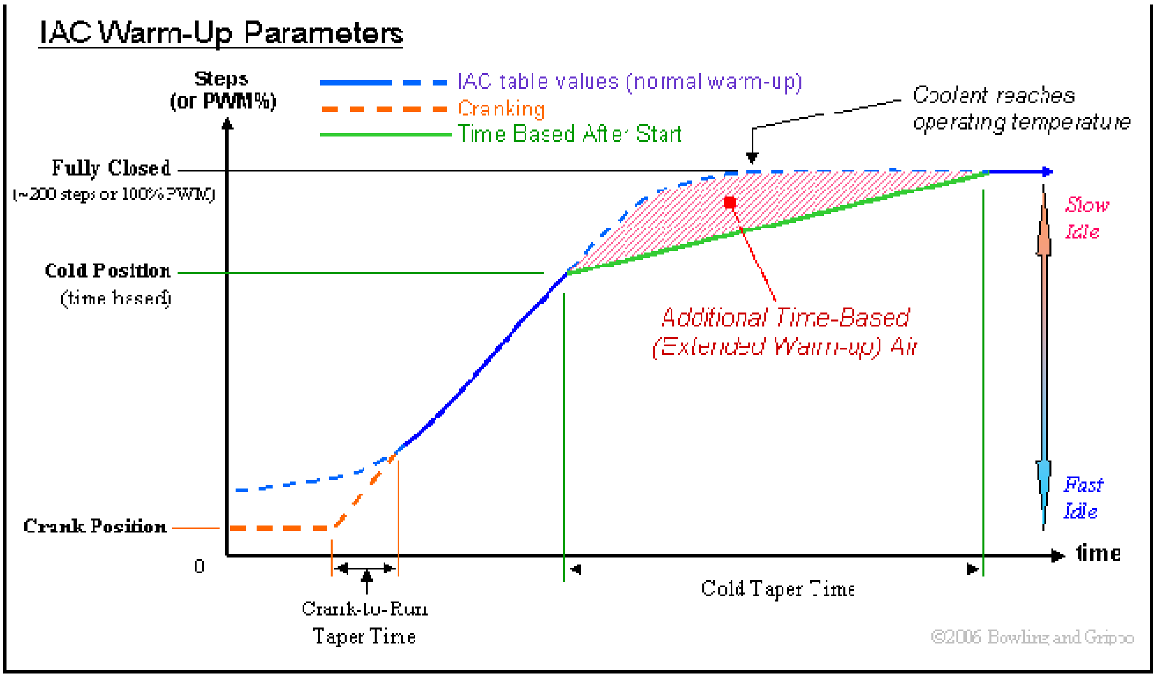
- **Algorithm** (*IdleCtl*): If you have a:
  - **Fast Idle Valve (*FIdle*)**:
    - For an on/off fast idle valve, set the algorithm to '**Solenoid**'. You can also set your **Fast Idle Threshold** if you have installed a fast idle solenoid. Enter a coolant temperature to turn on the fast idle solenoid. A typical value is 145° Fahrenheit. The Fast Idle valve will be activated below this temperature (145ºF) and turned off above 145ºF. The Fast idle Threshold is independent of any warm-up enrichment. Fast idle valves generally have one or two wires.
    - **PWM Warmup**: This is for the Ford or Bosch pulse width modulated idle air valves. Ten temperature dependent levels of PWM are user specifiable if this option is selected (see '*Idle PWM Dutycycle*' under 'Tables'). Modifications to the board are required, see this link for more details.
  - **Idle Air Controller (IAC)**: If you have a stepper motor IAC, you can set the **IAC Start** position, as well as ten intermediate positions based on the coolant temperature to allow a decreasing amount of "extra air" as the engine warms up. These are set under '*Tables/Idle Steps*' in MegaTune. Stepper motor IACs usually have four wires.
    - **IAC Stepper Moving Only**: Powers the stepper only when changes in pintle position are requested. This is the most common type, it holds its position if not powered, and is difficult to turn by hand.
    - **IAC Stepper Always On**: Powers the stepper at all times. Required if your stepper 'free wheels' when you spin its pintle un-powered with your hand.
    - **15 Minute IAC**: This operates the IAC stepper motor as 'always on' for 15 minutes, then switches to 'moving only'. This can be useful in some situations in which the stepper moves unreliably if moving only at the lower voltages of cranking and warming up, etc.

  To select the appropriate '*Idle Control/Algorithm*' for stepper motor control in MegaTune you may need to do some testing. In some cases setting the stepper motor to "IAC Stepper Always On" will cause the IAC to get hot. However setting it to "IAC Stepper Moving Only", might cause a problem with idle speed changing from one start to another.
  You can test if your IAC is suitable for 'always on' by leaving your stepper powered on the bench for 15 min or so. If it doesn't feel too hot to you, then set it to "Always On". Apparently this is what GM does. But if you want to be safe they should test it on the bench for 15 min or so, or monitor it closely in the car while not moving for at least 15 minutes, checking the IAC temperature frequently with your fingers. It may get warm, but it shouldn't burn your fingers just touching it.

- Leave the other values (below) alone, you can experiment with them when you get the engine running.

- **Time Step Size (ms)** (*IACtstep*): IAC stepper motor nominal time between steps (i.e., 2.5 milliseconds gives pulse frequency of 400 Hz).
- **Acceleration Step Size (ms)** (*IACaccstep*): not currently used.
- **Number of Acceleration Steps** (*IACnaccstep*): not currently used.
- **Start Value** (*IACStart*): The number of steps applied to retract the IAC pintle to 'wide open' at power up.
- **Cranking Position** (*IACcrankpos*): During cranking, extra air may be useful in the same way as extra fuel in cranking pulses. The table value for the starting temperature may be fine after the engine has started up, but during cranking more power may be needed, especially if the starting temperature is cold. To provide this, you can input a step position that provides a larger than normal air opening during cranking. So, if in cranking and 'Cranking Position' < table value, then the IAC motor position (or PWM%) is set to 'Cranking Position', and when cranking is done, the motor position starts tapering (over the 'crank to run taper time') up to the table value over a user input period, typically a few seconds. (See the diagram below) If this feature is not desired, Just set 'Cranking Position' to a value higher than any table value. Then the table value will always be used since it provides more opening.
- **Crank-to-Run Taper Time** (*IACcrankpos*): This is the time over which the cranking position of the idle (either the stepper steps or the PWM%) is moved to match the table value (see diagram below). Higher values give a higher idle for longer periods, which can improve starting performance.
- **Hysteresis (°)** (*IdleHyst*): This input can be used to avoid continuous motor motion (and wear) for small coolant temperature changes. Changes to the motor are only made when new coolant temperature > coolant temperature on the last move, or, new coolant temperature < (coolant temperature on the last move - Hysteresis temperature). What this does is allow constant motor motion while the coolant temperature is rising, but when it peaks, there will be no further motion unless things cool back down - which is unlikely.
- **Time Based After Start (extended warm-up)**: You should NOT use the Time Based After Start (extended warmup) option unless you find you need it, and very few will. Disable it by setting the 'cold temperature to -40°F. Then this feature will not be used unless the coolant temperature at startup (ECU first powered on) is below -40°F. This feature is used toward the end of the warmup sequence when the coolant temperature is close to its final operating temperature. In this case, fast idle will normally come off, but SOME cars (very few) may need extended fast idle. An example is a car that uses heavy weight oil, which is nowhere near at operating temp when the coolant gets there, plus a hot cam with not enough idle torque to overcome the oil drag.

  This feature is implemented by inputting a 'Cold Position' that is the step position at start of extended warmup, typically about 80% of the final, fully closed step position. The IAC behaves normally until the step position commanded from the table just exceeds this Cold position value (either PWM or stepper). From that point on, the steps are tapered in so as to reach the last step value in the table over the 'cold taper time' period. (see the diagram). This slows the reduction in idle air as the engine continues to warm up (increasing the idle speed for longer than the coolant temperature alone would do).



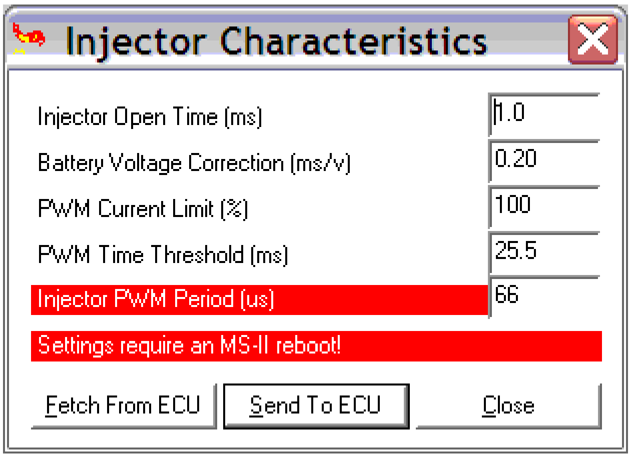
- **Cold Temperature (°)** (*IACcoldtmp*): This defines the initial coolant temperature below which the afterstart taper will be extended, based on the Cold Position and Cold Taper Time. It should be set fairly cold, generally not more than 20° F.
- **Cold Position (steps)** (*IACcoldpos*): The Idle PWM values at which time based afterstart tapering is initiated. Note that this value must be higher than the lowest value in your IAC PWM table, or you can get strange operating results.
- **Cold Taper Time (sec)** (*IACcoldxt*): This is the number of seconds that MegaSquirt® takes to move from the 'cold position' to the position indicated in the IAC step table for the current coolant temperature.

Setting the Injector Criteria

Before attempting to start your MegaSquirt-II (or MicroSquirt) equipped engine, you will need to set a number of parameters that determine how MegaSquirt® injects fuel. These include the injector open time, Req_Fuel, injector control criteria, PWM criteria, EGO characteristics, etc. These constants are either calculated, or based on the configuration of your system. For the most part, these are very similar to those from MegaSquirt.

Note that for a Wankel rotary engine (Mazda 13B, etc.), see the MegaSquirt & Rotary Engines document for settings and other advice.

On the Settings/Injector Characteristics page:



Injector Characteristics

Injector Open Time (ms) 1.0

Battery Voltage Correction (ms/v) 0.20

PWM Current Limit (%) 100

PWM Time Threshold (ms) 25.5

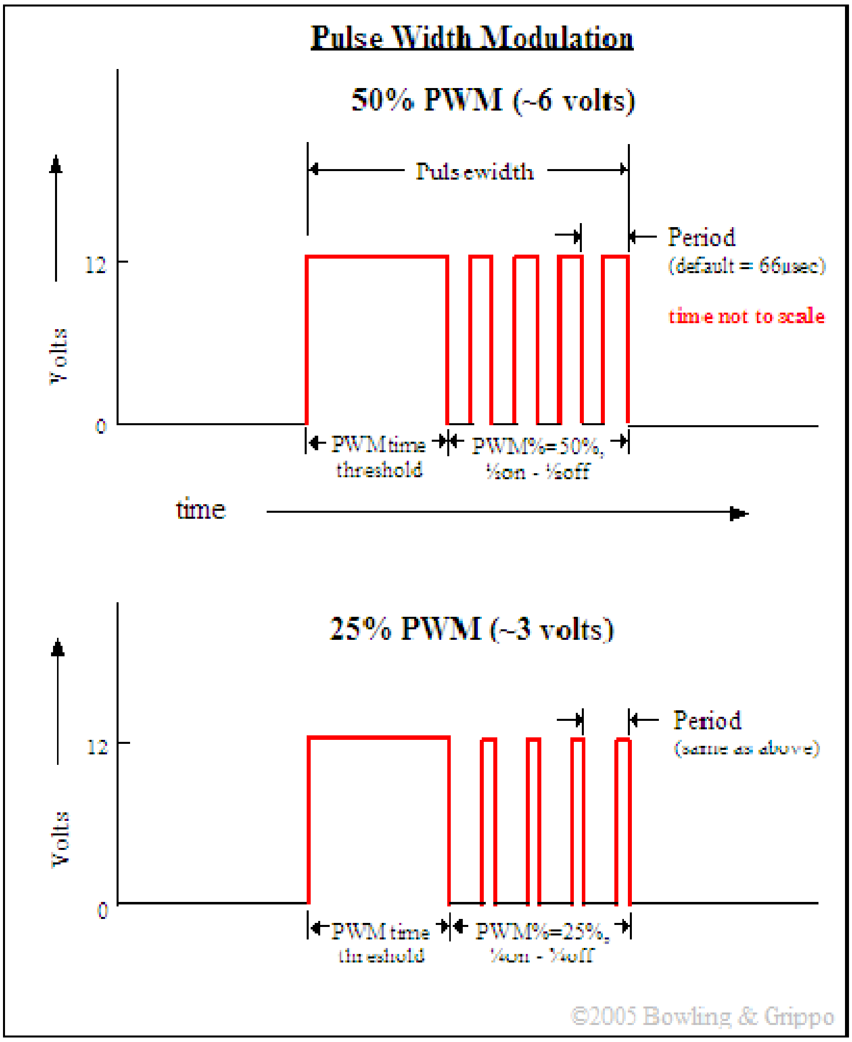
Injector PWM Period (us) 66

Settings require an MS-II reboot!

- **Injector Opening Time (ms)** (*InjOpen*) is the amount of time required for the injector to go from a fully closed state to a fully opened state when a 13.2 volt signal is applied. Since fuel injectors are electro-mechanical devices with mass, they have latency between the time a signal is applied and the time they are in steady-state spraying mode. Typically, this value is close to 1.0 milliseconds. Note that the closing time should be subtracted from this value, but it is generally very small and can be neglected.
- **Battery Voltage Correction (ms/V)** (*BatFac*) is the number of milliseconds that MegaSquirt-II (or MicroSquirt) adds to each fuel injection pulse to compensate for the slower opening of the injectors with lower supply voltages. Generally 0.10 ms/V to 0.20 ms/V is about right. So, with a 0.20 Battery voltage correction factor and supply voltage of 14.5 Volts, a 1.0 millisecond 'opening time' is adjusted to 1.0 - (14.5-13.2)*0.20 = 1.0 - 0.26 = 0.7 milliseconds.

  Conversely, If you had injector specs that stated the opening time was 0.7 milliseconds at 14.5 Volts, and your battery correction was 0.20, then you should enter 1.0 as the 'Injector Opening Time'.

- **PWM Time Threshold (ms)** (*InjPWMTim*) is the amount of time the Pulse width has to be on before PWM starts. This allows full voltage to reach the injectors while opening. Generally you should set this to the same value as your injector opening time (~1.0 millisecond)
- **Injector PWM Period (µsec)** (*InjPWMPd*) is the time between cycles of on/off and the Injector Duty Cycle is the % of time it stays on relative to the total time for one cycle. You use high frequency to make things smooth. Since the injectors stay open for milliseconds, you need a period that is much shorter than that. Such a frequency never lets the injector start to close - the turn off turn on cycle is so fast that the injector stays where it is. Keep this value between 10 and 25 KHz (100-40 µsec).



To tune the PWM [pulse width modulation] values for your engine, you need to know what kind of injectors you have- low impedance or high-impedance. If you are running **high-impedance injectors** (greater than 10 Ohms), then set the PWM time to a number like 25.4, in essence you are disabling the PWM mode. This allows full voltage to the injectors throughout the pulse width.

For **low-impedance injectors** (less than 3 Ohms), you need to limit the current to avoid overheating the injectors. To do this, there is a period of time that you apply full battery voltage [peak] current, then switch over to a lower current-averaged [hold] current, i.e. peak and hold. Alternatively, you can add resistors in series with the injectors. See the Injectors and Fuel Supply section of the MegaSquirt® manual for more details.

To run low-impedance injectors with the PWM current limit mode, you need to set two parameters - the "**PWM Current Limit %**" and the "**Time Threshold for PWM Mode**" - both are on the “Constants” page. The current limit % is the percent duty cycle when the current limit is invoked. The time threshold is the amount of time from when the injector is first opened until the current limit is activated.

High impedance injectors can run on 12 Volts without problems. Low-inpedance injectors require some form of current limiting. MegaSquirt® has pulse width modulation to limit the current. You need to set the PWM parameters to match your injectors:

- If you are running **high-impedance injectors** (greater than 10 Ohms), then set the:
  - **PWM Time Threshold** to **25.4 msec**, and the
  - **PWM Current Limit (%)** to **100%**.

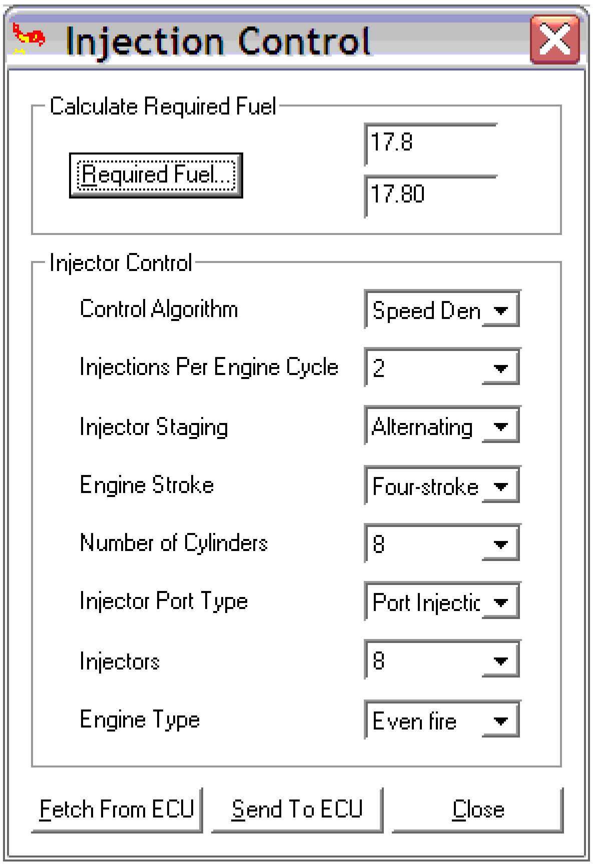
(The presence or absence of the active flyback circuit (v3) or flyback board (v2.2) doesn't matter for high impedance injectors.)

- If you have **low impedance injectors** (less than 4 Ohms), set the:
  - **PWM Time Threshold** to **1.0 msec**, and
  - **PWM Current Limit** to:
    - **30%** on a V3 or V3.57 main board (if you have the active flyback circuit installed).
    - **30%** on a V2.2 main board (if you have the separate flyback board installed).
    - **75%** on a V2.2 main board (if you **do not** have the separate flyback board installed).

    *Set the initial PWM% to **75%** if and only if you have you impedance injectors and have **NOT** installed:*

    - *the active flyback circuit on a V3 main board (assembly manual step #69), or*
    - *the 'Flyback Board' daughter card on a V2.2 main board).*

Note that most distributors are shipping the V3 main board with the flyback circuit installed. If you are not sure, look for the presence of **Q3** and **Q11** on the heat sink (3rd and 7th from the left), or ask the distributor.
You will tune these after getting the engine running. See “Setting the PWM Criteria” in the tuning section of this manual. **Failure to perform the tuning steps can result in damage to your injectors**. If you have high-impedance injectors, set these values to 25.4 ms and 100%, and you do not need to tune them further.
On the 'Settings/Injector Control' page:



Calculate Required Fuel

17.8

17.80

Injector Control

Control Algorithm: Speed Den

Injections Per Engine Cycle: 2

Injector Staging: Alternating

Engine Stroke: Four-stroke

Number of Cylinders: 8

Injector Port Type: Port Injectic

Injectors: 8

Engine Type: Even fire

- **Required Fuel** – (*ReqFuel*) this is top field of the Constants window. It has a calculation dialog to help you find an appropriate value. It should contain the injector pulse width, in milliseconds, required to supply the fuel for a single injection event at stoichiometric combustion and 100% volumetric efficiency.

  In order to come up with this value, MegaTune provides a calculator that will suffice for 99% of applications (those for which it will not work generally require changes to the MegaSquirt® controller code itself, and that is beyond the scope of this manual). To use the calculator, click on the Required Fuel button, and fill in the fields (*Engine Displacement, Number of cylinders, Injector flow,* and *Air:Fuel ratio*, then click 'Okay').
  For a 4-stroke, a complete stroke cycle is 720 degrees of crankshaft rotation (i.e. two revolutions); for a 2-stroke, it is 360 degrees (this is also factored in the REQ_FUEL value down loaded to MegaSquirt).
  In the tuning software, the **upper REQ_FUEL box** is the amount per cylinder, as noted above. The **lower REQ_FUEL box** is the value down loaded to MegaSquirt. It is the REQ_FUEL number on top, but scaled by your selected injection mode (number of squirts and alternate/simultaneous).
  For example, if you inject simultaneous and one injection, and have the same number of injectors as cylinders [i.e. port injection], then REQ_FUEL on the bottom is the same as REQ_FUEL on top. Same with alternate and two squirts. If you put in simultaneous and two squirts, then REQ_FUEL is divided in half - because you squirt twice, you need to inject 1/2 the fuel on each shot.
  Note: if you choose alternating for port injection, make sure your number of squirts is an even number (2,4,...) and evenly divisible into the number of cylinders. For example, with an eight cylinder engine, you could use alternating and 2, 4, or 8

squirts/cycle. With a six cylinder, if you choose alternating, you MUST use 2 or 6 squirts/cycle. Also, the only possible combinations for an odd-cylinder count engine are either 1 squirt/simultaneous or N squirt/simultaneous combination, where N is the number of cylinders."

Permissable Combinations:

Number of Cylinders

		1	2	3	4	5	6	8	10	12
	1	OK	*simultaneous only*	*simultaneous only*	*simultaneous only*	*simultaneous only*	*simultaneous only*	*simultaneous only*	*simultaneous only*	*simultaneous only*
	2	no	OK	no	OK	no	OK	OK	OK	OK
	3	no	no	*simultaneous only*	no	no	*simultaneous only*	no	no	*simultaneous only*
	4	no	no	no	OK	no	no	OK	no	OK
	5	no	no	no	no	*simultaneous only*	no	no	*simultaneous only*	no
Number	6	no	no	no	no	no	OK	no	no	OK
of	7	no	no	no	no	no	no	no	no	no
squirts	8	no	no	no	no	no	no	OK	no	no
	9	no	no	no	no	no	no	no	no	no
	10	no	no	no	no	no	no	no	OK	no
	11	no	no	no	no	no	no	no	no	no
	12	no	no	no	no	no	no	no	no	OK

*"**OK**" means the combination will work with either simultaneous or alternating. "no" means it will not work with either, i.e., not at all.*

The maximum injector pulse width possible with MegaSquirt-II (or MicroSquirt) is 65 milliseconds.

- **Control Algorithm** (*FuelAlpha*) lets you choose between Speed Density and Alpha-N. In all cases, you should choose speed density unless you have a good reason to do otherwise, and understand how this will change your tuning efforts. All tuning advice in this manual is based on the speed-density algorithm. Alpha-N uses the throttle position (alpha) and RPM (N) to calculate the amount of fuel to inject as opposed to using the manifold absolute pressure (MAP) and RPM to calculate the amount of fuel to inject. Alpha-N is useful for long duration cams where the resolution of manifold air pressure (map) would be small. It is also useful to get smoother idle on engines that have erratic map values. MegaSquirt® be converted from its default **speed-density** calculations to **Alpha-N** which uses RPM, temperature and TPS only. You must have version 2.0 (or higher) of the embedded software installed. Start up the tuning software, go to the Constants dialog and change speed density to Alpha-N. Re-map your VE table. You will no longer use the MAP sensor for estimating the load on the engine, the throttle position and rpm are used instead. This can help with cams with long duration and/or a lot of overlap, as they have low and variable vacuum at idle, making tuning very difficult.
- **Injections per Engine Cycle** (aka. "squirts") is set the number of squirts you want per engine cycle. You want this to be set so that your idle pulse width is no less than 2.0 ms, if possible, and your Req_Fuel is less than 15-18 milliseconds, but more than 6 milliseconds. These values allow proper tuning of the idle mixture while maintaining

the ability to apply enrichments (acceleration, warm-up, etc.) under full throttle. This is the total injector events that you wish to occur for every engine cycle (360 degrees for two stroke engines and 720° for four strokes).

Note that this works with the 'injector staging' parameter (*below*) to determine the base fuel pulse width, so be sure to read that section as well.
There is some benefit to choosing 2 squirts/alternating for port injection, since only half of the injectors fire at once, the pressure drop in the fuel rails is reduced and the fuelling is more consistent.
With throttle body injection, the number of injection/cycle you can will depend on your number of cylinders, plenum size, Req_Fuel, etc. You have to experiment to see what works best for your combination. Generally, you will need at least as ½ as many squirts per cycle as you have cylinder, though you can run this alternating.

- **Injector Staging** (*Alternate*) can be either simultaneous (both injector drivers - i.e., **all** the injectors - fire at once), or alternating (one injector driver fires on one injection event, the other fires on the next, and so on; 'ping-ponging' back and forth). If you choose alternating for port injection, make sure your number of squirts is an even number (2,4,...) and evenly divisible into the number of cylinders. For example, with an eight cylinder engine, you could use alternating and 2, 4, or 8 squirts/cycle. With a six cylinder, if you choose alternating, you MUST use 2 or 6 squirts/cycle. Also, the only possible combinations for an odd-cylinder count engine are either 1 squirt/simultaneous or N squirt/simultaneous combination, where N is the number of cylinders."

  Note that with port injection and alternating injection, you must choose at at least 2 squirts per cycle, otherwise every other cycle for each cylinder will get NO fuel! The engine will run very badly. (One squirt is fine with simultaneous.)
  There are a number of considerations in choosing the number of squirts:

  - Throttle body injection (TBI) and port injection are completely different in regard to the required number of squirts - so this must always be taken into account. TBI often requires at least half the number of squirts as cylinders fed by the common plenum (ex.)
  - Increasing the number of squirts alone increases the accel enrichment (since the accel enrichment is per squirt),
  - The timing of the injection (the 'injection timing delay' under 'Fuel Set-Up/General') can make a big difference to how sensitive the engine is to the number of squirts (since with fewer squirts you might be squirting into the disrupted airflow of the overlap period at low speeds).

  More squirts are more likely to help if the engine:

  - has a low idle rpm,
  - is tuned to relatively lean (ex. stoichiometric or leaner) idle mixtures,
  - has high idle advance (say more than 15° to 20° BTDC),
  - wants larger accel enrichments.

However, if the engine is tuned so that the idle MAP is minimized (likely 13.5:1 AFR or so), and has a higher idle speed (say 800 rpm compared to 600 rpm), then more squirts is less likely to help.
There are definitely some down sides to more squirts:

- More squirts makes the tuning more sensitive to the correct opening time, and this is one of the harder things to determine accurately.
- More squirts reduces the effective resolution of the req_fuel, accel enrichments, etc. (since they are adjusted in 0.1 msec steps, but on increasingly short pulse widths)
- As well, more squirts eats up more of the available time with open/close cycles, reducing the dynamic range of the injectors. So users who have chosen their injectors based on the net horsepower may find they are too small at high rpms and loads. This can damage an engine, and must be avoided at all costs, of course.

So more squirts may help. They may not, however. And this can be a bandage for poor idle or accel tuning. So users should start with 2 squirt/alternating for port injection, and tune it as well as they can.
Then, if they can't solve some issues, they should try more squirts. But jumping straight to more squirts is probably not a good idea - most often it just masks the need for a richer idle mixture or more accel enrichment.

Example: Suppose you have a 4 cylinder 4-stroke engine. Then in 720° (two revolutions) you have 4 spark events (at 0°, 180°, 360° and 540°, then starting over at 720°=0°). MegaSquirt can only inject on an ignition event, so 1, 2, or 4 squirts/cycle for this engine.
If you have 2 squirts/simultaneous, you have:

crankshaft degrees	0°	90°	180°	270°	360°	450°	540°	630°	720° =0°	...→
cylinder	1		3		4		2		1	...→
bank 1	inject				inject				inject	...→
bank 2	inject				inject				inject	...→

but with 2 squirts /alternating, you have:

crankshaft degrees	0°	90°	180°	270°	360°	450°	540°	630°	720° =0°	...→
cylinder	1		3		4		2		1	...→
bank 1	inject								inject	...→
bank 2					inject					...→

Both are two squirts per cycle, but alternating only has 1/2 as many total squirts per injector (i.e.,

since only one channel is active per squirt), so the pulse width must be doubled.
4 squirts/simultaneous would look like this:

crankshaft degrees	0°	90°	180°	270°	360°	450°	540°	630°	720° =0°	...→
cylinder	1		3		4		2		1	...→
bank 1	inject		inject		inject		inject		inject	...→
bank 2	inject		inject		inject		inject		inject	...→

There are 4x as many injection events with 4 squirts/simultaneous compared to 2 squirts/alternating, so the pulse width is just 1/4 as long (neglecting the opening times).
The req_fuel calculator will show this - watch the top and bottom numbers as you make changes to the alt/sim and number of squirts. The top number stays the same (it is the 'unadjusted number' for 1 squirt simultaneous), while the bottom 'adjusted' number used for the pulse width calculations changes with the number of squirts and simultaneous/alternating.

- **Engine Stroke** (*EngStroke*) values for engine stroke type are two-stroke or four-stroke. MegaSquirt® uses engine stroke to determine how many degrees are in an engine cycle.
- **Number of Cylinders** is the count of the cylinders on your engine. If you are unsure how many cylinders your engine has, you should not be installing MegaSquirt® on it. This value is actually the number if ignition events per cycle sent to the ignition input on the controller.
- **Injector Port Type** (*InjType*) is used to select the type of injectors that you are using, either *throttle body* (if the injectors spray above the throttle plates) or *multi-port* (if the injectors spray into the intake ports).
- **Number of Injectors** (*NoInj*) is the total number of injectors MegaSquirt® is controlling, whether port or throttle body injection.
- **MAP Type** values for this may be selected from the option menu, and are either 115 kPa or 250 kPa. All Version 2 MegaSquirt® partial kits have the 250 kPa MAP sensor (this is all MegaSquirt® sold in the last few years). The MAP sensor type should be auto-detected from MegaSquirt® EFI Controller, but if it is not, select the right one and hit "Send to ECU".

  You should check that MegaTune reads approximately the correct barometric pressure when no vacuum is applied (i.e. the engine isn't running). Below is a chart of the 'normal' barometric pressures for various geographic elevations. MegaSquirt-II (or MicroSquirt) should generally be within 2 or 3 kPa of the value below for your elevation. Because of the increased accuracy of the 10-bit ADC on MegaSquirt-II (or MicroSquirt), versus the 8-bit ADC of MegaSquirt® EFI Controller, the MAP reading will likely be 1 or 2 kPa higher with MegaSquirt-II (or MicroSquirt). If it is significantly different, check under '*Tools/Sensor Calibration*' and verify that you have values of 9.3 (@ volts) and 260.9 (@ 5 volts) for the MAP sensor.

Barometric Pressure vs. Elevation

Elevation Above Sea Level		Atmospheric Pressure	
Feet	**Meters**	**kiloPascals (kPa)**	***example***
0	0	101.33	New York, Vancouver, Washington
500	153	99.49	Dallas (435 feet), Detroit (585 feet)
1000	305	97.63	Geneva (1230 feet), Kelowna (1129 feet)
1500	458	95.91	Helena (1404 feet), Wichita (1290 feet),
2000	610	94.19	Canberra (1886 feet), Las Vegas (2030 feet), Regina (1893 feet)
2500	763	92.46	
3000	915	90.81	Red Deer (2968 feet)
3500	1068	89.15	Brasilia (3480 feet), Calgary (3750 feet)
4000	1220	87.49	
4500	1373	85.91	Banff (4500 feet)
5000	1526	84.33	Albuquerque (4945 feet), Denver (5280 feet)
6000	1831	81.22	Colorado Springs (5890 feet)
7000	2136	78.19	Mexico City (~7200 feet)
8000	2441	75.22	
9000	2746	72.40	
10,000	3050	69.64	
15,000	4577	57.16	La Paz (13,169 feet), Mauna Kea (~14,000 feet)

Note that weather stations usually report the barometric pressure 'corrected' to read as if 101.3 kPa was the 'normal' for your elevation. Do not expect these reports to correspond to what you get on MegaSquirt® unless you are at sea level.

MegaSquirt-II (or MicroSquirt) has the provisions for a second independent 'realtime' baro sensor. This will update your fueling continuously, which may be helpful if you are climbing the Rocky mountains or running Pike's Pike.

To add an independent baro sensor to your MegaSquirt-II (or MicroSquirt), use an MPX4250AP (the standard MS sensor). Run leads from:

1. sensor pin #1 (signal) to header position X7 on the MegaSquirt® PCB through a 1K Ohm resistor,
2. sensor pin #2 to ground (the non-banded end of D1), and
3. sensor pin #3 to 5 Volts - use the via at the "M" in the copyright notice near the existing MAP sensor.

You can use other 0-5 volt pressure sensors, but you will have to calibrate it using the 'Tools/Sensor Calibration' dialog in MegaTune.

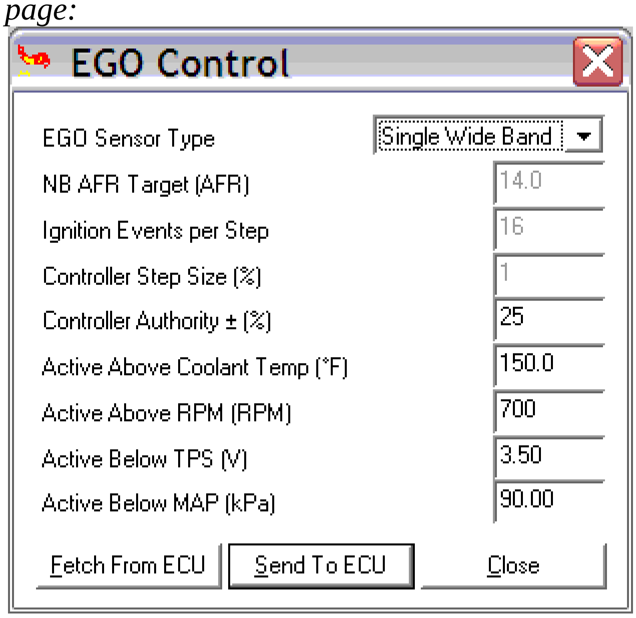
You do not use any caps or protection diodes. The baro sensor can be mounted to the top of the case, above the LEDs near the DB9. It should clear the MegaSquirt-II here, but not by much, be sure to check. You may be able to put it at the other end, if the flyback board is not in the way in your installation.

- **Engine Type** (*OddFire1, OddFire2*) has the options of odd fire or even fire. Odd-fire or even fire does not refer to the firing order, but rather the interval between successive firings.

  So if you have a 4 cylinder, and a spark every 180 degrees, you have an even fire. Almost all 4 cylinder engines are even fire.
  However some 90 degree V6s, some V4s, and most V-Twins (usually motorcycle engines), as well as a few others, have 'odd-fire' arrangements.
  For example, from 1978 to 1984, the GM V6 (200 and 229 cid) had a semi-even fire sequence, with firing intervals of 132°/108°. It is "semi" because the rod journals are offset, but not quite enough to make for even firing intervals. In MegaSquirt® terms, this is an 'odd-fire' engine, because the interval between firing can be either 132° OR 108° degrees.

Note that for release 1.000, odd-fire is only in the code as a placeholder, there is no code supporting it at this time.

On the 'EGO Control' page:



EGO Control

EGO Sensor Type	Single Wide Band
NB AFR Target (AFR)	14.0
Ignition Events per Step	16
Controller Step Size (%)	1
Controller Authority ± (%)	25
Active Above Coolant Temp (°F)	150.0
Active Above RPM (RPM)	700
Active Below TPS (V)	3.50
Active Below MAP (kPa)	90.00

- ***EGO Sensor Type*** (*EgoOption*):
  - **Disabled** If you don't have an oxygen sensor installed, choose '**Disabled**' under ***EGO Sensor Type*** in MegaTune, and MegaSquirt-II (or MicroSquirt) does not use the numbers in the AFR table to adjust the VE table numbers (as shown above) when calculating the pulse widths. There is no feedback.
  - **Narrow Band O2 Sensor** If you have enabled a **narrow band oxygen sensor**, choose '**Narrow Band**' under ***EGO Sensor Type*** in MegaTune, and MegaSquirt-II will try to adjust the amount if fuel injected, up to the limits you specify, to give the oxygen sensor voltage specified in the tuning software. The AFR table is not used, instead a single oxygen sensor voltage target is used.

    There is a bug in some versions (up to 2.36) of the code for narrow band EGO only (more accurately, there is a 'misunderstanding' between MS-II and MegaTune as to what the narrow band curve should look like). The bug doesn't allow the reported EGO voltage to exceed 0.41 Volts, regardless of the actual voltage. To fix it, set the EGO to one of the WB (DIY-WB, for example), press the OK button, and let the 1024 entries burn to MegaSquirt-II (or MicroSquirt). Then shut down both MegaTune and MegaSquirt. Then start them both up again and set the 'calibrate AFR table' to narrow-band - be sure to press 'okay' and let all the values burn - just because it is already highlighted doesn't mean you are done. Cycle MT and MS again, and EGO should work properly.

IMPORTANT NOTE: Do NOT burn tables ('Calibrate AFR Table' or 'Calibrate Thermistor Tables') on a running engine. Even idle is NOT allowed, because these tables ONLY exist in flash, so once a table is erased, there is nothing but garbage in there until it is re-programmed, one word at a time. Until that reprogramming is complete, operating the engine is unsafe.

- **Dual Narrow Band**: The dual lambda sensor feature has been in the MS-II code since V1.0. You connect the second sensor to the ADC6 input with appropriate circuitry and it adjusts the PW2 output independently of PW1. You connect the second sensor to the **JS5** hole (on a V3 main board) - **X7** on a V2.2 main board, duplicating the R10, R11, C10 circuit from the v3.0 PCB in the proto grid area. There is only one calibration because it is assumed you are going to use the same type of sensor on each side. If there is a small difference, you can compensate for it in the separate AFR target tables.
- **Wide Band** If you have a **narrow band O2 sensor OR a wide band oxygen sensor & controller**, choose the appropriate setting under ***Settings/EGO Control*** in MegaTune, and be **SURE** to go to Tools/Calibrate AFR Table and select your controller type. **Make sure to have the MegaSquirt-II (or MicroSquirt) connected and powered up while you do this - the calibration is saved for both MegaTune (on the PC) and to MegaSquirt-II (to which the calibration table is downloaded).** Then MegaSquirt-II (or MicroSquirt) will adjust the amount of fuel injected based on the AFR table until the sensor reports a voltage corresponding to the air/fuel ratio in the appropriate cell of the AFR table (for wide band) or switch point (for narrow band).

  IMPORTANT NOTE: Do NOT burn tables ('Calibrate AFR Table' or 'Calibrate Thermistor Tables') on a running engine. Even idle is NOT allowed, because these tables ONLY exist in flash, so once a table is erased, there is nothing but garbage in there until it is re-programmed, one word at a time. Until that reprogramming is complete, operating the engine is unsafe.

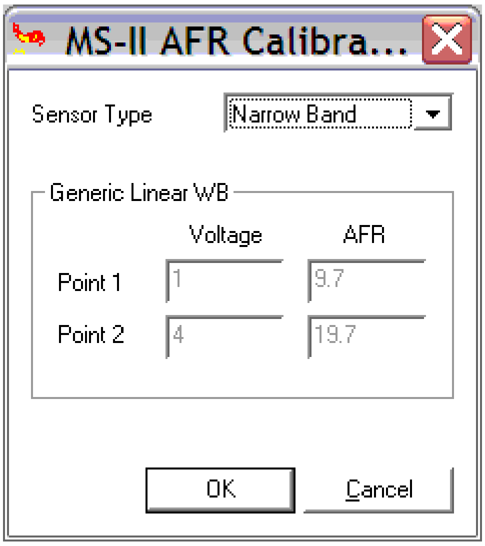
- **Dual Wide Band**: The dual lambda sensor feature has been in the MS-II code since V1.0. You connect the second sensor to the ADC6 input with appropriate circuitry and it adjusts the PW2 output independently of PW1. You connect the second sensor to the **JS5** hole (on a V3 main board) - **X7** on a V2.2 main board, duplicating the R10, R11, C10 circuit from the v3.0 PCB in the proto grid area. There is only one calibration because it is assumed you are going to use the same type of sensor on each side. If there is a small difference, you can compensate for it in the separate AFR target tables.

- **Ignition Events per Step:** (*EgoCountCmp*) This is the number of 'sparks' the engine sees before adjusting the fuel amount based on the EGO sensor feedback. Large numbers make the EGO feedback respond more slowly, but also tend to make it more stable.
- **Controller Step Size (%):** (*EgoStep*) This is the amount the EGO will be adjusted after the number of ignition events specified above. Smaller numbers make the response more stable, but slower.

  Note that the wide band control algorithm does not use the constant step size controller algorithm that was previously used in MegaSquirt® EFI Controller, it now has a P (just proportional) control algorithm, since it gives much more reliable information than the narrow band sensor does. MegaSquirt-II (or MicroSquirt) computes the difference between actual and target AFR, then uses that to do what it thinks is the exact adjustment to pulse width needed to attain the target AFR. So, step size and events don't play a role at all and they are grayed out when the wide band algorithm is selected.

- **Controller Authority ± (%):** (*EgoLimit*) This is the maximum the EGO feedback is allowed to adjust the fuel from the VE table, regardless of the state of the O2 sensor feedback. Large numbers (50% to 80%) are better when the set-up is rough, smaller numbers (5% to 15%) are better when the VE table is 'dialled in'.
- **Active Above Coolant Temp (°):** (*EgoTemp*) This is the lowest coolant temperature at which EGO feedback is allowed to operate. It is necessary to prevent the EGO feedback from working against the warm-up enrichments when the engine needs to be particularly rich while cold.
- **Active Above RPM:** (*RPMOXLimit*) This is the lowest engine speed at which EGO feedback is allowed to operate. It is necessary for those engines that need to be rich while idling.
- **Active Below TPS (V):** (*TPSOXLimit*) This prevents EGO feedback with *either* a narrow-band or wide band sensor from operating at wide-open throttle (WOT), because narrow band sensors are not effective at measuring the rich mixtures required.
- **Active Below MAP (kPa):** (*MAPOXLimit*) This prevents EGO feedback with *either* a narrow-band or wide band sensor from operating at high loads, because narrow band sensors are not effective at measuring the rich mixtures required.

Note that after you select your EGO sensor type, you should go to the '*Tools/Calibrate AFR Table*' dialog and select the sensor output curve.



MS-II AFR Calibra...

Sensor Type: Narrow Band

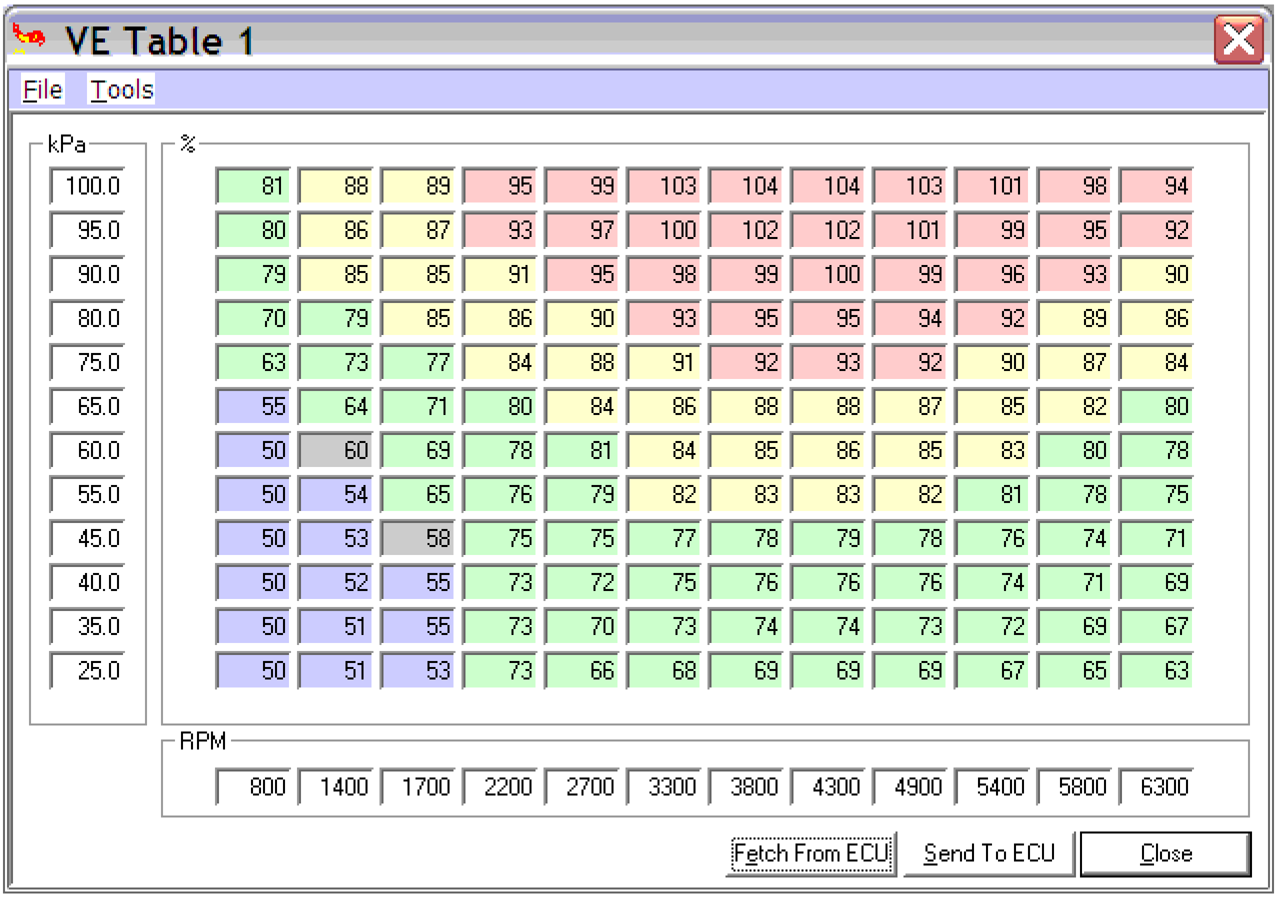
Generic Linear WB

	Voltage	AFR
Point 1	1	9.7
Point 2	4	19.7

For example, if you have a narrow band sensor, select that; or if you have a DIY-WB controller select that, etc. Then click on the '*OK*' button and a file will be create and downloaded to MegaSquirt-II (or MicroSquirt) that indicates the corresponding air/fuel ratio for various sensor/controller output voltages.
You can also create your own custom table by entering two point on the output curve.
There is more information on how to use these settings while tuning in the Tuning section.
IMPORTANT NOTE: Do NOT burn tables ('Calibrate AFR Table' or 'Calibrate Thermistor Tables') on a running engine. Even idle is NOT allowed, because these tables ONLY exist in flash, so once a table is erased, there is nothing but garbage in there until it is re-programmed, one word at a time. Until that reprogramming is complete, operating the engine is unsafe.
Setting the Tables:
Before starting your engine, you need to populate three types of tables: the VE table(s), the AFR table(s), and the ignition advance table.

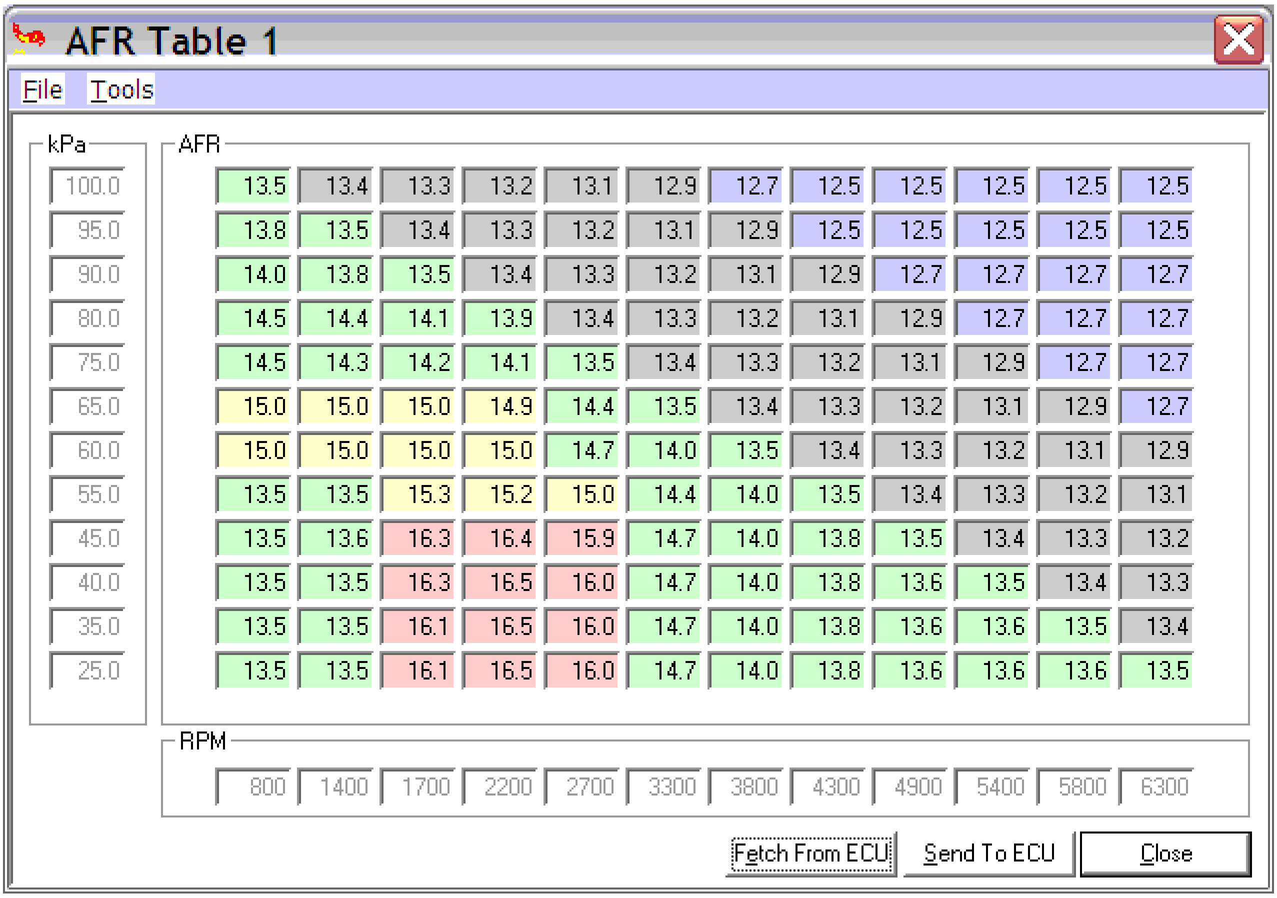
- **VE Table(s)** (*ve_table*): Unless you already have a very good idea what your VE table should look like, use the '**Generate VE Table**' under '*Tables/VE Tables/Tools*' to automatically fill in the VE table(s). You will have to do this twice if you have selected the dual table option above.

  However, if you already have a suitable VE table (from tuning with MegaSquirt® EFI Controller, for example), you can 'Tables/VE Table/File/VE Import' an 8×8 VE table. MegaTune will correctly interpolate it and give you a corresponding 12×12 table.
  So when you are in the table editor, and you do a VE Import, it will automatically interpolate from whatever size the VE table is in the file to whatever it should be in the controller. If you have a good working table in 8×8 MegaSquirt® EFI Controller, just 'VE Export' it, pop in your MegaSquirt-II (or MicroSquirt) and VE Import it. You are warned that this is happening, but other than that it's fully automatic.
  The load and rpm bins are interpolated, thus maintaining their original distribution (if you have tight values down low, and big gaps up top in the old bins, that is obvious in the new ones). The table is then interpolated using the new bins with the same algorithm that MS uses internally (which is also used to place the "spot" on the 3D VE tuning screen).
  This makes transition from MegaSquirt® to MegaSquirt-II much simpler for those who upgrade.



kPa	800	1400	1700	2200	2700	3300	3800	4300	4900	5400	5800	6300
100.0	81	88	89	95	99	103	104	104	103	101	98	94
95.0	80	86	87	93	97	100	102	102	101	99	95	92
90.0	79	85	85	91	95	98	99	100	99	96	93	90
80.0	70	79	85	86	90	93	95	95	94	92	89	86
75.0	63	73	77	84	88	91	92	93	92	90	87	84
65.0	55	64	71	80	84	86	88	88	87	85	82	80
60.0	50	60	69	78	81	84	85	86	85	83	80	78
55.0	50	54	65	76	79	82	83	83	82	81	78	75
45.0	50	53	58	75	75	77	78	79	78	76	74	71
40.0	50	52	55	73	72	75	76	76	76	74	71	69
35.0	50	51	55	73	70	73	74	74	73	72	69	67
25.0	50	51	53	73	66	68	69	69	69	67	65	63

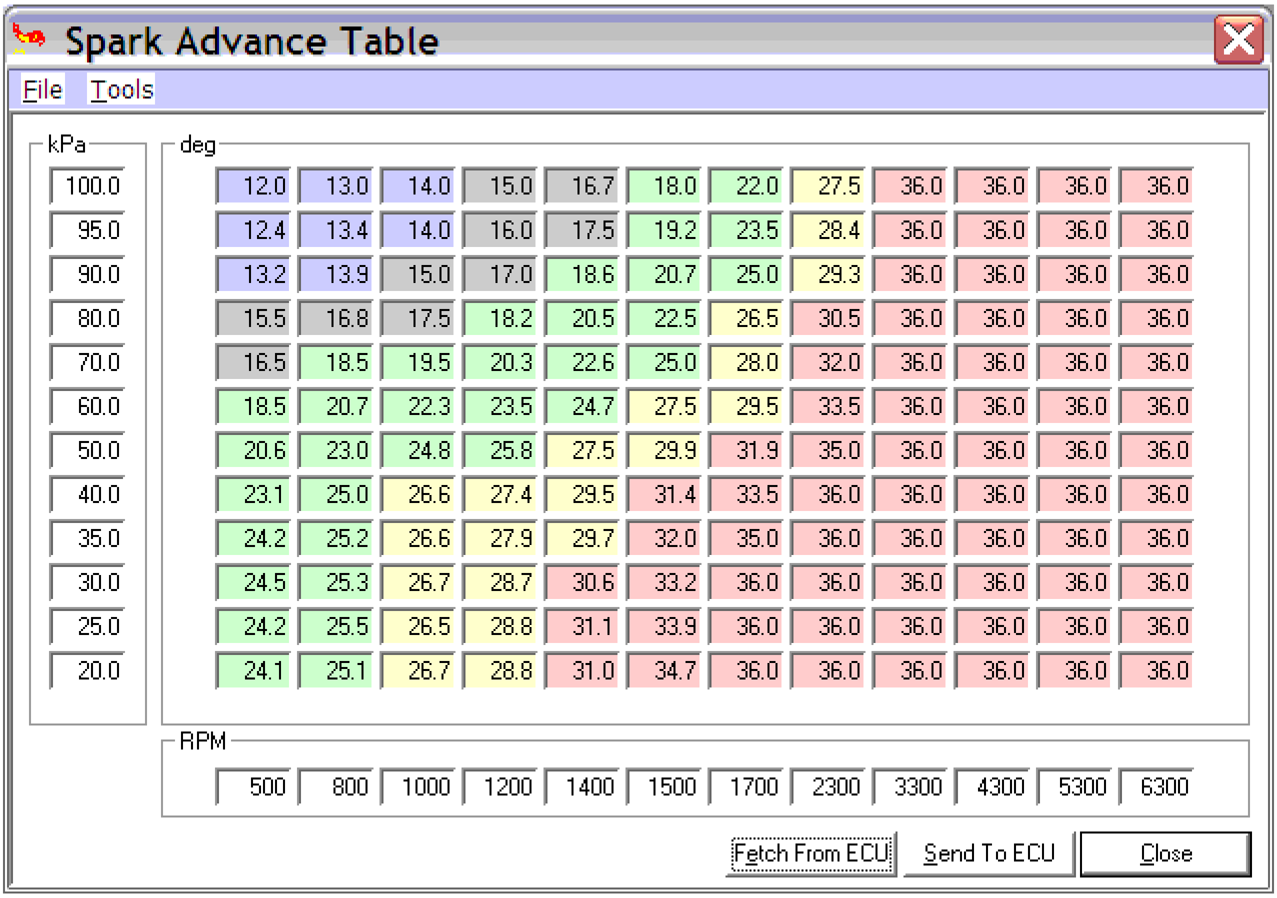
- **AFR Table(s)** (*afr_table*): If you have a wide band sensor, fill in the air/fuel ratio table. Generally, you want it to be lean in the areas where it is lightly loaded and you want best economy. You may be able to run as lean as 17:1 in these areas. At WOT, conventional wisdom is that you want 13.0:1 at peak torque, and 12.5:1 at peak horsepower. Then blend the WOT and economy areas so that there is a smooth transition. You will have to do this twice if you have selected the dual table option above.



kPa	800	1400	1700	2200	2700	3300	3800	4300	4900	5400	5800	6300
100.0	13.5	13.4	13.3	13.2	13.1	12.9	12.7	12.5	12.5	12.5	12.5	12.5
95.0	13.8	13.5	13.4	13.3	13.2	13.1	12.9	12.5	12.5	12.5	12.5	12.5
90.0	14.0	13.8	13.5	13.4	13.3	13.2	13.1	12.9	12.7	12.7	12.7	12.7
80.0	14.5	14.4	14.1	13.9	13.4	13.3	13.2	13.1	12.9	12.7	12.7	12.7
75.0	14.5	14.3	14.2	14.1	13.5	13.4	13.3	13.2	13.1	12.9	12.7	12.7
65.0	15.0	15.0	15.0	14.9	14.4	13.5	13.4	13.3	13.2	13.1	12.9	12.7
60.0	15.0	15.0	15.0	15.0	14.7	14.0	13.5	13.4	13.3	13.2	13.1	12.9
55.0	13.5	13.5	15.3	15.2	15.0	14.4	14.0	13.5	13.4	13.3	13.2	13.1
45.0	13.5	13.6	16.3	16.4	15.9	14.7	14.0	13.8	13.5	13.4	13.3	13.2
40.0	13.5	13.5	16.3	16.5	16.0	14.7	14.0	13.8	13.6	13.5	13.4	13.3
35.0	13.5	13.5	16.1	16.5	16.0	14.7	14.0	13.8	13.6	13.6	13.5	13.4
25.0	13.5	13.5	16.1	16.5	16.0	14.7	14.0	13.8	13.6	13.6	13.6	13.5

Note that you can export or import VEX files into any of these tables. Use the '*Table/XXX Table/Files/VE Table Import*' to import them, '*Table/XXX Table/Files/VE Table Export*' to export them. These are all saved in the VEX format, so be sure to give your saved files descriptive names. The table will import VEX files of different sizes and automatically adjust them to match the current table size.

- **Ignition Table** (*adv_table*): In general, you want:
  - Low MAP (low engine load) = more spark advance
  - High MAP (high engine load) = less spark advance
  - Low CLT (cold engine) = more spark advance
  - High CLT (warm engine) = less spark advance
  - Low RPM = less spark advance
  - High RPM = more spark advance



kPa	500	800	1000	1200	1400	1500	1700	2300	3300	4300	5300	6300
100.0	12.0	13.0	14.0	15.0	16.7	18.0	22.0	27.5	36.0	36.0	36.0	36.0
95.0	12.4	13.4	14.0	16.0	17.5	19.2	23.5	28.4	36.0	36.0	36.0	36.0
90.0	13.2	13.9	15.0	17.0	18.6	20.7	25.0	29.3	36.0	36.0	36.0	36.0
80.0	15.5	16.8	17.5	18.2	20.5	22.5	26.5	30.5	36.0	36.0	36.0	36.0
70.0	16.5	18.5	19.5	20.3	22.6	25.0	28.0	32.0	36.0	36.0	36.0	36.0
60.0	18.5	20.7	22.3	23.5	24.7	27.5	29.5	33.5	36.0	36.0	36.0	36.0
50.0	20.6	23.0	24.8	25.8	27.5	29.9	31.9	35.0	36.0	36.0	36.0	36.0
40.0	23.1	25.0	26.6	27.4	29.5	31.4	33.5	36.0	36.0	36.0	36.0	36.0
35.0	24.2	25.2	26.6	27.9	29.7	32.0	35.0	36.0	36.0	36.0	36.0	36.0
30.0	24.5	25.3	26.7	28.7	30.6	33.2	36.0	36.0	36.0	36.0	36.0	36.0
25.0	24.2	25.5	26.5	28.8	31.1	33.9	36.0	36.0	36.0	36.0	36.0	36.0
20.0	24.1	25.1	26.7	28.8	31.0	34.7	36.0	36.0	36.0	36.0	36.0	36.0

Making a Spark Advance Table

You can get a base starting point for the VE table by using the 'Tables/VE Table/Tools/VE Specific/Generate Table' function of MegaTune, using the peak horsepower and torque figures for your engine.
For timing, we don't have a generator written yet. The basic principles are to determine a maximum advance for your engine and work backwards from there with heuristics ('*rules of thumb*'):

- older engines (1960s up to 1990 or so) with two valves - max advance = 36°
- newer two-valve engines - max advance = 30°
- three or four valve engines - max advance = 26°

then adjust for bore size:

- under 3.5" (89mm) - subtract 3°
- between 3.5" and 4.000" (101.6mm) - no adjustment
- over 4.001" (+101.6mm) - add 3°

then adjust for the fuel:

- regular - subtract 2°
- mid-grade - subtract 1°
- premium - no adjustment

That gives us a maximum advance figure. It you have an aftermarket combination with a good squish area and optimized quench, subtract another 2°. If you have a flathead, add 3° or 4° or more.

We will use this to fill in the table at 100 kPa from 3000 rpm to the redline.

From idle to 3000 rpm, we want the advance (@100kPa) to increase fairly linearly from the idle advance to the maximum advance. idle advance is really a matter of tuning, but assume 8° to 16° in most cases, with stock engines being on the lower end, and 'hotter' engines being on the upper end.

So if we have a hot engine with 36° maximum advance and 16° idle advance (at 800rpm), the spark table might look like this for 100kPa:

```
100    16°  16°   18°   24°   28°   36°

rpm    600  800  1000  1500  2000  3000
```

Below 100 kPa, we add 0.3° per 1 kPa drop. So for example, if our total spark at 100kPa and 4000 rpm was 36°, the advance at 50 kPa would be:

36° + 0.3° x (100-50) = 51°

and the advance at 45 kPa and 800 rpm would be:

16° + 0.3° x (100-45) = 32.5°

However all of these would need to be tuned, and it often helps idle stability to limit the advance at idle to under 20°.

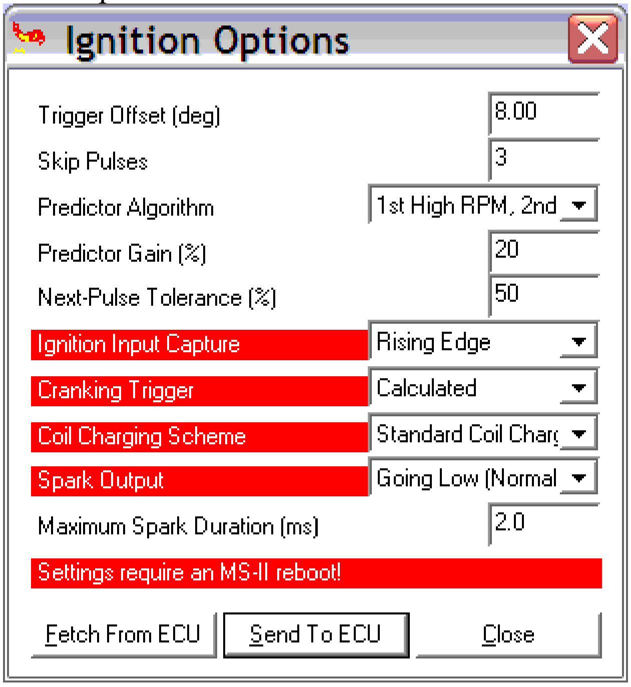
Finally, for boosted engines, you subtract 0.3° of advance for every kPa above 100 (it's not a coincidence that this is the same factor as for the 'vacuum' adjustments). Because 101.3kPa=~14.7psi, this works out to ~2° per pound of boost. It is often the case that you want to limit the retard under boost as well, typically so that it takes out no more than about ½ of the maximum advance at 100 kPa.

None of these will give you the 'right' values for your engine though, and like the VE table calculator, are just a relatively safe starting point. They should be somewhat closer than starting with an empty table, though!

Advance numbers can be specified in tenths of a degree ($^{1}/_{10}$°)

Setting the Ignition Options

MegaSquirt has a number of options:



Ignition Options

Trigger Offset (deg)	8.00
Skip Pulses	3
Predictor Algorithm	1st High RPM, 2nd
Predictor Gain (%)	20
Next-Pulse Tolerance (%)	50
Ignition Input Capture	Rising Edge
Cranking Trigger	Calculated
Coil Charging Scheme	Standard Coil Charg
Spark Output	Going Low (Normal
Maximum Spark Duration (ms)	2.0

Settings require an MS-II reboot!

If you are not entirely sure about your ignition settings, check the supported list for the settings you should use:

- Distributor Pickups
- Ignition Timing Algorithm
- GM HEI
- GM DIS
- Ford EDIS
- Ford TFI
- Bosch 0 227 100 124
- MSD 6A
- Direct Coil Control

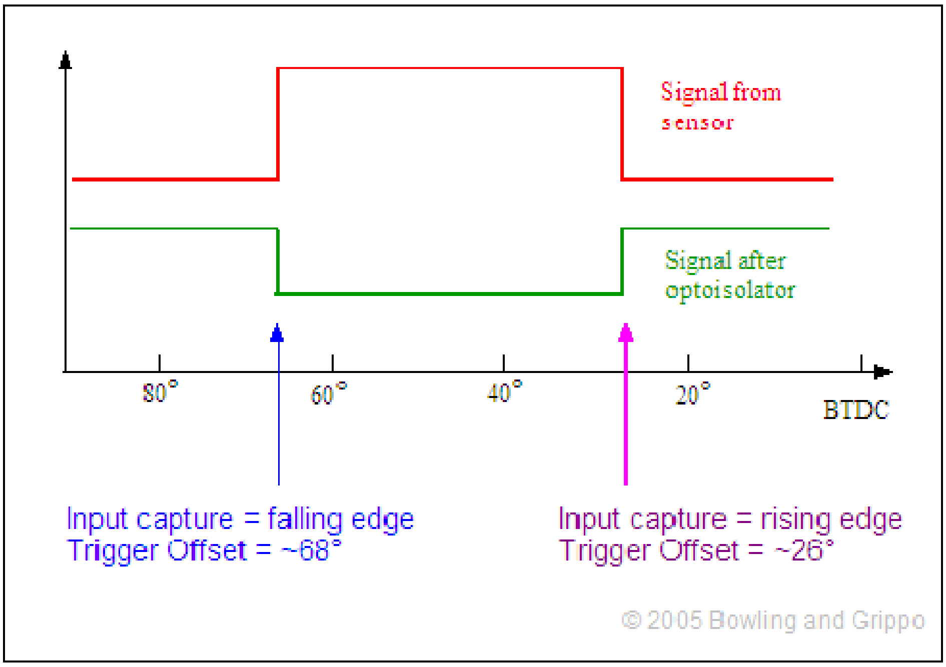
If your module isn't on the 'officially supported' list yet, you'll need to understand the relationship between the trigger offset, the input capture (falling or rising edge), and the spark out (going high or going low). There are some tip on setting these parameters in the tuning section of the MS-II manual.
In the 'Settings/Ignition Options' dialog you can select the:

- **Trigger Offset (deg)** (*adv_offset* in the code) is the advance before (or after) top dead center (BTDC) that the engine gets in it's signal from the engine's variable reluctor or Hall sensor. In many cases, this will be used as the 'base timing' for cranking as well as if the module loses it's connection the the ECU.

  For example, with the 7/8-pin HEI or Bosch 0 227 100 124 modules, the '**trigger offset**' in MegaTune is:

  - HEI: the number of degrees before TDC at which the VR sensor output goes from negative to positive, the HEI module converts this to a falling edge, and and the falling edge of the square wave is sent from the 7/8-pin HEI module to MegaSquirt-II (which is then inverted again by the optoisolator or VR circuit on a V3 main board (***not** MicroSquirt*) to become a rising edge at the processor).
  - Bosch 124: the number of degrees before TDC at which the Hall, optical, or points output goes from ground to positive, and the rising edge of the square wave is sent from the 7/8-pin module to MegaSquirt-II.

  This tells MegaSquirt-II where the crankshaft is positioned so that timing advance can be calculated appropriately. (Note that since the optoisolator (U4) inverts the trigger signal, you specify 'rising edge' for the 'Input Capture' in MegaTune, which refers to the signal *at the processor*.) Positive trigger offsets are used to specify the number of degrees **before** top dead center (BTDC), negative numbers are used for triggers that occur after top dead center (ATDC).


Signal from sensor
Signal after optoisolator
80°
60°
40°
20°
BTDC
Input capture = falling edge
Trigger Offset = ~68°
Input capture = rising edge
Trigger Offset = ~26°
© 2005 Bowling and Grippo

- **Skip Pulses** (*no_skip_pulses*) is the number of ignition pulses at start-up that MegaSquirt-II (or MicroSquirt) uses to calculate the rpm *before* sending calculated advance signals.
- **Predictor Algorithm** is the scheme used to anticipate the amount of time before the next TDC event.

First, the line of code that uses the prediction is:
dtpred += (inpram.Dtpred_Gain * (long)(dt3 - dtpred_old) / 100);
The Dtpred_gain value is the gain percentage read in (input in MegaTune above). The dt3 variable is the just-computed time delta. The drpred_old is the last delta time value.
In our case here we control ignition, which then causes the crank acceleration. So we predict the point to fire, and when the spark occurs the crank gets the "kick" to accelerate.
So ... if we have two tach points it is possible to predict into the future with simple "last interval" derivative (compared to higher derivatives) since the change in crank angular velocity is caused by our ignition event. After the explosion the crank accelerates, but hopefully we get another tach input in order to build up a new time derivative to predict where to place the next spark event.
All of this excludes changes in crank angular velocity due to compression, load variations, moments of inertia, etc. And the ignition event and subsequent expanding gases are not an infinite, short step function impulse on the crank (look up mass fraction burned and Weibe function in Heywood).
But this does illustrate the "causality" of the whole crank position problem - you do not know exactly where the crank is positioned until you get the next position pulse event. At least to the positional error of the tach circuitry. After this point the position is just a prediction w/ associated prop error until the next pulse comes in...
There is a "hidden" variable () within the code that simply subtracts the last prediction from the current tach pulse update. In other words the code uses a derivative to compute when the next tach pulse comes in, then later on the actual tach pulse comes in. The error in this is simply the subtraction. See the line of code:

```
outpc.spare[0] = (int)((outpc.rpm * (dt3 - dtpred_old)) / 16667); // degx10
```

What this is doing is taking the current delta_t based on the latest tach update and subtract the previous delta_t value (dtpred_old). This value is multiplied by the current RPM and divided by 16667 in order to get the variable into degrees crank * 10. So the result is in crankshaft degrees, and shows the error in the last prediction compared to the current updated prediction.

The result is placed into the spare output runtime variable. Now this variable is not defined in the .ini file so it would have to be placed there. It appears after the egoV2 variable, see the runtime variable list in the code and correlate with the ini file. Be sure to note that it is x10 units. Useful is putting this in the gauge list and the datalog.

Now - some warning on this, since this variable can be easily misinterpreted without thinking what it represents. First, note that it will always be in error whenever there is an acceleration event. The MS code does not have a psychic code mod built into it (yet) so the acceleration event has to be sensed via the tach input stream. So there will be places where the last delta_t does not correlate with the new delta_t with acceleration/decel, and there will be corresponding error. This is why it is a good idea to place your ignition events somewhere close to a point where the latest update occurred.

Second, even with a steady state where the tach pulses are the same, like from the stimulator, there will be a slight error in the derivative calculation. The one thing we discovered from this testing was with high order derivatives that having the pulses the same was causing a little introduced error term. People who have had calculus know exactly what I mean. If the first derivative does not change (i.e. the tach pulses are the same time period) then the higher derivatives will be zero - there is no rate-of-change of RPM. The way the code determines the numerical derivatives is by using time periods, and the calc method was introducing a small amount of error in exact time periods. So if one monitors the prediction error variable they will note more degree jitter when there is no change in RPM (magnitude is less than a degree) for higher derivative use. To simmer down this effect Al has put a check in his (soon to be released) code where the higher-order derivatives are not applied to the calculation if there has been minimal change in tach pulse spacing.

Third, note that this variable represents the worst-case error. Its occurs at the tach update point so the time-propagated period error is the worst here. Remember you are (or should be) firing the ignition even before this point so the time-propagated error effect is less than this magnitude.

In my mind the proper use of this variable is to monitor it on a vehicle while trying the different derivative levels. The goal is to obtain the least degree deviation for accel/decel. The physics should be such that higher derivatives should help predict accel events better than simple last-interval calculation since it takes into account the rate-of-change in RPM over many delta_t intervals - but then there will be a greater error (overshoot) during the beginning and end of an accel/decel trend compared to the last interval. Datalog and compare to see what works best for you.

Again - if you use this variable then be sure to interpret what it represents in a proper manner. This is one of those numbers where someone sees a 5 degree crankshaft error and they then tell the world that "the code has a 5 degree error" without putting it into proper constraints. Now, if there is always a 5 degree error, then there is a real problem....

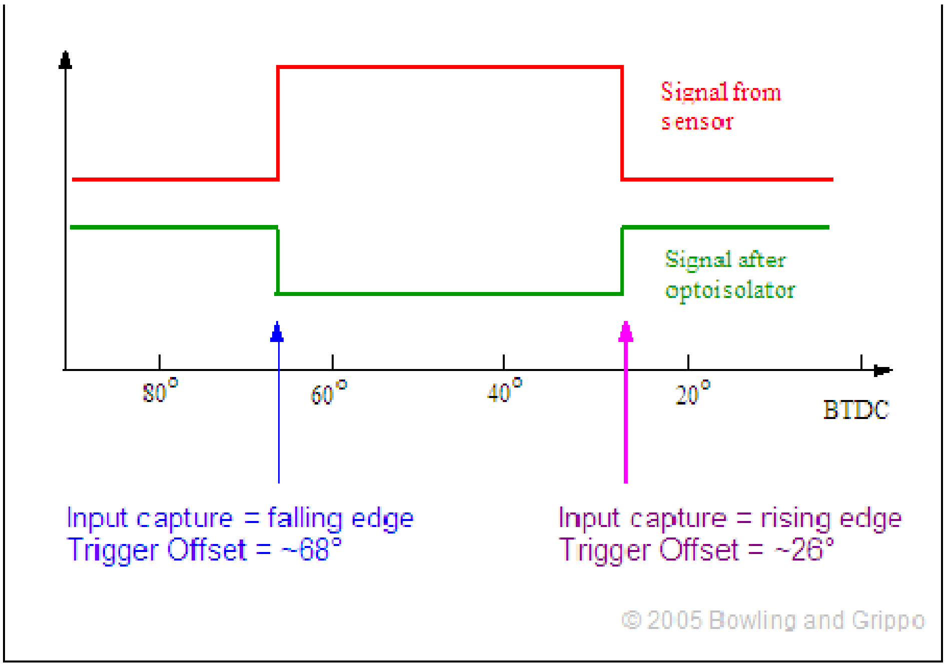
- **Predictor Gain (%)** (*Dtpred_Gain*) is part of the Kalman filter random error correction, where the change in the predicted interval between the last and next ignition event is proportional to difference between current dt3 and last predicted dt3. The Predictor gain (%) is the multiplier for the difference to apply to the new predicted time. So:

  $$\mathbf{dt_{pred} = dt_{pred} + ([Predictor\ Gain] \times (dt3 - dtpred\_old) / 100)}$$

  What the gain value does is choose (add in) between the latest calculated time delta (i.e. 0%), or use up to 100% of the last calculated time interval. This is known as a predictor-corrector, meaning you use the last calculated time interval, or a blend of this value (corrector) versus the predicted value of the time delta.
  In pretty much all vehicle setups here we want to use the last hardware tach input pulse and use the delta time between this and the previous pulse. It is desirable to use the hardware tach inputs to generate the derivative prediction (i.e. into the future for when the ignition event needs to occur). This is the best information that we have and is better than blending this in with the previous derivative predictions.
  This predictor-corrector is a very simplistic form of a Kalman filter. A Kalman filter can take many sources of input data, be it a running prediction or hard input data (with different, known error sources) and generate the absolute best prediction. The key here is multiple sources of input data with their own known error distribution.
  In our case, we have one input source, namely the tach hardware input, and the assumption is that each and every pulse has the same magnitude error - nothing differentiates one tach pulse for having less error than another, they have equal error.
  The default predictor gain is currently 20% (v2.36 release code), but will eventually be changed to 0% as new code is released. Originally the default was 20% because it gave a small improvement in simulation studies, but it appears to hurt more than help in the real world, so the latest code defaults this to 0%.
  So - when would it make sense to try to use a gain setting other than zero? Lets say you are using a VR sensor and you are triggering off of both the rising and falling zero-crossing waveform. In one direction, the VR waveform transitions rapidly thru zero, which is the direction normally used in VR triggering - and this yields a sharp, deterministic transition point. In the other direction, the signal may not be as sharp vertically and there may be a indeterminate trigger point (this is easily visualized from VR voltage output traces). So, half of the VR triggers are really accurate, and the other 50% have jitter. So this case it may work better to use some of the running predictions in the final delta computation as opposed to 100% of the input VR triggers. Note that there are many factors involved. And to implement Kalman properly there should be a corresponding error sigma assigned - the accurate VR triggers have less error than the ones with jitter. And when acceleration/decel occurs (any angular rate-of-change), a hardware input tach is the only way to know this happened, previous predictions are not very useful (unless the hardware tach had positional jitter...).
  For those interested, one of the absolute best (simplistic) descriptions of the Kalman filter is by Peter Maybeck - see the "Simple example" text in:
  www.cs.unc.edu/~welch/media/pdf/maybeck_ch1.pdf

- **Next Pulse Tolerance (%)** (*PulseTol*) is the tolerance during which the next pulse is not allowed to count as a 'true' pulse, and is counted as a false trigger. So if the time between the last two events is X, then if then next pulse occurs within X-PulseTol, it is rejected as a false trigger. If a pulse is NOT received after X+PulseTol, it is assumed to have been missed. This only occurs after the first few pulses (as specified in the Skip Pulses).

- **Ignition Input Capture** (*ICIgnOption* Bits 0-3) is the ignition input signal event that should signal the base timing (Advance Offset) has been achieved.


Signal from sensor
Signal after optoisolator
80°
60°
40°
20°
BTDC
Input capture = falling edge
Trigger Offset = ~68°
Input capture = rising edge
Trigger Offset = ~26°
© 2005 Bowling and Grippo

It is important to note that this is the input signal as seen by the processor. In typical applications, you will be running the input signal from pin 24 through the optoisolator (4N25, U4) which **inverts** this signal, so you need to use the opposite of what the sensor/module puts out.

For example, the GM 7-pin HEI module takes the variable reluctor signal, and generates a 'going low' pulse when the base advance point is reached. In this case, the 'Rising Edge' is chosen, as the signal 'going low' (i.e., falling edge) is used as a trigger, but this is inverted by the 4N25 optoisolator (or VR input), becoming the rising edge of the square wave at the processor.

If your are using the VR sensor circuit on the V3.0 main board, with the

- If you are using 'standard input' to the processor (TSEL to VROUT) then the signal goes high on the negative-going zero transition, and low on the positive-going zero transition (i.e., it is 'inverted').
- If you are using the 'inverted input' (TSEL to VROUTINV), then the signal goes; low on the negative-going zero transition, and high on the positive-going zero transition.

For example, with a GM 7/8 pin HEI VR sensor, which gives a positive-going transition as a trigger (most, but not all, 4 pin HEI modules use a negative-going VR trigger), attached to the V3.0 main board VR circuit, you would set the ignition input capture to 'Rising Edge', which corresponds to the positive-going zero transition for the VR sensor.

Compared to V3 main board, the MicroSquirt and the MS-II/Sequencer controller:

- opto (aka. Hall) input is the same (use same edge as a MS-II/V3 main board for triggering),
- VR input is inverted (use opposite edge for trigger),
- the ignition output is the same compared to the V3 main board. Use the same spark output setting as MS-II (i.e. 'going high' and 'going low').

- **Cranking Trigger** is used to determine when to fire the spark plugs between 50 and 300 rpm (by default) The cranking trigger can be calculated, trigger return, or trigger rise.:

1. ***Calculated***: this means that MegaSquirt-II fires the spark the same way it does when it is above the cranking rpm (CRANK_RPM, default is 300 rpm). That is, it makes a prediction of when the next tach pulse will arrive, based on the previous pulse times, and sets timing based on this and the values in the spark table. If the starting tach pulses are erratic or noisy, this can make the spark timing erratic, resulting in poor cranking behavior.
2. ***Trigger Return***: The crankshaft speed varies quite a bit at cranking speeds. It is affected by the compression and power strokes more than it would be during normal running. This makes it difficult to predict when the next TDC will occur from the last TDC event. As a result, some OEMs have devised a 'trigger return' wheel. This is a wheel with broad teeth, such that during cranking the spark is fired on one edge of the tooth with no computer advance, only the advance set by the width of the tooth edge; when rpm goes above cranking, it switches to firing on the opposite edge of the tooth with calculated advance. This edge typically has a smaller built in advance, possibly 0°. This latter advance should be what is entered in the trigger offset. This option should NOT be selected unless you have a trigger wheel made for this purpose and understand how it is set up.
3. ***Trigger Rise***: is the best all around selection for most installations. It fires the spark plugs when the 'normal' trigger is seen (i.e., the signal that comes at the trigger offset). Use this if it you are able to set up your tach input to MegaSquirt-II so that it comes in at a good timing advance for starting, generally something like 5-12° BTDC. Then MegaSquirt-II (or MicroSquirt) will fire the spark at the exact instant the trigger signal comes in and you will have consistent advance during cranking (at the trigger offset), no matter how uneven the engine speed.

- **Coil Charging Scheme** (*ICIgnOption* Bits 4-7) is used to specify whether the spark occurs on the falling edge of the output signal ('Standard Coil Charge' such as HEI) or the spark timing is generated by the ignition module (EDIS, etc.) In a standard system, the coil 'sparks' when the current to the coil is interrupted. This is the 'Standard Coil Charging' option and is used with the GM 7-pin HEI module, for example. With the EDIS system, and others, though, it is the ***length*** of the pulse that determines the timing, not the actual timing of the pulse, so you would select the EDIS option.
- **Spark Output** (*spkout_hi_lo*) 0 equals spark when low (ground). Production MegaSquirt-II's do not have a transistor for ignition control on the daughter card. This transistor inverted the processor output on the beta units. As a result, if you are using a production MegaSquirt-II and NO transistor, the Coil Charging Scheme needs to be reversed.

  You should set the spark output such that no current flows through the coil when the rpm=0. To do this, replace the coil with an LED tester (see step #6).

Setting the spark output one way should result in the LED being off with no rpm - that's the way you want it. If It is on, the coil is flowing current all the time, and it will overheat, and the ignition module (or VB921 or BIP373) may overheat too.
If you are using the 'high current ignition driver' on the v3.0 main board to drive a coil directly, you need to:

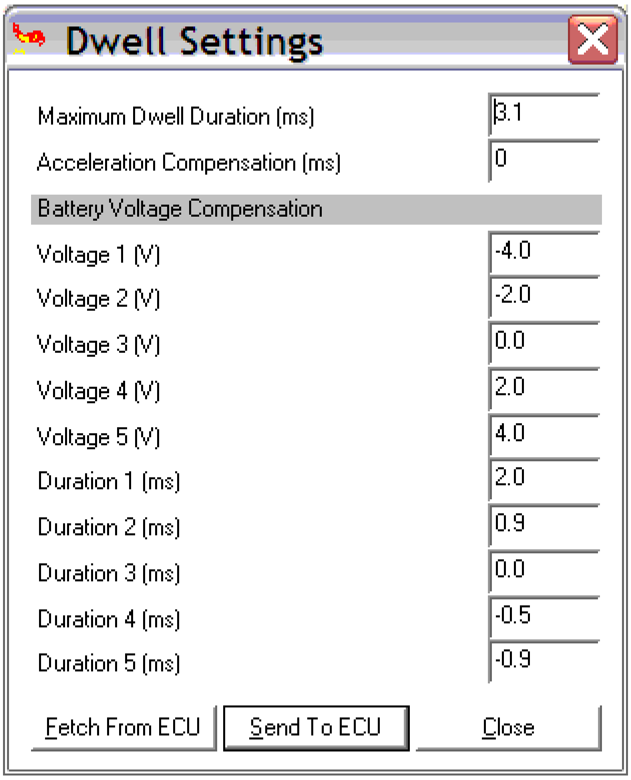
0. Run 12 Volts from the switched side of the ignition (on a relay) to the positive terminal of the coil,
1. Run a wire from the negative terminal of the coil to pin #36 of the MegaSquirt® DB37,
2. Set the coil charging scheme to '**Going High (Inverted)**'.

- **Maximum Spark Duration (ms)** (*max_spk_dur*) is the amount of time MegaSquirt-II (or MicroSquirt) tries to wait before starting another charging cycle. MS-II tries to fit the dwell time + max. spark duration into the time between sparks. If it can fit them, then the full dwell time is used. If the time is too short (because the rpm is high) to fit both the dwell and the max.spark duration, then both are shortened proportionally.

  If the max spark duration is too long, however, then the actual dwell value gets shortened at a lower rpm, limiting the spark energy at higher rpms. Typical values are between 1.5 and 2.0 milliseconds.

Dwell Setting

Dwell is the length of time the coil charges to make each spark. It has to be long enough to make a decent spark, but too long heats the coil unnecessarily. It is in milliseconds, since the time between spark is typically a few dozen or less milliseconds. For example, for a V8 at 600 rpm, the time between sparks is 25 milliseconds. At 6000 rpm the time is just 2.5 milliseconds.



Maximum Dwell Duration (ms)	3.1
Acceleration Compensation (ms)	0
Battery Voltage Compensation	
Voltage 1 (V)	-4.0
Voltage 2 (V)	-2.0
Voltage 3 (V)	0.0
Voltage 4 (V)	2.0
Voltage 5 (V)	4.0
Duration 1 (ms)	2.0
Duration 2 (ms)	0.9
Duration 3 (ms)	0.0
Duration 4 (ms)	-0.5
Duration 5 (ms)	-0.9

- **Maximum Dwell Duration (ms)** (*max_coil_dur*) is the longest period the coil is allowed to charge. The coil charges when a current creates a magnetic filed in the coil (when the current stops and the field collapses, we get our spark). Our goal is to maximize the magnetic field energy, while minimizing the heat build-up. If you charge for too long you can burn out your coil, too short and the spark may be weak. Normally this value is between 2.0 and 4.0 milliseconds. For the GM HEI 'in-cap' coil, a value of 3.5 milliseconds is typical.

A couple of points are that few automotive coils are designed for more than about 6-6.5 Amps, and reduced coil resistance has been used to shorten dwell times as engine rpm limits have risen over the years (shorter dwells means higher rpms before the dwell is cut back), so as a result most coils are between 2.0 and 4.0 milliseconds, with a general trend towards shorter times for more recent designs and longer times for older designs (wasted spark and coil on plug can be different, of course, as they fire less often and thus don't get cut back until a much higher rpm).
For example, the older 7-pin HEI coil uses 3.5 milliseconds dwell, the newer 8-pin HEI external coil uses 2.5 milliseconds of dwell. Note that times longer than 4.0 milliseconds would mean the dwell is being reduced at as little as 2500 rpm, meaning longer dwell settings won't make the spark 'hotter'.
MegaSquirt-II attempts to set up a charge/ discharge cycle that charges the coil for the specified charge time, discharges it for at least the specified spark duration time, and then remains in the discharge state until it is time to start charging for the next cylinder. At high rpm, there may not be time to fit in the specified charge + discharge times. In this case, MegaSquirt-II (or MicroSquirt) scales down both the charge (max. dwell duration) and discharge (max. spark duration) times proportionally so they just fit within the time between tach pulses.
Note that this is all to do with physics and the properties of inductors, and not related to the VB921 (which does have a current limit of ~6.5 Amps as well)
There are four ways to set the dwell:

1. **Calculate it**: If you know the DC resistance and inductance of the coil's primary circuit (either from a spec sheet or from measuring these with an LCR meter), you can use the following formula for an idea inductor to calculate what the dwell should be. The formula for the time it takes for an inductor to charge up to a certain current (assume t=0, I=0) is:

   $$\mathbf{T = (-L/R) * \ln(1 - (R * I / E))}$$

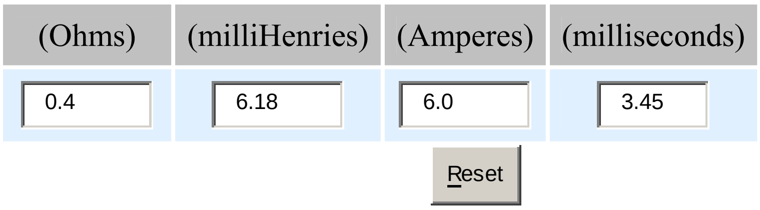
   Where:

   - T = time (seconds)
   - L = inductance (Henrys)
   - R = resistance (Ohms)
   - E = voltage (Volts)

   *(**ln** is the 'natural logarithm, often available as 'LN' on calculators)*

   To save a bit of calculation, you can enter your coil resistance, inductance and current in the form below and press the "Compute Dwell" button (for the dwell at 12 Volts).

Resistance	Inductance	Current	Dwell



(Ohms)	(milliHenries)	(Amperes)	(milliseconds)
0.4	6.18	6.0	3.45

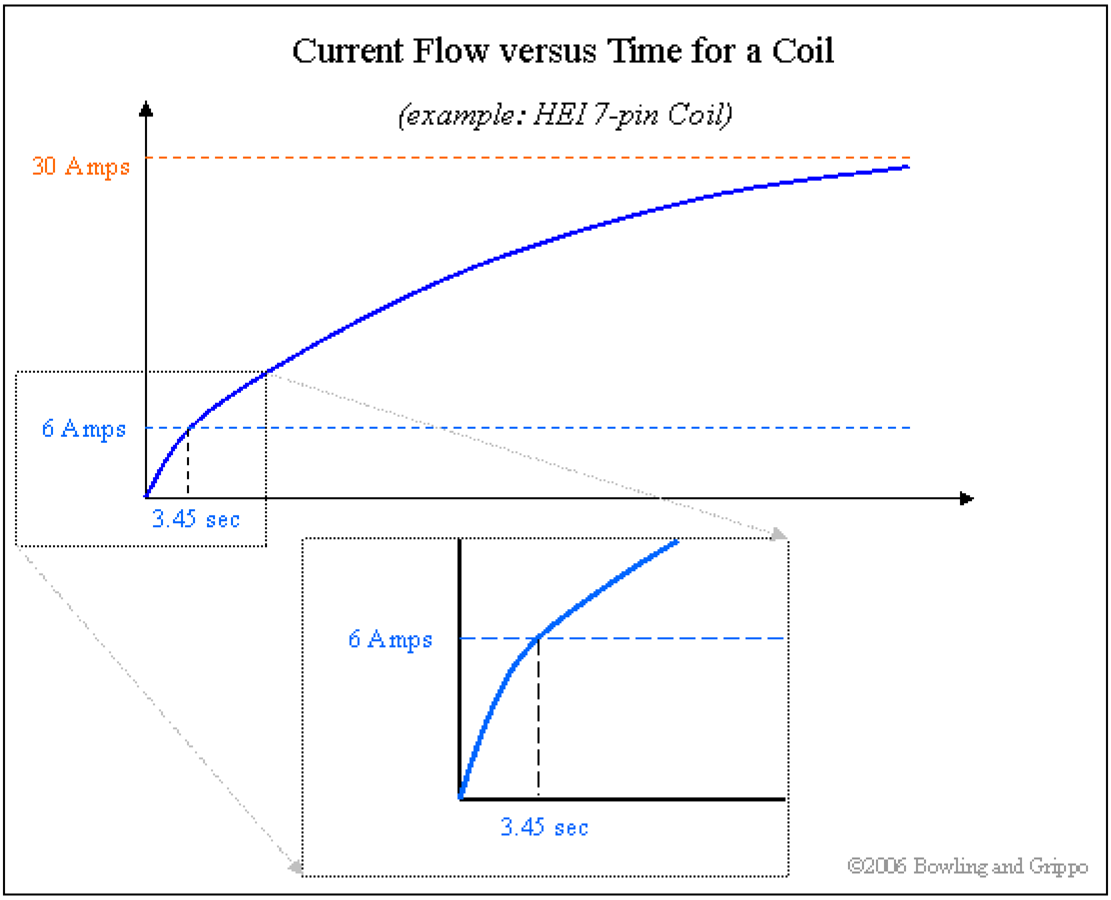
In one case, the values for a 7-pin HEI large cap set-up with red/yellow leads are:

- L = 6.18 milliHenrys
- R = 0.4 Ohms
- E = 12.0 Volts
- I = 6.0 Amps

The values give a calculated time (T) of 3.45 milliseconds.
In the car, it was found that 3.4 worked, but it was set to 3.5 to be sure of an optimal spark under all conditions. At 3.3 or less, this set-up would generate some misfires, especially under 1000 rpm. Note that the 'full DC current flow' would be 12/0.4 = 30 Amps.
The current creates a magnetic field ("charging the coil"), and a change in magnetic field creates a current (we use that to make the spark). In order for the current to increase, it must increase the magnetic field. But if the magnetic filed is already saturated, the current can't increase (without wasting the same amount of energy as heat), so the current increases more slowly (and makes the coil hot). So typically you see a quick rise while the field builds, then a fairly suddenly change in the rate of rise (but still rising) when the field is saturated.
If you plot some values from the equation below (for the 7-pin HEI example) you will see this effect:

Amps	Time (milliseconds)
1	0.5238
2	1.0659
3	1.6278
4	2.2109
5	2.8169
6	3.4476
7	4.1051
8	4.7919
9	5.5106
10	6.2644
15	10.7091
20	16.9736
25	27.6827

Note that it takes 6.3 milliseconds to reach 10 Amps, but 17.0 (2.7x) to reach 20 Amps. Similarly, the coil reaches 6 Amps in 3.45 milliseconds, but takes 8 times as long to reach 4.16 times the current (25 Amps).



If the current was left to build, it would eventually reach the full DC value - if the coil didn't burn up first, of course. *Note that the approach to 30 Amps is 'asymtotic', meaning it never quite gets there and you can't enter 30 Amps in the above example, you will get an error trying to find the natural logarithm.* The magnetic field strength would not increase significantly though, and the spark would not be stronger.

2. **Oscilloscope**: If you have access to an oscilloscope and have installed R43 on a V3 main board (and you are using the VB921 direct coil driver), you can use the voltage drop across this 0.01 Ohm resistor to monitor the current flow through the coil. You are aiming to keep the dwell down to the point of where it just starts current limiting, which you will see as a 'bend' from a rising voltage towards a flat line in the voltage trace on the scope..The time at which that bend occurs should be the dwell point (though this is subjective). That tells you what the max dwell is for the VB921, but this isn't necessarily optimal for the coil.

   To set the proper current limit for the coil, you can measure the voltage across R43 (with a scope, measuring the peak voltage at the top of the rise = peak current) and divide by R43's resistance (.01 Ohm) that's the current. For example, on an in-cap HEI coil and a dwell of 3.5 milliseconds, you would get:

   1.9 division * 0.020 Volt/div / 0.01 = 3.8 Amps

This works well in the car. (It takes about 5.5 milliseconds dwell to reach the current limiting of the VB921 with this coil.) The way to set the dwell is to measure this current and set the dwell to the point at which the current reaches the 'proper value' for the coil (from the spec sheet, etc.). Now the hard part is knowing what the 'proper' value for the coil is. Some coils will have this listed as a spec (try googling). For others, you might assume the correct coil current is between 3 and 5 Amps, though there's often little evidence to choose a figure like this. If you don't know the 'proper value' then you can use trial and error to determine it (next).
There is a good article on dwell settings, with 'scope shots, here: www.tonyfoale.com/Articles/Ignition/Ignition.htm.

3. **Trial and Error on the Vehicle**: To set the dwell, you want to set it as low as you can without misfiring at idle and low engine speeds. Generally this is between 2.0 and 4.0 milliseconds. If you start at 3.0 and have no misfires, try reducing it a bit (0.1 milliseconds) at a time until you get misfires, then raise it 0.2 milliseconds. If you get misfires at 3.0, raise it a bit at a time until the misfires are gone (then add 0.2 milliseconds). If this seems familiar, it is very similar to the process of setting the PWM% for low impedance injectors.
    - Do not use misfiring and high engine speeds and loads to set the dwell - misfires under those conditions will likely be caused by other factors, and will lead to you setting very long and damaging dwell periods. Recall that the dwell is shortened with higher rpm in any cases, so increasing the maximum dwell setting doesn't help at higher rpms. For example, with a 3.5 millisecond dwell time, and a 2.0 maximum spark duration, the rpm at which the dwell begins to be reduced is just 2727 rpm on a V8.
    - The sometimes suggested method of setting the dwell by increasing it until the VB921 gets hot, then backing it off a bit is NOT RIGHT. This only ensures that the VB921 is not current limiting. However the coil current may be much higher than design rating for some coils (which might be in the typical range of 4 to 6 Amps), and cause the coil to burn up much sooner than expected.
    - Do not increase the dwell to create a 'hotter' spark. Increasing the dwell above the 'saturation point' does not make a hotter spark, it only makes a hotter coil. For example, for a 2.5 millisecond saturation time, the current is increasing to it's maximum for 2.5 milliseconds. If this is 6 Amps, then the average current during this time is about 3 Amps. During this time, much of the energy being supplied is building the magnetic field. However, after 2.5 milliseconds (once saturation of the magnetic field is achieved), the current is a constant 6 Amps, all of which is heating the coil, so it presents a much larger load on the coil. So leaving the dwell too long is a problem. As well, the VB921 goes into current limiting mode if the current gets high (6.5 to 7.5 Amps). If the current is kept below the current limit, the VB921 should not heat up much at all. However, if run with a significant amount of current limiting, it heats up pretty fast (which can ruin it). That's why you should lower it as much as possible without misfires under normal operating conditions. If you need a hotter spark, you need another coil.

4. **Other sources of information**: Ask someone on the forums, look in a factory service manual, or Google to see what might be reasonable dwell values for your coil. For example, the GM 7-pin HEI large cap coil should be 3.5 milliseconds, the GM 8-pin HEI external coil should be 2.5 milliseconds.

Note that you can not set dwell times with EDIS, the EDIS module does all the dwell control internally.

- **Acceleration Compensation (ms)** is the amount of time added to the duration when the accel enrichment is activated.
- **Battery Duration Compensation (ms & V)** (*bat_dur_comp*) is the time added to the dwell to compensate for low battery voltage. You can select both the trigger bin levels (V) as well as the duration added to the dwell (ms).

In general, use the default options and values unless you have a reason to do otherwise. You can calculate the appropriate times using the formula above and substituting different values for **E**.

MAP Sensor

If you have a non-standard MAP sensor (i.e., not the default MPX4250AP), then you need to do a calibration under 'Tools/Sensor Calibration'. You need to set both the MAP and barometer sensor to match the sensor(s) you are using. Press F1 while in the sensor calibration dialog to see values for some common sensors. Do NOT change the barometric correction, just the MAP and barometer sensor. You may need to edit the MegaTune ini files to show the full range of your sensor.

Last Updated: 05/07/2008 21:52:59

MegaSquirt® is an experimental device intended for educational purposes.
MegaSquirt® controllers are not for sale or use on pollution controlled vehicles. Check the laws that apply in your locality to determine if using a MegaSquirt® controller is legal for your application.