PJSC マニュアル

VER1.00 RELEASE　JUN. 2018
VER1.01 RELEASE　JUL. 2019
VER1.02 RELEASE　SEP. 2020

目次

第 1 章 導入

1.1 はじめに

PJSC（Pump Jet Solenoid Controller）は仙人氏が考案し、野郎ワークスが改良実用化した『ポンプジェット』のコントローラーです。

ハードウェア、ソフトウェアは Arduino を使用したオープンソース ECU プロジェクトで開発が進められている Speeduino（https://speeduino.com/wiki/index.php/Speeduino）をベースにしており、ポンプジェットを制御する為の機能を追加して点火制御の機能と汎用出力の一部を省いた Speeduino のサブセット版となります。

最大 4 チャンネルのインジェクタードライバーはインジェクター制御とポンプジェットソレノイドの制御を切り替えて使用する事が可能で、燃調制御に特化した DIY ECU です。

仕様

コントローラー	Meduino Mega2560 R3（Arduino Mega2560 R3 互換ボード）
電源電圧	9-16V
入力	ピックアップ信号 x 2 TPS（Throttle Position Sensor） MAP（Manifold Air Pressure） BALO（Balometric） CLT（Coolant Temperature） IAT（Intake Air Temperature） O2 センサー 排気バルブポジションセンサー
出力	インジェクター／ポンプジェット出力（最大 5A） x 4 汎用出力（最大 1A） x 2 以下の機能から選択 ➢ 燃料ポンプ制御 ➢ ISCV（Idle Speed Control Valve）2 線式 PWM ➢ ブーストコントロール ➢ VVT コントロール ➢ 冷却ファンコントロール ➢ タコメーター信号

燃調制御方式	α－N
	スピードデンシティ
燃調マップ分解能	16 x 16 / 12 x 12
通信	RS-232C シリアルバス　x 1
	USB クライアント　x 1
オプション	Bluetooth シリアル変換モジュール（PC との通信を Bluetooth に変換）

PJSC プロジェクトはオープンソースプロジェクトですが、Speeduino の非公式ユーザープロジェクトという位置付けです。PJSC に関する質問、リクエスト等はプロジェクトオーナーである MAHARU へ問い合わせて下さい。

Speeduino プロジェクトの公認ではありませんので、PJSC に関する質問、リクエストを Speeduino フォーラムには投稿しないよう御注意願います。

1.1.2　ライセンス

PJSC の原型となった Speeduino は GNU GPL に準拠したオープンソースプロジェクトであり、PJSC も GNU GPL ライセンスを継承します。PJSC のソースコード、ハードウェア設計ドキュメント（回路図、基板設計図、パーツリスト）は再配布、改変を伴う再利用を自由に行う事が出来、個人使用、商用利用問わず用途を制限しません。PJSC のソースコード、HW 設計図を再利用して開発したプログラム、ハードウェアもまた GNU GPL ライセンスを継承し、再配布に当たっては GNU GPL ドキュメントを併せて配布しなければなりません。

GNU GPL の詳細については、別途ライセンスドキュメントを参照して下さい。

1.1.3　マニュアル

本マニュアルはセンサー結線等のハードウェア導入と、チューニングソフト『Tuner Studio』を使用したチューニングを行う為の基本的な操作方法について説明しています。PJSC を車両へ設置するに当り、まず始めにこのマニュアルを熟読する事をお薦めします。

エンジンの構造、理論については PJSC を導入する為に必要な最低限の情報しか掲載しておらず、エンジンマネージメントに関する詳細について本マニュアルでは触れていません。エンジンマネージメントシステムの設定は正しく行われないと、エンジンへ致命的なダメージを及ぼす危険があります。PJSC 導入に当たってはエンジンマネージメントを熟知したチューナーが実施する事をお薦めします。

本マニュアルは、PJSC の元となった Speeduino の wiki（https://speeduino.com/wiki/index.php）に記載されている情報に基づいて作成されています。しかし Speeduino プロジェクトは現在も開発中であり、他のオープンソースプロジェクト同様、仕様が頻繁にアップデートされます。その為、本マニュアル内の情報は最新版ファームウェア仕様と異なる場合があります。

最新版の情報が必要な場合は、上記 URL を参照して下さい。

1.1.4　導入

PJSC を導入するには、最初に PJSC ボードに接続する Arduino へ専用のファームウェアを書き込み、チューニングソフトウェア『Tuner Studio』に接続する必要があります。くれぐれもこれを行う前に車両へ接続しないで下さい。

Arduino は USB ケーブルで PC と接続するだけでファームウェアを書き込む事が出来、他に追加のハードウェアは必要ありません。

以降の項ではファームウェアのコンパイルと Arduino への書き込みの方法、Tuner Studio の新規プロジェクト作成方法について説明しています。

1.2　ファームウェアコンパイルと Arduino への書き込み

1.2.1　必要な物

- Windows または Mac OS か Linux がインストールされた PC
- Arduino IDE バージョン 1.6.7 以降
- 最新版の PJSC ファームウェア（以下を参照）
- Tuner Studio MS Lite / Ultimate

1.2.2　ファームウェアのダウンロード

PJSC ファームウェアは OSDN を通じてインターネット上に公開しています。OSDN の下記 URL からリリースパッケージ一式をダウンロードして下さい。

- PJSC Personal Forge：https://osdn.net/users/maharu/pf/PJSC/wiki/FrontPage
- ダウンロード：https://ja.osdn.net/users/maharu/pf/PJSC/files/

補足：

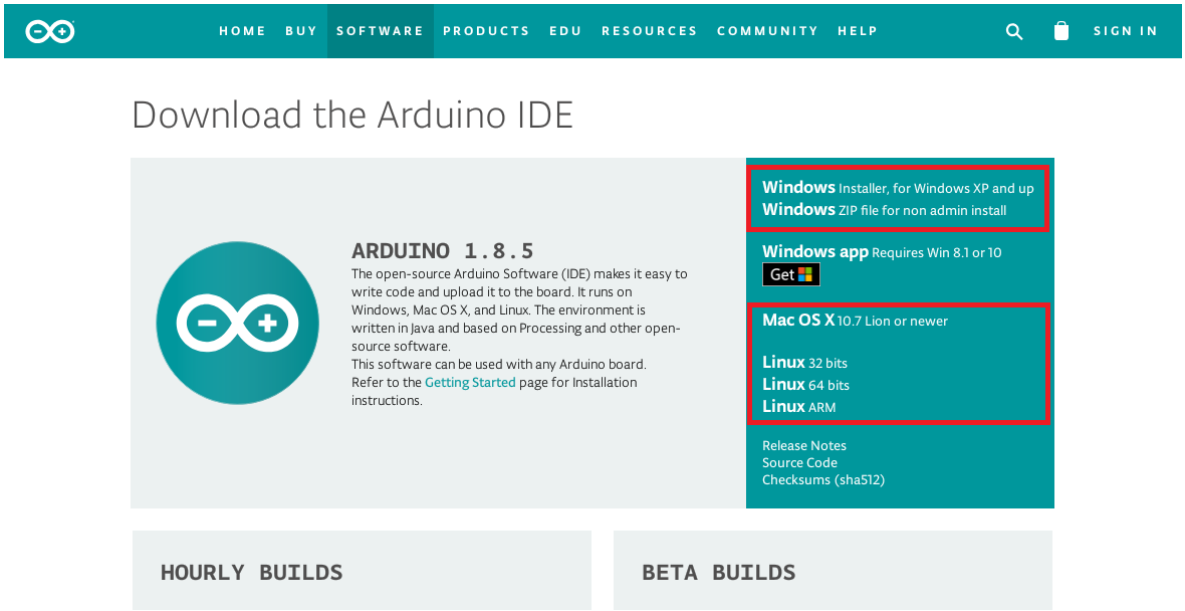
- ファームウェアを更新したら、必ず一緒にリリースされた PJSC.ini ファイルを TunerStudio のプロジェクトに読み込ませて下さい。
- Tuner Studio はバージョン 3.0.2 以降が必要です。
- Arduino IDE バージョン 1.6.7 以降を Windows XP で使用する場合、既知の問題が生じます。回避策については下記 URL を参照して下さい。

  http://speeduino.com/forum/viewtopic.php?f=13&t=555&p=7664&hilit=xp#p7665

1.2.3 ファームウェアコンパイル

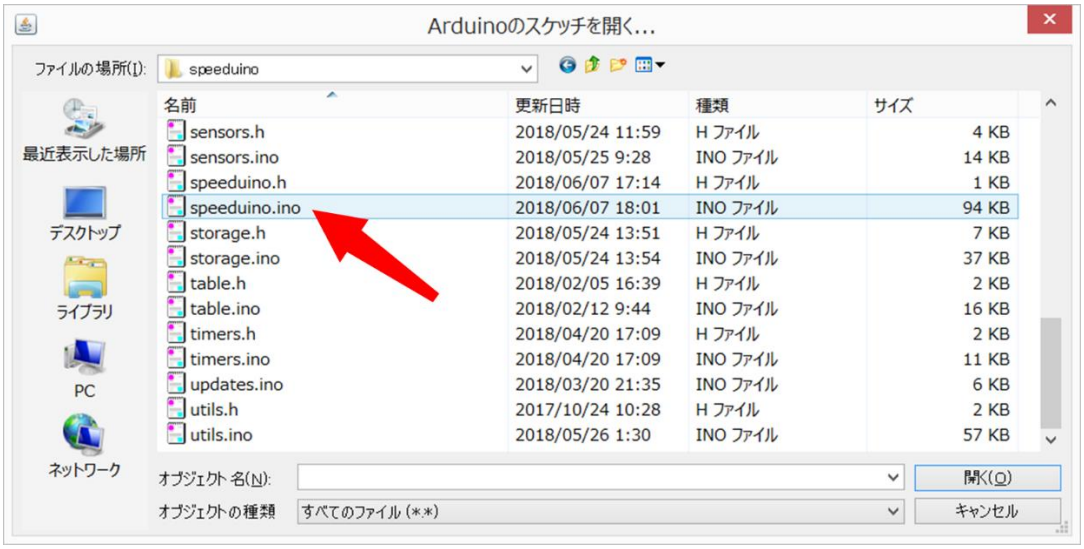
ファームウェアをコンパイルするには、Arduino IDE をインストールした PC が必要です。コンパイルを行う前に、Arduino LLC の下記 URL から Arduino IDE をダウンロードして下さい。

- Arduino Software：https://www.arduino.cc/en/main/software



コンパイルに使用する PC にインストールされている OS に適合するパッケージ、インストーラをダウンロードし、インストールを実施して下さい。

1. Arduino IDE を起動し、メインメニューからファイル＞開くをクリックします。ダウンロードしたファームウェアの”Speeduino”ディレクトリ下にある”Speeduino.ino”を開きます。



2. ボードタイプを選択して下さい。メインメニューのツール＞ボードで”Arduino/Genuino Mega or Mega 2560”を選択します。
3. プロセッサタイプを選択して下さい。メインメニューのツール＞プロセッサで”ATmega2560 (Mega 2560)”を選択します。
4. メインメニューのスケッチ＞検証・コンパイルをクリックすると、コンパイルを開始します。

<< オプション >>

以下はコンパイラーを最適化する為のオプションです。Arduino IDE はデフォルトだとコンパイルオプションとして”-Os”を選択します。これはバイナリファイルサイズを小さくする事を優先するコンパイルオプションです。コンパイルオプションとして”-O3”を選択すると、バイナリのサイズを約 40%大きくする代わりに実行速度を約 20%向上させる事が可能です。コンパイルオプションを変更するには、platform.txt ファイルを編集する必要があります。

1. Arduino IDE が起動している場合は、Arduino IDE を終了します。
2. 以下の場所にある”platform.txt”をテキストエディタで開きます。
   - Windows：c:¥Program Files¥Arduino¥hardware¥arduino¥avr
   - Mac：/Applications/Arduino/Contents/Resourses/Java/hardware/arduino/avr/
3. 開いたファイル内で下記エントリーを”Os”から”O3”に書き換えます。
   - compiler.c.flags
   - complier.c.elf.
4. 編集したファイルを保存し、Arduino IDE を再起動します。

1.2.4　ファームウェア書き込み

前項でコンパイルしたファームウェアを、Arduino へ書き込みます。

1. Arduino Mega と PC の USB ポートを USB ケーブルで接続します。
2. Windows PC では初めて Arduino と接続した際、Arduino のシリアル通信デバイスのドライバーをインストールする必要があります。Arduino オフィシャルボードと多くの互換ボードでは、シリアル通信用 IC として ATMega16U2 または ATMega8U2 を使用しています。この場合、対応ドライバーが自動的にインストールされます。しかし一部の互換ボードでは CH340G が使用されており、この場合は WCH サイトよりドライバーをダウンロードする必要があります。
3. メインメニューのツール＞シリアルポートから Arduino が接続されているシリアルポートを選択します。適切なシリアルポートが選択されている場合、ツール＞ボード情報を取得からボード情報を見る事が出来ます。ボード情報が得られない場合は、異なるシリアルポートが選択されているので、違うポートを選択し直して下さい。
4. メインメニューからスケッチ＞マイコンボードに書き込む、をクリックするとファームウェアが Arduino に書き込まれます。

注）初回のファームウェアの書き込みが正常に完了しても、そのまま PJSC を車両に接続しないで下さい。Arduino は工場出荷状態では EEPROM の内容が全て”FFh”となっています。この状態で PJSC を車両に接続すると、車両の電装系を破損する恐れがあります。車両に接続する前に、必ずファームウェアバージョンに対応したベースチューニングファイルをロードして下さい。ベースチューニングファイルのロードについては、『1.6 章　プロジェクト作成』を参照して下さい。

1.2.5　ファームウェアの確認

ファームウェアの書き込みが完了したら、確認の為に TunerStudio との接続を行います。

Tuner Studio を使った方法以外に、Arduino IDE のシリアルモニタ機能を使ってファームウェアの確認を行う事も出来ます。この機能はメインメニューのツール＞シリアルモニタから起動出来ます。

シリアルモニターウィンドウのコンソールで、大文字の "S"（引用符なし）を入力し Enter を押します。Arduino はインストールされているファームウェアがビルドされた年月を返します。（例：Speeduino 2017.03）

注）ボーレートが 115200 に設定されていることを確認してください

またコンソールで"?"を入力すると Usage が表示されます。

===コマンドヘルプ===

すべてのコマンドは 1 文字で表され、続いてスペースを入れずにパラメーターを入力します。 一部のパラメータはバイナリであり、通常はプロンプトを介して入力する事は出来ません。

構文：<command> + <parameter1> + <parameter2> + <parameterN>

===コマンド一覧===

A - 81 バイトの currentStatus の値をバイナリで表示します。

B - 現在のマップと configPage の値を EEPROM に書き込む

C - COM ポートをテストします。 ECU が接続されているシリアルポートを確認するために TunerStudio が使用します。バイナリ値が返されます。

L - マップページ（テーブル）または configPage 値を表示します。 ページ内容を変更するには P コマンドを使用して下さい。

N - 新しい行を出力します。

P - 現在のページを設定します。構文：P + <ページ番号>

R - A コマンドと同じ

S - 署名番号を表示する

Q - S コマンドと同じ

V - map または configPage の値をバイナリで表示する

W - map または configPage に 1 バイトを設定します。バイナリパラメータが必要です。構文：W + <offset> + <newbyte>

t - 校正値を設定します。バイナリパラメータが必要です。テーブルインデックスは 0、1、または 2 です。

構文：t + <tble_idx> + <newValue1> + <newValue2> + <newValueN>

Z - 較正値を表示する

T - バイナリで 256tooth 分のログエントリを表示します。

r - 256tooth 分のログエントリを表示します。

? - このヘルプページを表示します。

Tuner Studio を起動して接続を試みる事で、新しい PJSC ファームウェアのテストが出来ます。 詳細は「Tuner Studio 接続」の項を参照して下さい。

1.2.6　トラブルシューティング

対応していない Arduino ボードを接続した場合

ファームウェアをコンパイルした時に以下の様なエラーメッセージが出る場合、ボードの選択が不適切です。ツール＞ボードからボードタイプ”Arduino/Genuino Mega or Mega 2560”を選択し直して下さい。

scheduler.ino:317:7: error: 'OCR4A' was not declared in this scope

scheduler.ino:323:8: error: 'TIMSK5' was not declared in this scope

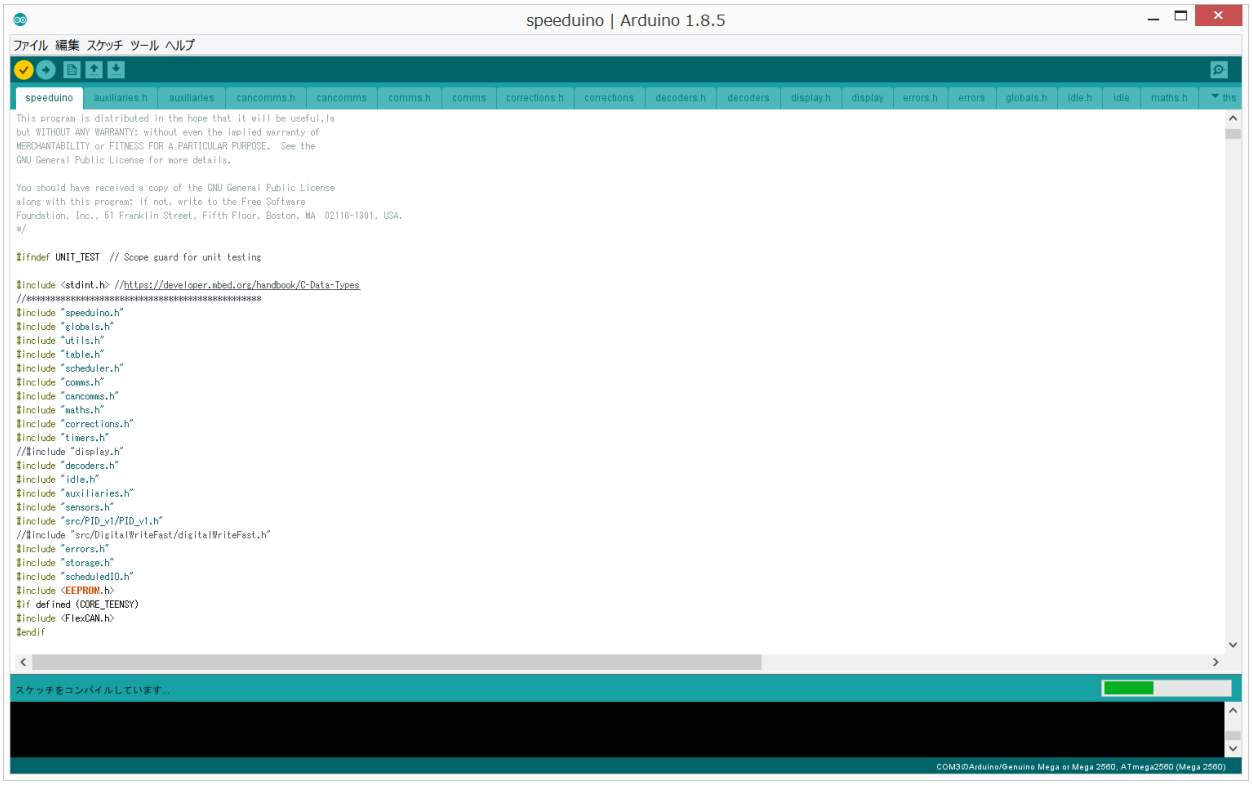
scheduler.ino:323:25: error: 'OCIE4A' was not declared in this scope

Speeduino プロジェクトが開かれていない場合

Arduino IDE で Speeduino.ino しか開かれておらずプロジェクトとして開かれていない場合、以下の様なエラーメッセージが表示されます。

speeduino.ino:27:21:fatal error: glovals.h: No such file or directory

この場合、全てのソースコードファイルを同じディレクトリにコピーし、ファイル＞開くから speeduino.ino を開き直して下さい。プロジェクトが正しく開かれた場合、全てのソースコードファイルのタブが画面のトップに表示されます。



1.3 Tuner Studio との接続

Tuner Studio は PJSC の推奨チューニングソフトウェアです。Tuner Studio は Windows、Mac、Linux 上で動き、PJSC の設定とログを取得する機能を持っています。

ファームウェアのコンパイルと Arduino への書き込みが完了していれば、TunerStudio への接続の準備が出来た事になります。ファームウェのコンパイルと Arduino への書き込みが完了していない場合は、1.2 項を読んで書き込みを完了させて下さい。

1.3.1 Tuner Studio のダウンロード

Tuner Studio を使用するには、下記 EFI Analytics の HP より使用している OS に対応した Tuner Studio インストールファイルをダウンロードして下さい。

- EFI analytics: http://www.tunerstudio.com/index.php/downloads

Tuner Studio には無料版の Tuner Studio Lite と、有料版の Tuner Studio Ultimate があります。
TunerStudio Lite は一部の機能が使えませんが、PJSC で基本的なセッティングを行うのに必要な機能は揃っています。

Tuner Studio Ultimate では Tuner Studio の全ての機能が利用出来ます。その一部は、AF センサーのフィードバックを用いたオートチューニングや、自動で燃調初期マップを作成するマップジェネレーター等です。
Tuner Studio Lite をインストールしているなら、EFI Analytics の HP からライセンスを購入する事で Ultimate へアップデートする事が出来ます。

1.4 Tuner Studio インターフェイス

1.4.1　スタートメニュー

Tuner Studio を起動すると、以下に示すようなスタートメニュー画面が表示されます。

①ファイルメニュー

[ファイル]メニューからプロジェクトを開いたり、新規プロジェクトを作成することが出来ます。

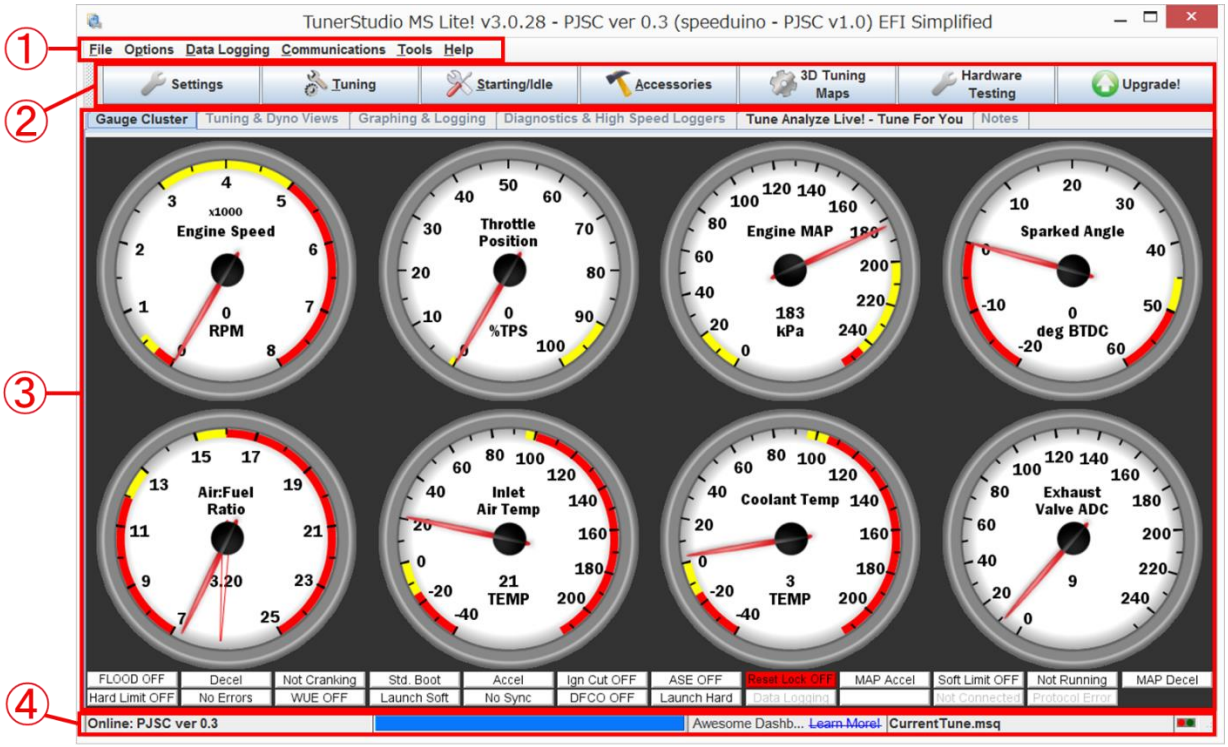
この画面の[ファイル]メニューからチューニングファイル（拡張子.msq）を開く事が出来ます。しかしここではチューニングファイルの中身を参照或いは解析が出来るだけで、PJSC へ接続したり、この画面から開いたチューニングファイルを PJSC に転送する事は出来ません。PJSC への接続、転送を行うにはプロジェクトを作成または開く必要があります。

②Open Last Project

Open Last Project をクリックすると、最後に開いていたプロジェクトを開く事が出来ます。プロジェクトのオープンまたは作成の詳細については、このガイドの 1.1 項を参照してください。

1.4.2　メインスクリーン

プロジェクトを開いたり作成したりすると、以下のような TunerStudio のメイン画面が表示されます。



①トップメニュー

トップメニューについては、このガイドのセクション 4.7 に詳細が記載されています。このメニューは、主にアプリケーションに関連した項目があります。具体的には、次のような項目です。

- プロジェクトの作成、オープン、バックアップ、およびチューニングファイルの作成。
- Tuner Studio の動作設定を変更
- 解析のために PJSC からデータを採り込む
- PJSC と TunerStudio 間の通信設定
- PJSC で使用するセンサーのキャリブレーション
- TunerStudio のヘルプファイルにアクセスして、TunerStudio のバージョンに関する情報を取得

②ツールバーメニュー

ツールバーメニューについては、このガイドのセクション 4.16 に詳細が記載されています。このメニューには、PJSC の設定とチューニングの機能が含まれています。

③メイン画面のタブ

メインスクリーンのタブについて、このガイドの第 17 章から第 20 章に詳細が記載されています。メイン画面のこの領域には、選択したタブに従って 2 組のビジュアルデータが表示されます。

- ゲージクラスタと関連する標識ラベル。
- 診断およびデータロガー。

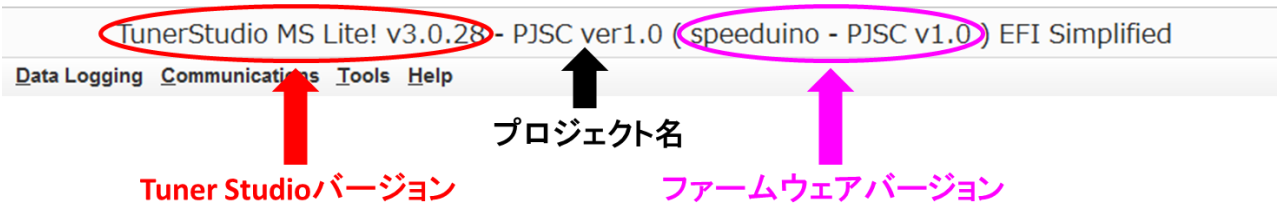
④ステータスバー

ステータスバーには以下の情報が表示されます。

- 現在のプロジェクトの名前。
- ソフトウェアがタスクを実行している間に時々使用されるプログレスバー。
- 製造元のインターネットサイトへのリンク。
- オプションで、現在のチューンファイル（または CurrentTune.msq）の名前 - セクション 2.3 を参照してください。ファイルのチューニングの詳細については、このガイドの「
- 通信インジケータ - ステータスバーの右側にある赤と緑のボックスは、TunerStudio と MS2 の間で情報がいつ転送されているかを示します。

タイトルバー

タイトルバーには、TunerStudio のバージョン、現在開かれているプロジェクト名、ファームウェアのバージョンが表示されます。

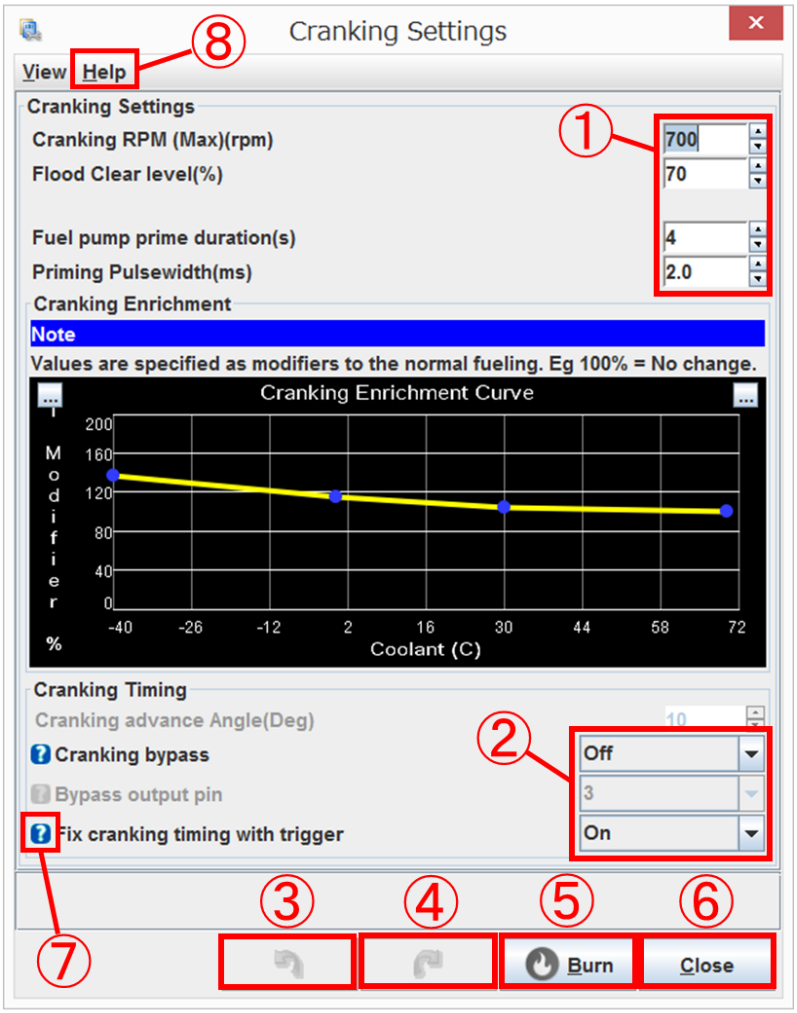


1.5　　アプリケーションインターフェースと使用方法

1.5.1　　データ入力画面

チューニングの為の設定を入力する各種設定ダイアログは、テキストボックス、ドロップダウンボックス、およびボタンで構成されています。

以下の画像はツールバーメニューの Starting/Idle＞ Cranking Settings で表示される始動補正設定ダイアログ（Cranking Settings Dialog）です。これを例に、ダイアログの構成を説明します。



①テキストボックス

テキストボックスはキーボードからの数値を入力を受けつけるように設計されており、数字と小数点のみ入力出来ます。 またテキストボックスの右側には上下（▲▼）ボタンがあり、これにより数値を増減出来ます。

②ドロップダウンボックス

ドロップダウンボックスでは右側のドロップダウンボタン（▼）をクリックすると、その項目で選択可能なオプションが表示されます。表示されたオプションの内の一つを選択してクリックする事で、その項目の設定を決定する事が出来ます。

ドロップダウンボックスで機能を有効にすると、関連する他の項目も有効になって入力可能になるものがあります。その様な項目では機能を無効にすると、関連する項目はグレーアウトされ入力を受け付けなくなります。

例えば上記の始動設定ダイアログでは、Cranking bypass を有効（On）にすると Bypass output pin の項目が入力可能になります。Cranking bypaas を無効（Off）にすると Bypass putput pin がグレーアウトして入力を受け付けなくなります。

③/④アンドゥ（Undo）、リドゥ（Redo）ボタン

アンドゥボタン（③）をクリックすると直近で入力した項目が変更前に戻ります。リドゥボタン（④）をクリックすると、アンドゥで戻した項目をもう一度変更後状態にします。

⑤書き込み（Burn）ボタン

TunerStudio で変更した設定を PJSC に反映させるには、PJSC の EEPROM に書き込む必要があります。Burn ボタンをクリックすると変更した項目の設定が EEPROM に書込まれ、PJSC の電源を切っても次回電源投入時に設定が反映されます。但し燃料テーブル（VE Table）は例外で、テーブルの値を変更すると PJSC のメモリ上のテーブルに変更が即時反映され、リアルタイムでチューニングを行う事が出来ます。しかしメモリ上の VE テーブルは電源を切るとクリアされてしまう為、そのままでは次回の電源投入時に変更は反映されません。変更を永続的に反映させるには、Burn ボタンで EEPROM に書込む必要があります。

⑥クローズ（Close）ボタン

このボタンをクリックすると、開かれているダイアログが閉じられます。当該ダイアログで変更した項目があり EEPROM へ書き込みを行っていない場合、このボタンをクリックしてダイアログを閉じた時に変更内容が自動的に EEPROM に書き込まれます。

⑦TIPS

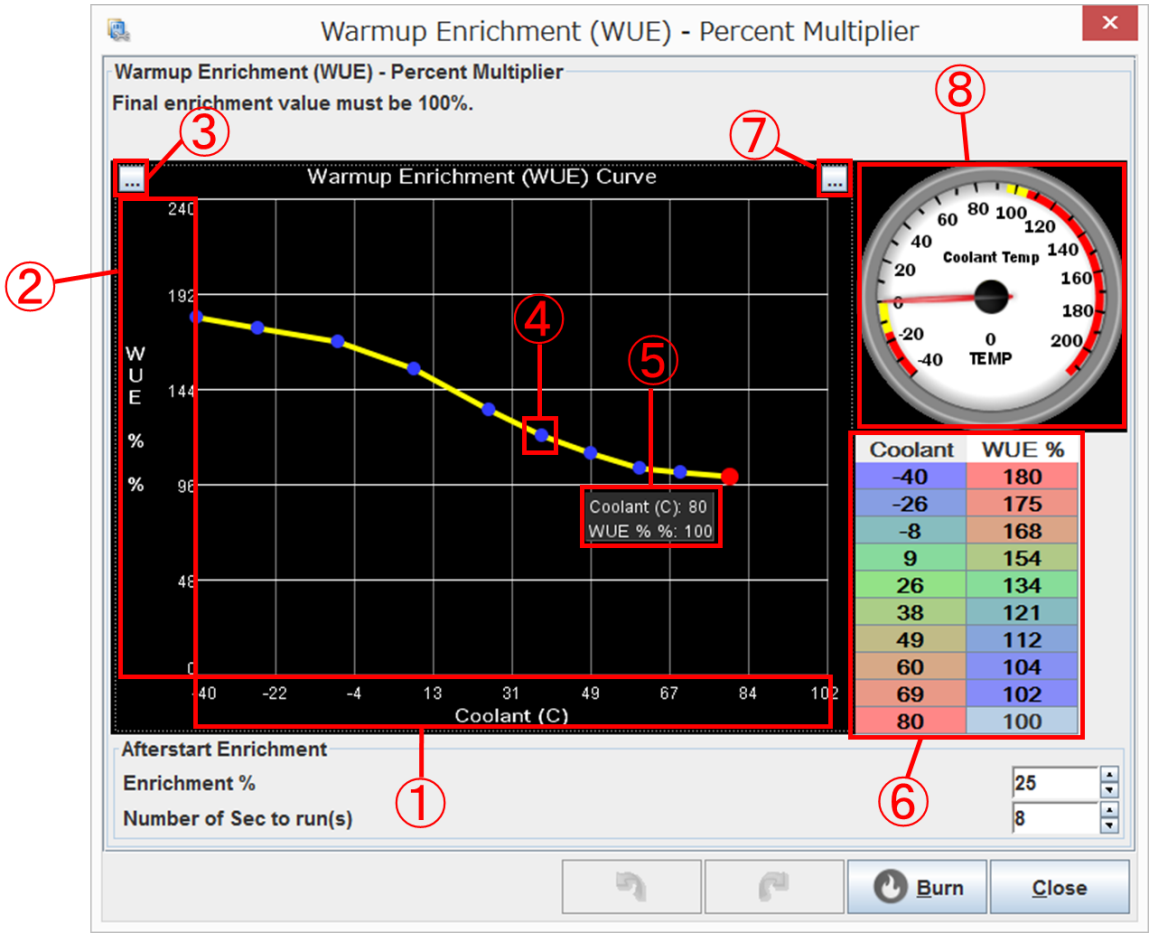
青いクエスチョンマークのボタンをクリックすると、該当項目の簡単な説明がポップアップ表示されます。

⑧ヘルプ

オンラインヘルプメニューです。クリックすると、Speeduino wiki ページの解説ページを開きます。

1.5.2　　設定カーブ

Tuner Studio では、種々の補正値をエンジン回転数に対して任意の非線形な値に設定する事が可能です。以下にはその一例として、暖機補正設定ダイアログ（Warmup Enrichment Dialog）を示します。



①X 軸

X 軸には、設定項目の主変数（この例では水温）とその単位（この例では℃）が表示されます。

②Y 軸

Y 軸には、設定項目の従変数（この例では燃料補正係数）が表示されます。

③オプションメニューボタン

左上のオプションボタンをクリックすると、各軸のスケール設定を変更出来るポップアップメニューが表示されます。

④ポイント

各カーブには固定数のポイントがあり（この例では9つあります）、カーブを補正する為に使用出来ます。任意のポイントをクリックして移動する事で、カーブを任意の形に成形する事が出来ます。

⑤選択ポイント情報

任意のポイントが選択されると（この例では一番右端のポイント）、ポイントの色が変わり座標（この例では、水温 80℃、暖機補正係数 100%）を示すポップアップボックスが表示されます。

⑥データテーブル

カーブ上のポイントをクリックしてドラッグ以外に、データテーブルに値を入力する事でカーブを変更する事が出来ます。カーブ上のポイントをクリックしてドラッグすると、テーブルのエントリのバックグラウンドカラーが変わります（この例では、9 番目のボックスのバックグラウンドが灰色になっています）。

⑦データテーブルの表示/非表示

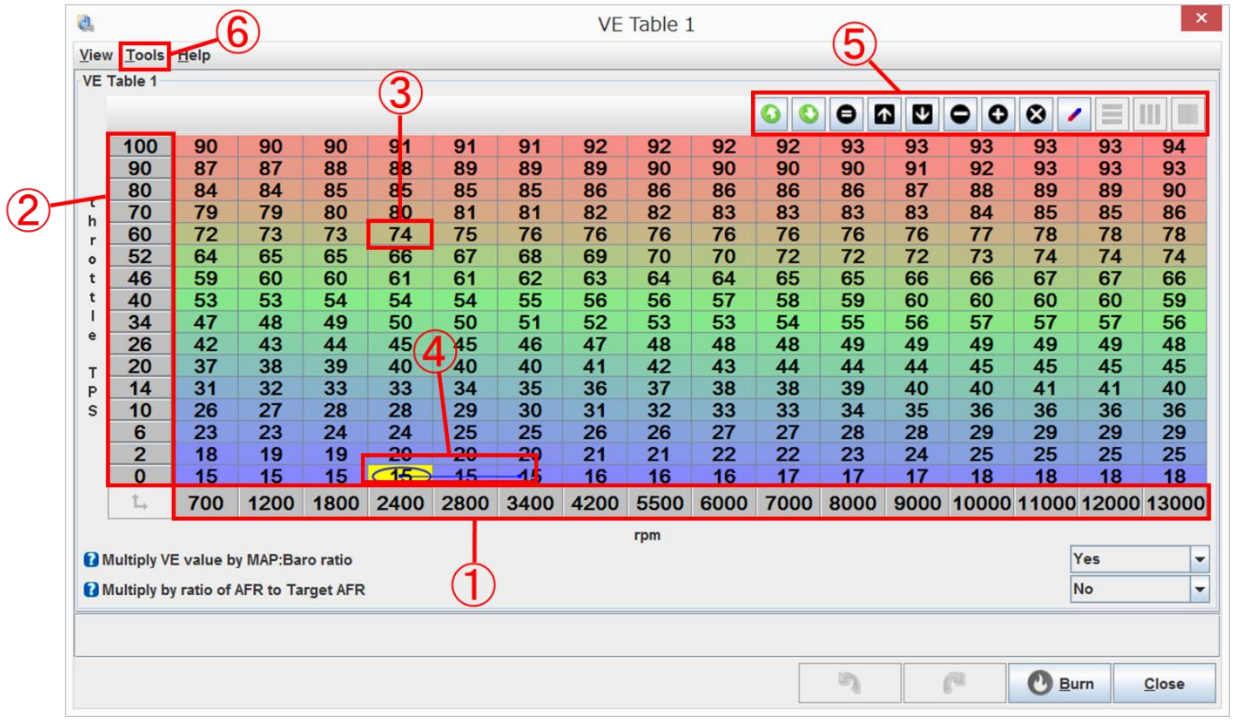
右上のボタンをクリックすると、⑥データテーブルを表示にするか非表示にするかを切り替えられます。一部のカーブはポイント数が少なく、デフォルトではデータテーブルが非表示になっています。ポイント数が少ない小さなカーブは、データテーブルを同時に表示するとカーブを表示するエリアが狭くなり、カーブが見難くなってしまいます。その様な時に、このボタンをクリックしてデータテーブルを非表示にするとカーブを識別し易くなります。

⑧ゲージ

エンジン始動時はこのゲージで、数値の変化をリアルタイムにモニターする事が出来ます。

1.5.3　3D テーブル

3D テーブルは、チューニングパラメーターの変化をイメージし易い様に視覚化したもので、PJSC では燃料 VE テーブルに用いられます。 以下の画像は、その一例です。



①/② X 軸/Y 軸の値

これらの値は、テーブル内で値を任意に設定出来るポイントの座標を表します。 この例では、エンジン回転数（RPM）と燃料負荷－スロットル開度（Throttle position sensor, TPS）に設定可能な 16 個の座標値があります。 軸上の何れかの値をクリックすると、任意の値を入力する事が出来ます。

③セル

VE テーブルには 16 x 16 の合計 256 のセルがあり、それぞれ任意の値を設定出来ます（この例では、設定された値が容積効率%として反映されます）。任意のセルをクリックするとキーボードから値を入力出来るようになります。

キーボードで入力する以外に、以下の方法があります。

ホットキー

単一のセルをクリック（またはドラッグしてセルの範囲を選択）し、右クリックするとポップアップメニューが表示されます。 このメニューで選択出来る機能はボタン⑤でも実行出来ます。また各メニューの右側

にアサインされているホットキーが表示されており、このホットキーを使用する事でテーブルへの値の入力を簡便化する事が可能です。

④現在参照セルと軌跡

テーブル上の現在参照されているセルを示します。 また直近で参照されたセルとその軌跡を表示し、エンジンの運転状態の変化に伴う参照セルの移動を示します。

軌跡は直近数回分の参照セル履歴をから作成されますが、この履歴の数を変更する事も可能です。テーブルを右クリックし、ポップアップメニューの下部にある“History Trail Length”（参照軌跡履歴長）をクリックします。開かれたポップアップテキストボックスに表示したい履歴数を入力する事が出来ます。

⑤ボタン

テーブル内の値を、これらのボタンを使用し変更する事も可能です。変更したいセル（単一セルでも、複数セルでも可）を選択し、ボタンをクリックすると選択したセルに対して変更操作が実行されます。

 テーブルをファイルにエクスポートします。

 ファイルからテーブルをインポートすることができます。

 ポップアップテキストボックスを開き、選択したセルの値を入力した値に設定します。

 ボタンをクリックするたびに、選択したセルの値が 1 ずつ増加します。

 ボタンをクリックするたびに、選択したセルの値が 1 ずつ減少します。

 ポップアップテキストボックスを開き、選択したセルの値を入力した値だけ減らします。

 ポップアップテキストボックスを開き、選択したセルの値を入力した値だけ増加させます。

 ポップアップテキストボックスを開き、元の値に入力した値を掛け、選択したセルの値を増加させます（たとえば、1.25 を入力すると、選択したセルの値が 25%増加します）。これは値を減らすためにも使用できます（たとえば、0.8 を入力すると選択したセルの値が 20%減少します）。

 選択した全てのセルを、選択範囲の四隅のセルによって補間された値に調整します。

テーブル操作には、Windows デフォルトのショートカットキーも使用可能です。コピー（Ctrl＋C）およびペースト（Ctrl＋V）のショートカットキーがサポートされており、選択した値を別のセルやテーブル、または Excel などの表計算ソフトにコピーする事が出来ます。

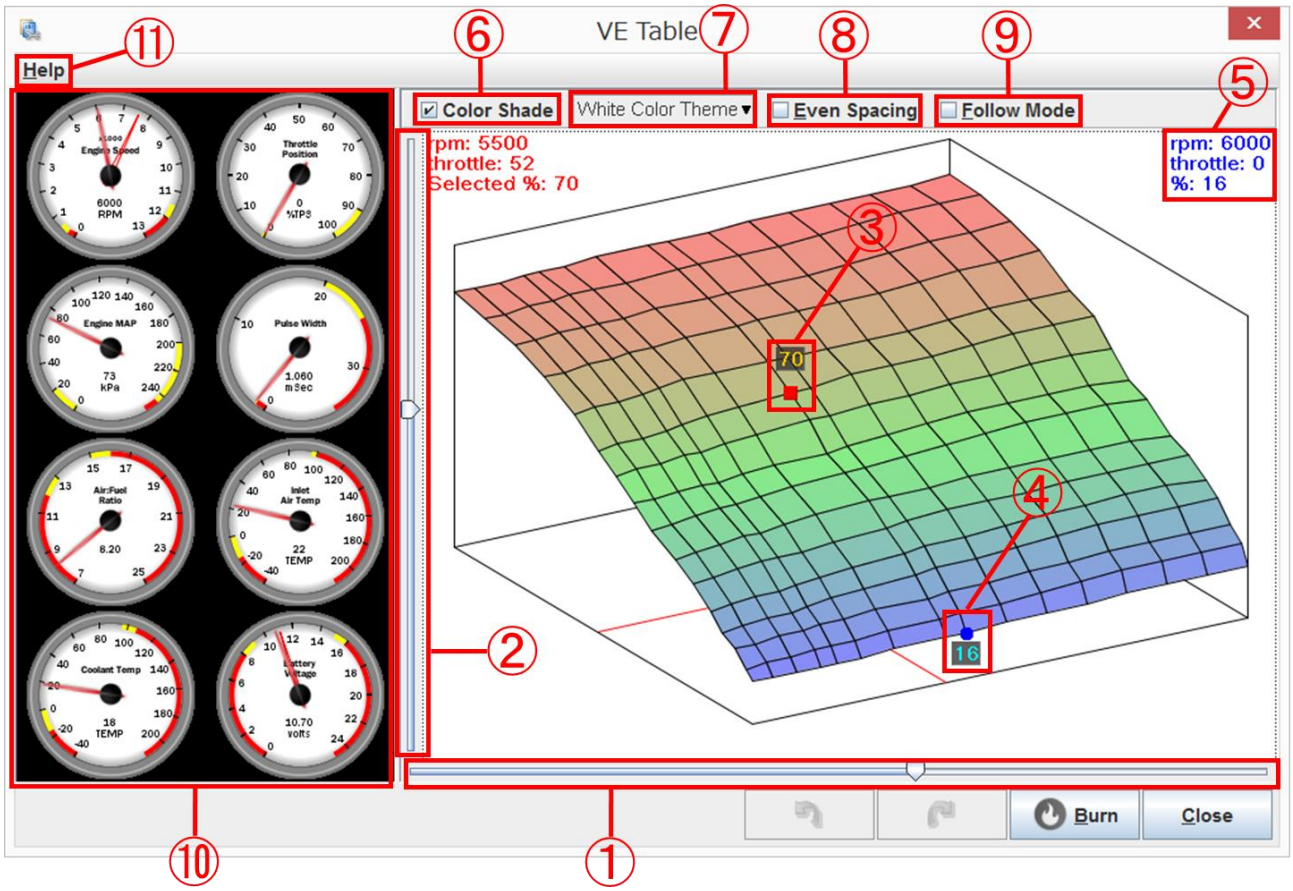
⑥ツール

このメニューは一部のテーブル画面にて、テーブルジェネレータオプションを表示します。これは設定されたエンジンとインジェクターの仕様から、理論上の要求燃料を計算して VE テーブルを自動で作成する機能です。この機能は Tuner Studio の有料版でのみ使用可能です。

1.5.4. 3D チューニングマップ

3D チューニングマップは、VE テーブルを 3 次元表示する事で直感的に捉えられるようにし、燃調セッティングの手助けとなります。VE マップ以外にもテーブルデータの殆どが、3D マップ表示に対応しています。

下図は前項の例で示した燃料 VE テーブルに対応する 3D マップです。



3D マップは 3D テーブルの 256 個のポイントそれぞれの X 座標および Y 座標軸に対して、ポイントに入力された数値を高さ（Z 軸）の座標として図式化した物です。3D マップによって、エンジンの回転数、負荷（TPS または MAP）の変化に対して燃調グラフの傾き（変化率）が直感的に把握出来ます。

3D テーブルと同様、数値が低い箇所は青で、高い箇所は赤で表示され、中間の範囲は緑で表示されます。

また 3D マップはマップをクリックしてドラッグすると、異なる角度から 3D マップを見る事が出来ます。マップの左と下にあるスライダーを使って視点を変更する事も可能です。

①ヨースライダー

ヨースライダを使用すると、マップを水平面に沿って回転させることが出来、前面、背面、側面から見ることが出来ます。

②ロールスライダー

ロールスライダを使用すると、マップを垂直面内で回転させる事が出来ます。

ヨーとロールの組み合わせを使用すると、全ての角度からマップを表示する事が出来ます。

③選択ポイント

全ての線の交点（ポイント）は 3D テーブルのセルに対応します。これらのポイントはクリックする事で選択出来ます。選択されたポイントは色（赤）付きのマーカーで強調表示されます。選択したポイントを上下にドラッグする事で、対応する 3D テーブルのセルの値を増減させる事が出来ます。

④アクティブポイント

PJSC が接続されているエンジンが稼動中の場合、エンジンが動作している位置が青色のマーカーで表示されます。このポイントがアクティブポイントになります。X 軸に回転数（RPM）が設定されている 3D マップでは、スロットルを開閉するとこのマーカーが動くのを見る事が出来ます。

エンジンが停止している状態や、PJSC が Tuner Studio と接続されていないオフラインの状態では、アクティブポイントは表示されません。

⑤選択ポイントとアクティブポイントの情報ボックス

アクティブポイントが表示されると、3D マップ右上のテキストエリアにアクティブポイント情報が表示されます（この例では rpm：6000、throttle：0、VE%：16）。

マップ上のポイントが選択されると、マップ左上のテキストエリアに選択ポイントの情報が表示されます（この例では rpm：5500、throttle：52、VE%：70）。

⑥カラーシェード（Color Shade）

マップにカラーシェードのオン／オフを切り替えられるチェックボックスです。ボックスをクリックしてチェックを入れるとマップのカラーシェーディングがオンになり、チェックを外すとオフになります。

⑦カラーテーマ（Color Theme）

マップの背景色を白、灰、黒から選択出来ます。

⑧等間隔化（Even Spacing）

“Even Spacing”横のチェックボックスをクリックしてチェックを入れると、ポイントとポイントの間隔を軸の値に関係無く等間隔にします。

⑨追従モード（Follow Mode）

“Follow Mode”左側のチェックボックスをクリックしてチェックを入れると、追従モードがオンになります。追従モードがオンに設定されている場合、選択ポイントがアクティブポイントに追従して移動します。追従モードがオフの時は、アクティブポイントに制限される事無く任意のポイントを選択出来ます。

エンジン稼働時に稼働ポイントの設定を変更したい場合、追従モードがオンになっていると設定値の変更が容易になります。

⑩ゲージクラスター

エンジン稼働時にはゲージクラスタでエンジンの状態をリアルタイムにモニターする事が出来ます。

⑪ヘルプ

ヘルプを選択すると、Tuner Studio 内の"3D Table Usage"ヘルプファイルを開く事が出来ます。

3D マップホットキー

3D マップの設定値を変更するために使用できるショートカットキーについて、以下に説明します。

上、下、左、右カーソルキー - 選択ポイントを移動出来ます。

選択ポイントの値の増減：

- **Shift + ↑ or → キー** - 選択ポイントの値を 1 だけ増加させます。
- **Shift + ↓ or ← キー** - 選択ポイントの値を 1 だけ減少させます。
- **> or . or q or + or = キー** - これらのキーは選択ポイントの値を 1 だけ増加させます。
- **< or , or w or - キー** - これらのキーは選択ポイントの値を 1 減少させます。
- **CTRL + ↑ or → or > or . or q + or = キー** - 選択ポイントの値をユーザー定義値で増加させます（デフォルト値は 5）。

- **Ctrl + ↓ or ← or < or , or w or - キー** - 選択ポイントの値をユーザー定義値で減少させます（デフォルト値は 5）。

G - 選択ポイントを現在のアクティブポイントに移動します。

F – 追従モードのオン/オフを切り替えます。

M – 3D マップのヨー角を 10 度増加させます。

K – 3D マップのヨー角を 10 度下げます。

N – 3D マップのロール角を 10 度上げます。

J – 3D マップのロール角を 10 度下げます。

Z – 3D マップを真上から見たトップダウンビューを表示します。

3D マップメニューオプション

3D マップ自体を右クリックすると、カスタマイズ用ポップアップメニューが表示されます。これらについて以下に説明します。

Smart Select Movement – カーソルキーによる選択ポイントの移動方向の定義を変更します。オンの場合（デフォルト設定）、選択ポイントは 3D マップのアングルに関係無く画面に対してカーソルキーの示す方向に移動します。オフの場合、→は X 軸の増加方向へ、左は X 軸の減少方向へ、↑は Y 軸の増加方向へ、↓は Y 軸の減少方向へ選択ポイントを移動します。

Show Active Table Values - オンにすると（デフォルト設定）、選択ポイントとアクティブポイントの値がポイント上に表示されます。

Show Selected X & Y Values - これはデフォルトではオフです。オンにすると選択ポイントの値が X 軸及び Y 軸上に表示されます。

Increment All Active Cells – この機能は追従モードと併用して使用します。追従モードとこの機能をオンにすると、エンジン稼働状態で選択ポイントの値を変更すると、ポイント移動軌跡とその周辺のセルの値にも一定の重み付けがされた変更が適用されます。これにより、最大でアクティブポイントを含む 4 つのセルの値を一度に調整する事が出来ます。オフにすると、選択ポイントのみが調整されます。

Active Weight Threshold - これは Increment All Active Cells がオンに設定されている場合にのみ有効です。アクティブポイントの周辺セルの変更に適用される重みを変更する事ができます。設定可能な値は 0%～100%です。

Select Active Color - アクティブポイントとその情報テキストの色を変更できます。デフォルト設定では、アクティブポイントとテキストは上記の例のように青色になります。

Select Selected Color - 選択ポイントと選択ポイント情報のテキストの色を変更できます。デフォルトでは、上記の例のように赤色になります。

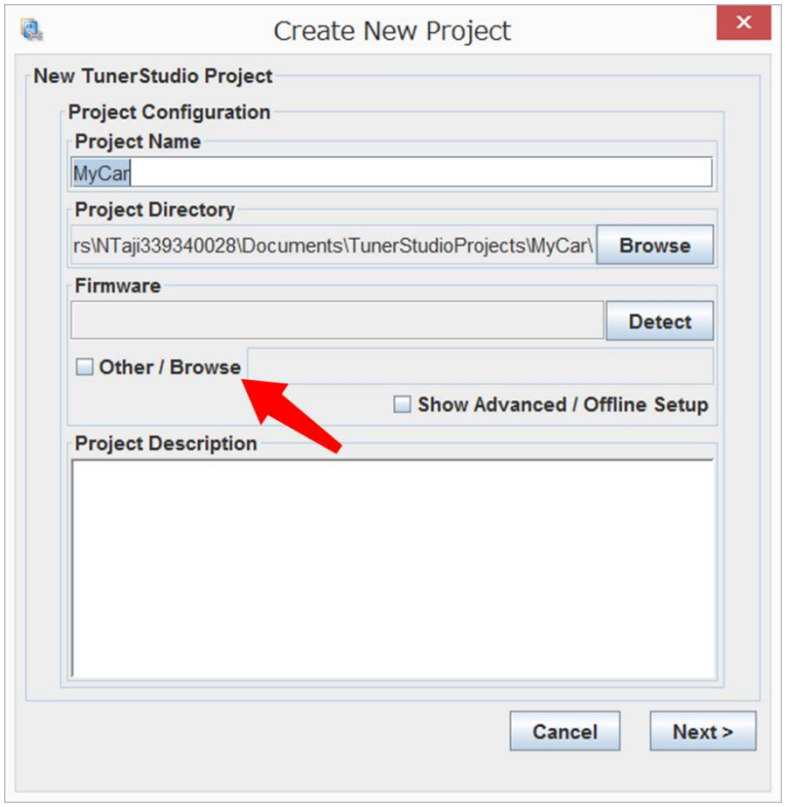
CTRL Increment By - ホットキーセクションで説明したように、CTRL キーを押しながらテーブル値を調整すると値を増減できます。このオプションを使用すると、値を増減するステップ値（デフォルト値は 5）を変更できます。

1.6 プロジェクト作成

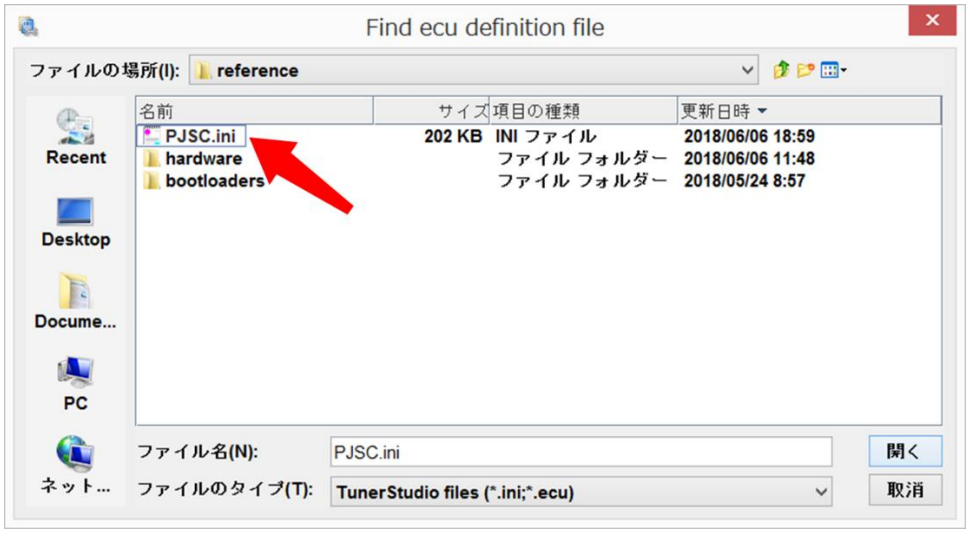
1.6.1 新規プロジェクト作成

Tuner Studio の初回起動時には、セッティングを進める為に新規プロジェクトを作成する必要があります。Tuner Studio スタートアップ画面にて'Create new project'をクリックして下さい。

プロジェクト作成ダイアログにて、任意のプロジェクト名を入力して下さい。プロジェクトは使用する ECU 設定、燃調マップ等のチューニングデータを含むので、プロジェクト名はチューニングを行う車両を識別出来るものにして車両毎にプロジェクトを作成する事を推奨します。



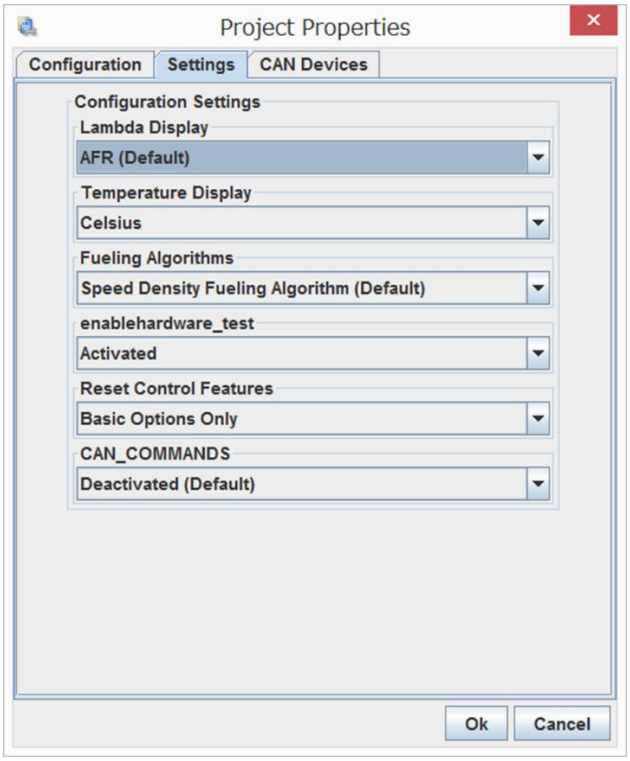
また Tuner Studio が PJSC と通信するには、ファームウェア定義ファイルが必要です。プロジェクト作成ダイアログの'Other / Browse'ボタンをクリックし、Speeduino ソースディレクトリ下の reference サブフォルダから'PJSC.ini'ファイルを選択して下さい。



'PJSC.ini'ファイルを選択したら、プロジェクト作成ダイアログの Next ボタンを押して下さい。

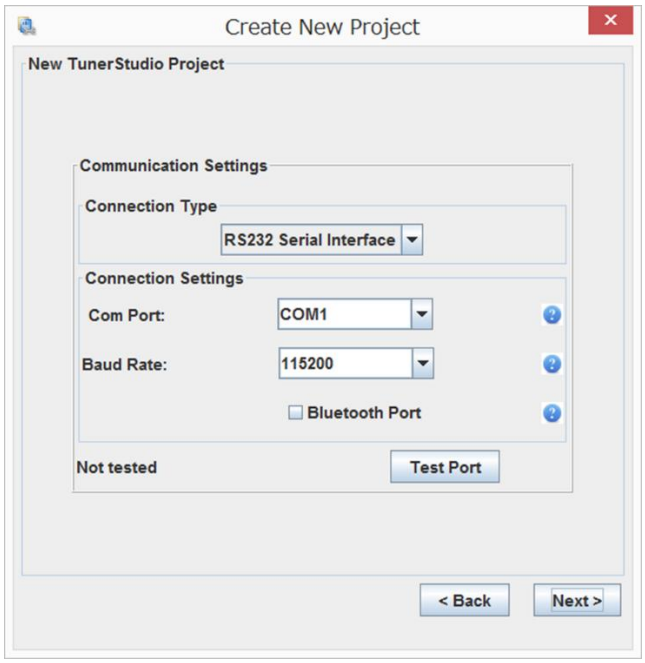
Configuration options

次にコンフィギュレーションセッティングダイアログにて、作成するプロジェクトに合わせたコンフィギュレーションパラメーターを指定して下さい。このパラメーターはプロジェクト作成後にいつでも変更可能ですので、プロジェクト作成時はそのまま OK をクリックして次に進んでも構いません。



Communication settings

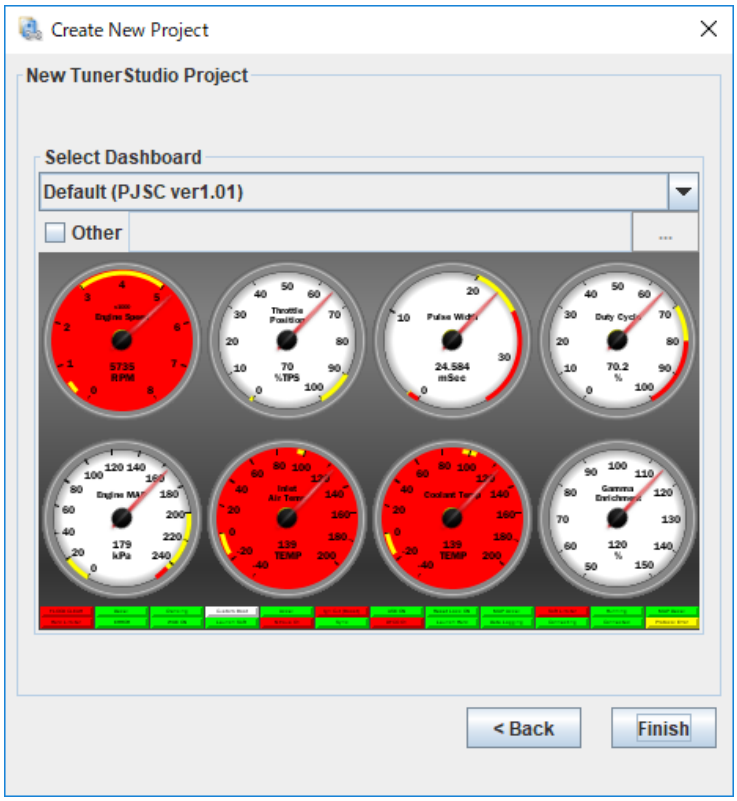
Arduino と PC が通信する為に使用するポート、設定を選択して下さい。これは Arduino IDE で選択したポートと同じです。ボーレートは 115200bps を選択して下さい。



Select Dashboad

ダッシュボード選択ダイアログが表示されます。初回はデフォルトダッシュボードしか選択出来ません。ダッシュボードはプロジェクト作成後に変更可能ですので、そのまま'Finish'ボタンを押して構いません。

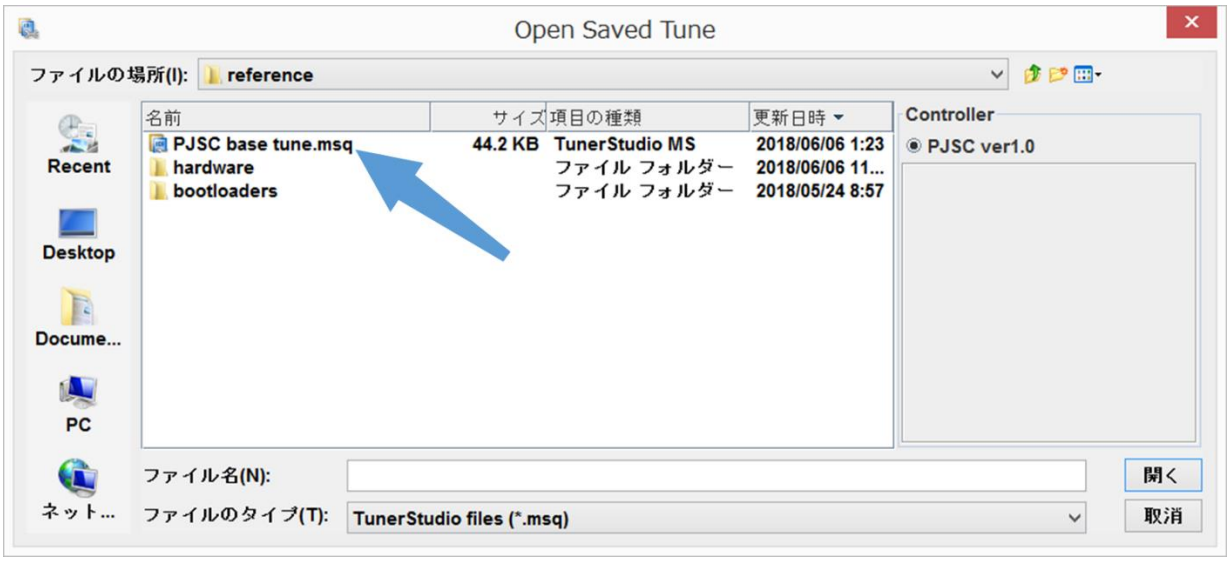
Finish ボタンを押すと新規プロジェクト作成は完了し、ダッシュボード画面が表示されます。



ベースチューニングファイルの選択

新しくプロジェクトを作成した場合、多くのパラメーター値は不定です。これを初期値にする為にベースチューニングファイルを読み込んで下さい。ベースチューニングファイルを読み込まずにチューニングを進めると、一部のパラメーターが不適切な値のままとなり、PJSC が正しく動作しない可能性があります。

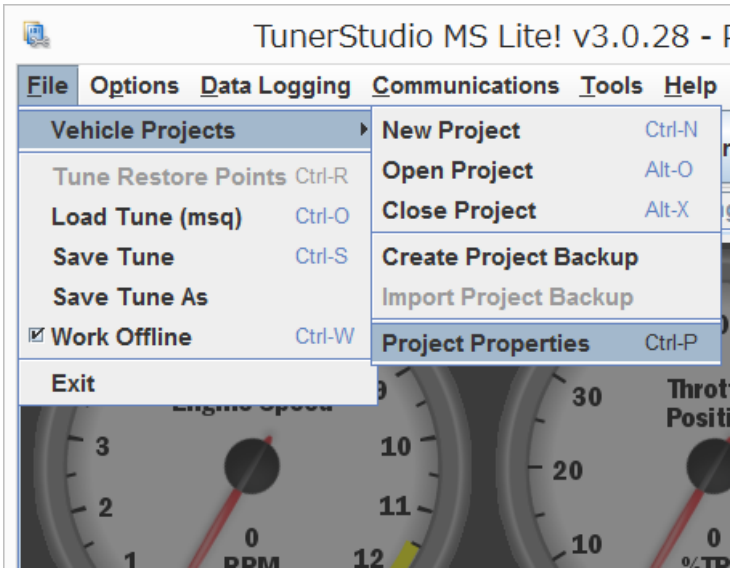
ベースチューニングファイルは'PJSC base tune.msq'という名称で、PJSC ファームウェアディレクトリの reference サブフォルダ内にあります。TunerStudio メインメニューの File＞Open Tune(msq)からベースチューニングファイルを選択して、PJSC へ書き込んで下さい。



1.6.2　プロジェクトプロパティ設定

TunerStudio メインメニューの File＞Vehicle Projects＞Project Properties をクリックするとプロジェクトプロパティダイアログが表示され、プロジェクトの設定を変更出来ます。

以下は各設定項目の説明になります。

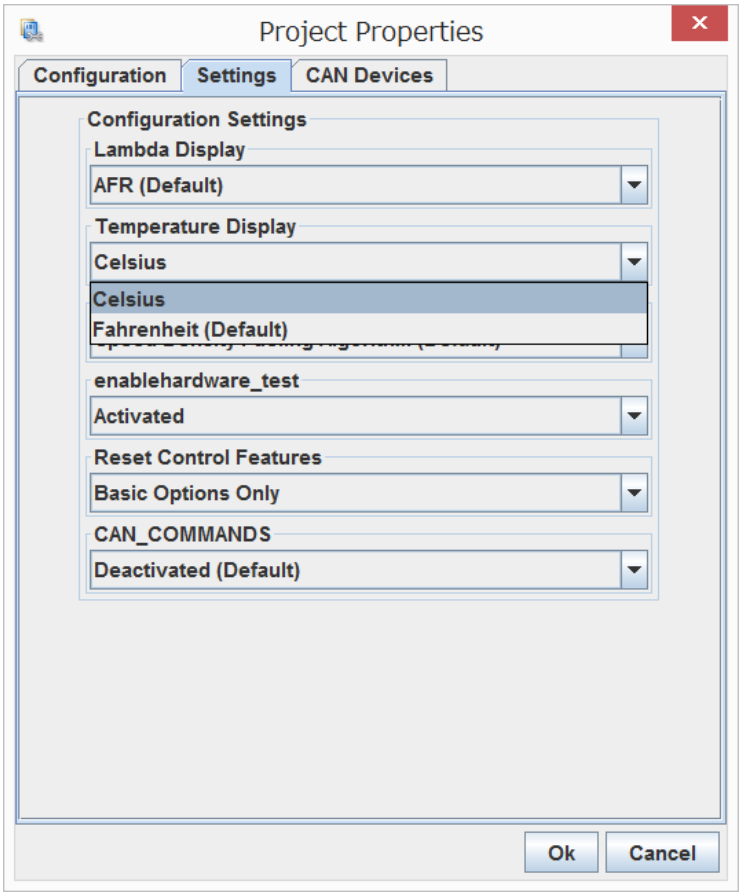


1.6.3　Settings タブ

Temperature Display

プロジェクトプロパティダイアログの'Settings'タブに、'Temperature Display'という項目があります。この項目で温度表示の単位を以下の二つから選択出来ます。

- Fahrenheit (Default)
- Celsius



註）TunerStudio は米国製のソフトウェアの為 Fahrenheit がデフォルト設定となっていますが、日本では一般的ではありませんので Celsius に変更する事を推奨します。

Fueling Algorithms

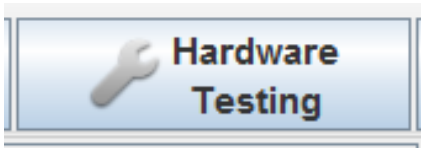
PJSC では回転数軸と負荷軸から燃料噴射量を求める 3D マップ制御を、燃調制御方式として採用しています。負荷軸にどのセンサー出力を用いるかによって、以下の 2 種類から燃調方式（Fueling Algorithm）を選択します。

・Speed Density Fueling Algorithm (Default)：負荷軸に MAP（Manifold Air Pressure、インテークマニホールド圧力）センサー出力を用いる場合、この方式を選択して下さい。

・Alpha-N Fueling Algorithm：負荷軸に TPS（Throttle Position Sensor、スロットルポジション）出力を用いる場合、こちらの方式を選択して下さい。

Enable Hardware Test

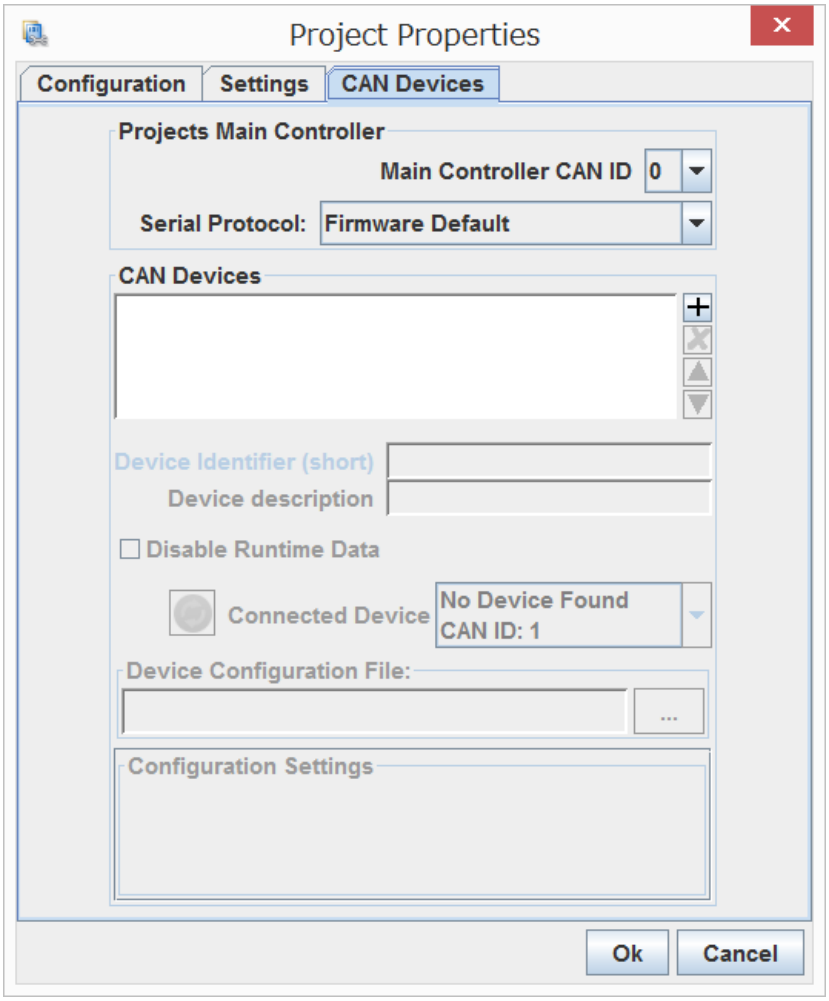
エンジン停止時にインジェクター動作テストを行うモードを有効または無効にするか選択する項目です。デフォルトではハードウェアテストは無効になっています。



CAN_COMMANDS

デフォルト設定は CAN 通信機能を無効にしています。

1.6.4 CAN Devices タブ



第２章 ハードウェア

2.1　　ハードウェア仕様

2.1.1　マイコンボード

PJSC は Arduino Mega2560 R3 の小型互換ボードである Epalsite 製 Meduino Mega 2560 Pro Mini R3（http://wiki.epalsite.com/index.php?title=Mega2560_Pro_Mini）をコントローラーとして使用します。

2.1.2　通信インターフェース

PJSC で燃調チューニングを行う際、Tuner Studio をインストールした PC と PJSC を通信可能なインターフェースで接続します。PJSC は USB を標準インターフェースとしてサポートしています。

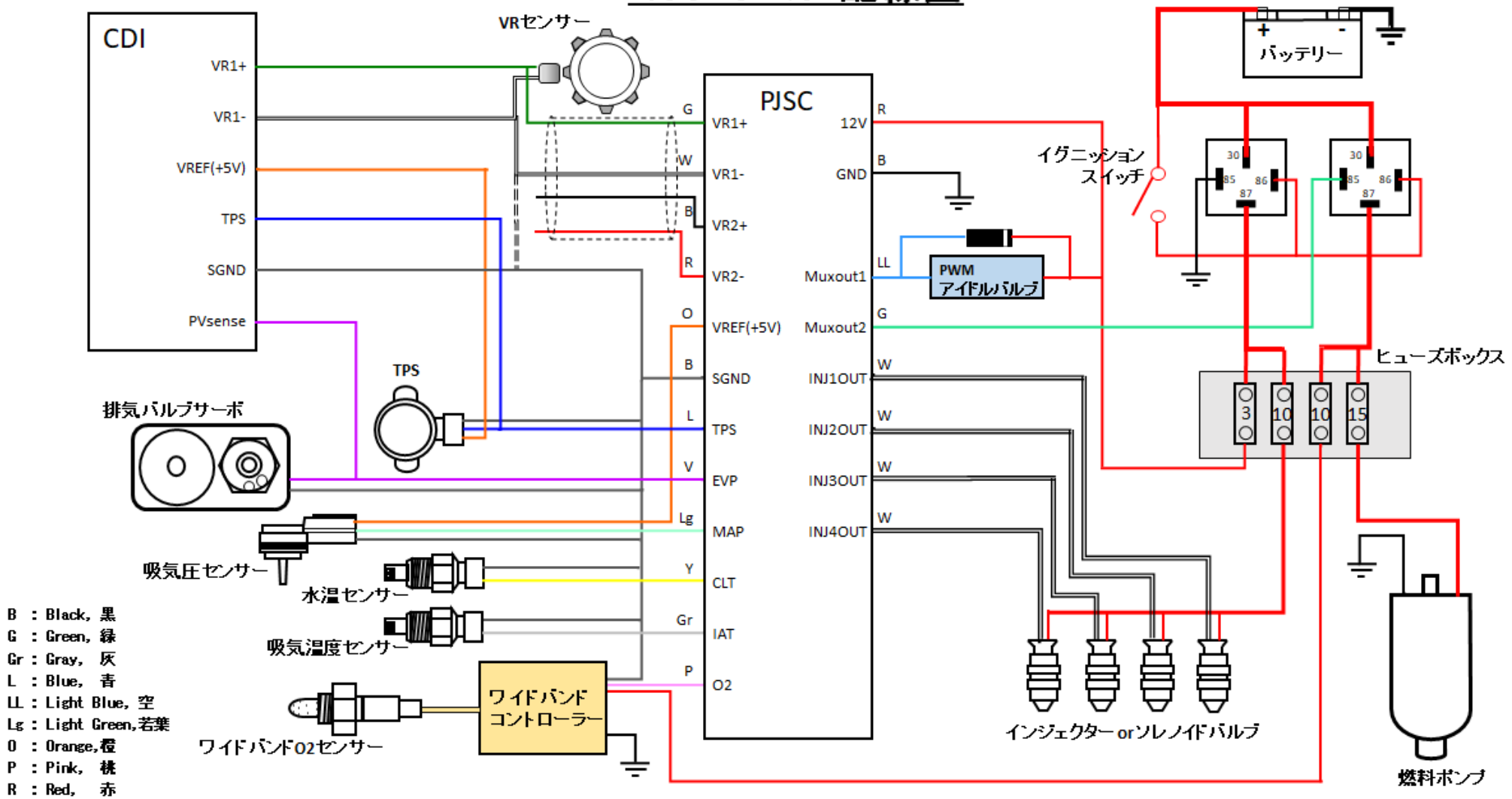
USB mini-B ケーブルで、PC と PJSC を接続して下さい。

オプションとして Bluetooth も利用可能です。Bluetooth で PJSC と PC を接続する方法は、別項で解説します。

2.1.3　配線

PJSC を車体に設置する為に、各種センサーと PJSC を接続する配線を準備する必要があります。PJSC と各種入出力の結線は、下図を参照して下さい。

PJSC ver1.01配線図



B : Black, 黒
G : Green, 緑
Gr : Gray, 灰
L : Blue, 青
LL : Light Blue, 空
Lg : Light Green,若葉
O : Orange,橙
P : Pink, 桃
R : Red, 赤
V : Violet,紫
W : White, 白
Y : Yellow,黄

2.1.2　入力信号

PJSC はセンサーからの信号入力に応じて、燃料噴射量を制御する為の信号を出力します。

クランクセンサー

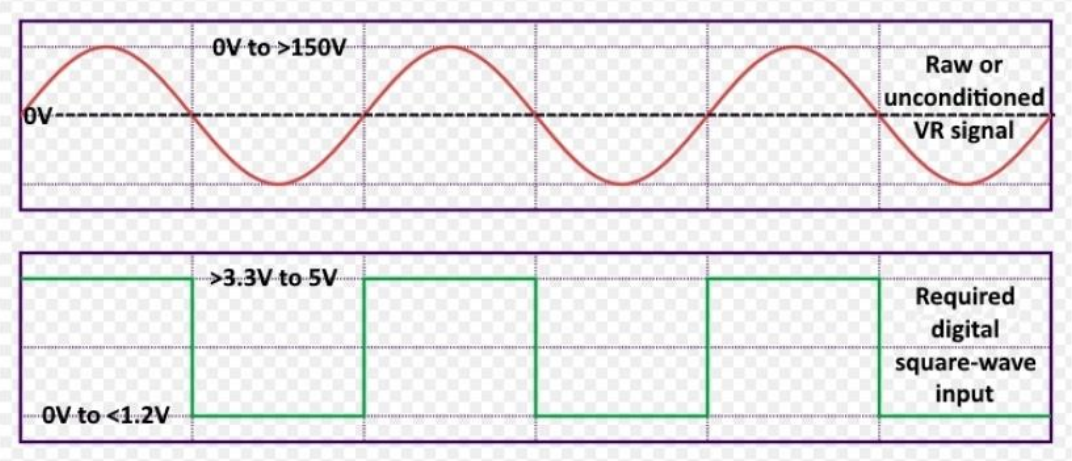
クランクセンサーは PJSC を動作させる上で、最も重要な信号です。Arduino がクランク角速度とクランク角を正しく認識する為に、クランクセンサー信号はクランク（又はカム）の回転に同期した 0-5V 矩形波信号に変換して入力する必要があります。ホールセンサー或いはフォトセンサーの信号は矩形波なので、これらをクランクセンサーと使用しており且つ信号レベルが 0-5V であればセンサー出力をそのまま Arduino に入力する事が出来ます。

もしカムセンサーを使用せずクランクセンサーのみ Arduino に入力するのであれば、クランクセンサーから回転数とクランク角を検出する為にクランクホイールはミッシングトゥースホイールにしなければなりません。Speeduino で動作実績のあるミッシングトゥースは、4-1、12-1、36-1、60-2 の 4 種類です。

カムセンサーを追加すれば、クランクホイールがミッシングトゥースでなくてもクランク角検出が可能になります。インジェクター噴射方式をシーケンシャルインジェクションとするには、カムセンサーが必要になります。クランクとカム両方にホイールとセンサーを設置するデュアルホイール方式を表現する場合、カムホイールの歯数を表す”/x”を追加します。例えば”60-2/1”はクランクホイールの歯数が 60（ミッシングトゥース 2 を含む）で、カムホールの歯数が 1 サイクル当たり 1 歯という事になります。

クランクセンサーとして VR（Variable Reluctance）センサーを使用する事も可能です。Speeduino オフィシャルボードでは VR センサー信号を矩形波信号に変換する回路を搭載しておらず、別途変換回路を追加しなければなりません。しかし PJSC はバイクに特化している為はじめから変換回路を搭載しており、VR センサー信号をそのまま入力する事が可能です。

クランクセンサー信号入力に適した矩形波と VR 信号



TPS

TPS には 3 線式のポテンショナー（可変抵抗）タイプのセンサーが必要です。稀に 2 線式の On/Off タイプのセンサーを装着しているスロットルがあります。また 3 線式でもポテンショナータイプではない物があるので注意が必要です。

TPS はスロットル開度を PJSC に認識させる為に、信号レベル可変のアナログ信号を出力します。通常はリファレンス電圧として 5V とグランドが TPS に供給され、スロットル開度に応じたポテンショナーの分圧比によりスロットル開度が小さい時は低い電位が、スロットル開度が大きくなると高い電位が TPS 信号として出力されます。

もし TPS センサーのピン割当てが判らない場合は、テスターで抵抗値を測定して調べる事が出来ます。

(1) テスターを抵抗値測定モードに設定します
(2) 任意の 2 ピン間の抵抗を測定しながらスロットルを全閉から大きく開け、抵抗値を記録します。これを全ての 2 ピンの組合せで行います。
(3) スロットル操作をしても抵抗値に大きな変化が無かった組合せが、リファレンレンス電圧（5V、グランド）を供給するピンです。
(4) 残りの 1 ピンが信号ピンとなります。
(5) 信号ピンと組合わせて抵抗値を測定した際、スロットルを開けると抵抗値が増えるピンがグランドピンです。逆に全閉で抵抗値が大きく、スロットルを開けると抵抗値が小さくなるピンが 5V ピンです。

3 線式の TPS が最も使い易くシンプルなセンサーです。もし使用する TPS のピン数が 3 ピンより多い場合、車両サービスマニュアルの結線図を参照して 5V、グランド、信号に該当するピンを探して下さい。

註）TPS 信号を純正 ECU（或いは社外 ECU）と PJSC で共用する場合 PJSC の 5V 信号は接続せず、センサーのグランドと TPS 信号線を分岐して純正 ECU、PJSC それぞれのセンサーグランド、TPS 信号入力に接続して下さい。

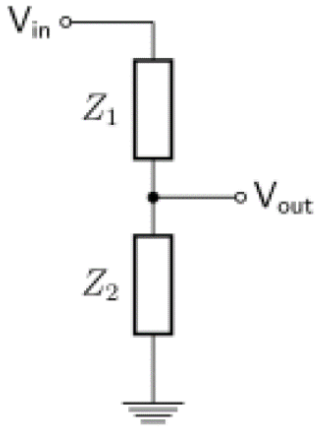
MAP（Manifold Air Pressure）

Speeduino が幾つかの MAP センサーはについてキャリブレーションデータをプリセットで用意しており、PJSC もそのまま同じプリセットデータを持っています。プリセットデータがある MAP センサーを使用する場合は、Calibration MAP ダイアログ（2.4.1 項を参照）のプルダウンメニューから選択する事が出来ます。

プリセットデータが無い MAP センサーを使用する場合は、キャリブレーションデータを入力する事でそのセンサーも使用可能となります。

温度センサー（CLT、IAT）

PJSC は冷却水温センサー（CLT、Coolant Temperature sensor）と吸気温度センサー（IAT、Intake Air Temperature sensor）を燃調制御に利用する事が出来ます。使用出来るセンサーは、2 線式のサーミスターとなります。またデフォルトのバイアス抵抗値は 2490Ωとなります。



サーミスターを用いた温度検出回路は、上図の様な抵抗分圧回路を構成しています。Vin には+5V 電圧を入力します。Z1 はバイアス抵抗 2490Ωで、Z2 がサーミスターに該当します。GND は PJSC のシグナル GND に接続します。

この回路では水温、吸気温に応じて出力電圧 Vout が遷移します。Vout を Arduino の ADC 入力ピンに接続し、Arduino に温度を認識させます。

排気ガス酸素濃度センサー（EGO、Exhaust Gas Oxygen Sensor）

PJSC では排気ガス中の酸素濃度を測定する O2 センサーを使用して、燃調補正をする事が可能です。また TunerStudio Ultimate では O2 センサー出力を採り込んで、燃調のオートチューンを行う事も可能です。

O2 センサーを使用する場合、TunerStudio メインメニューの Tool＞Calibrate AFR Table で表示される AF 比キャリブレーションダイアログ内で該当するセンサーを選択して下さい。

ナローバンド O2 センサー

PJSC はナローバンド O2 センサーの信号を直接読み込む事が出来ます。TunerStudio は殆どの標準的なナローバンド O2 センサーの非線形な 0-1V 出力て自動的にキャリブレーションを行います。キャリブレーション実施後は、AFR テーブル（Tuning＞AFR Table）で指定された空燃比を目標値として燃調を補正する為に、ナローバンド O2 センサーを使う事が出来ます。

註）ナローバンドセンサーは空燃比をキャタライザーが効率的に機能するストイキメトリ空燃比（ラムダ 1.0）に合わせる事を目的に設計されています。その為、希薄燃焼モードやパワー空燃比に合わせるチューニングには向いていません。

ワイドバンド O2 センサー

ワイドバンド O2 センサーはナローバンド O2 センサーよりも広範囲の空燃比を検出する事が出来ます。センサーとコントローラーにもよりますが、およそ 10:1 から 20:1 の空燃比（ラムダ 0.7 から 1.3）を検出可能です。

PJSC にワイドバンド O2 センサーの出力を直接入力する事は出来ません。ワイドバンド O2 センサーにはヒーターのコントロールとセンサー信号を増幅するアンプを搭載したコントローラーが必要で、コントローラーが出力する 0-5V のアナログ信号を PJSC に入力します。TunerStudio メインメニューの Tool＞Calibratte AFR sensor で表示される AFR calibration ダイアログにコントローラーのメーカーとモデルのリストが表示されるので、使用するコントローラーに該当するものを選択して下さい。

コントローラー出力信号が一般的な線形特性であれば、'Custom Linear WB'をリストから選択し計測可能範囲の最少 AFR 値と最大 AFR 値をを示す電圧を入力すれば、リストに無いモデルのコントローラーでも使用可能です。

コントローラー出力信号が非線形の場合、リストから'Custom inc File'を選択して信号特性プロファイルを記述した INC ファイルを TunerStudio に読み込ませて下さい。

PJSC はワイドバンド O2 センサー信号を使って、空燃比が AFR テーブル（Tuning＞AFR Table）に指定された値となるように燃調を補正する事が出来ます。TunerStudio メニューの Tuning＞AFR/O2 で表示される AFR/O2 設定ダイアログで補正の詳細な設定をする事が可能です。

燃調補正、オートチューン機能を有効にする場合、ワイドバンド O2 センサーを使用する事を推奨します。

排気バルブポジションセンサー

1980 年代後半から 1990 年代にかけて販売された日本製 2 ストロークバイクには、排気ポートにバルブを備えた物があります。その中でもサーボモーターにポテンショナーを内蔵してバルブポジションを検出し、排気ポートが開くタイミングを可変とするタイプ－例えば RC バルブ（ホンダ）、YPVS（ヤマハ）、AETC（スズキ）のポジション信号を採り込んでログに表示する事が可能です。

ポテンショナー方式の排気バルブは TPS と同様、３線式（5V、グランド、バルブポジション）の可変抵抗が用いられています。通常 5V は純正 ECU から供給されているので PJSC には接続せず、センサーのグランドとバルブポジション信号線を分岐して純正 ECU と PJSC のそれぞれの入力に接続して下さい。

註）排気バルブポジションセンサーのグランドを PJSC に接続しないと純正 ECU と PJSC のグランド電位に差が生じ、排気バルブポジション信号が不安定になる場合があります。その様な場合、純正 ECU が排気バルブポジションを正しく認識出来ず、過電流を流してモータードライバーが焼損する恐れがあります。

2.1.3　出力

インジェクター

PJSC のインジェクタードライバーは電流飽和型で（PWM ではありません）、ハイインピーダンスインジェクターの使用を想定しています。このタイプのインジェクタードライバーはバッテリーの電圧をそのままインジェクターに印加します。ハイインピーダンスインジェクターの抵抗値は通常 8Ω以上であり、抵抗値がこれより低いインジェクター（ローインピーダンスインジェクター）は PJSC では使用出来ません。ローインピーダンスインジェクターを使用する場合は、過電流によるボードへのダメージを防ぐ為にインジェクターと直列に抵抗を接続する必要があります。接続する抵抗の抵抗値と定格電力は、オームの法則から算出します。

PJSC はインジェクター出力 1ch 当り、2 本のハイインピーダンスインジェクターを並列に接続する事が出来ます。

また PJSC 独自の機能としてインジェクター出力を周波数固定の PWM 信号にし、90 年代の 2 ストロークオートバイに多く使用されていたエアソレノイドを駆動する PJSC モードを使用する事が出来ます。

インジェクター出力は定格電流 7A の MOS-FET を使用していますので、オートバイで使用されている殆どのソレノイドバルブを駆動する事が可能です。ソレノイドバルブをポンプで加圧した燃料ラインに挿入し、これを PWM で駆動して燃料をスロットルボア内に噴射する事で、キャブレターを使用した車両でも擬似的なインジェクションの様に燃調チューニングを行う事が可能です。これが PJSC（Pump Jet Solenoid Controler）の名前の由来となったポンプジェット（Pump Jet）です。

ポンプジェット用のソレノイドバルブとしては、エアソレノイドで使用されていたソレノイドを使用する事が可能です。この場合ソレノイドの抵抗値が 20-30Ω なので、インジェクター出力 1ch 当り 4 本まで並列に接続する事が可能です。

またバルブ開閉には 5ms 程度要する為、PWM 周波数は 10-20Hz 程度が適切です。但しこれらの仕様はソレノイドバルブによって異なりますので、PJSC に接続する前にバルブの仕様を確認して仕様に合った値を設定して下さい。

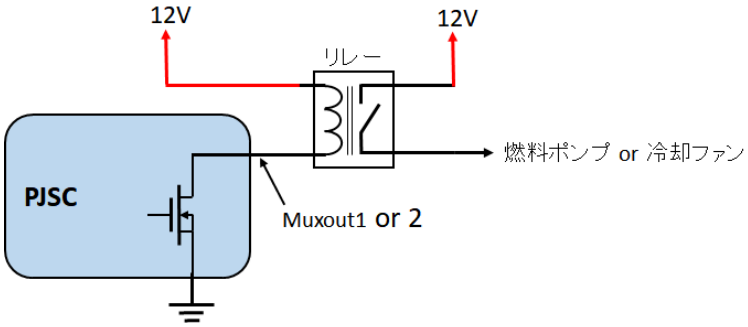
MUX Outputs

PJSC は以下の 6 つの用途の中から選択して使用する為の汎用出力を 2ch 持っています。これは Arduino のデジタル出力で PJSC ボード上の MOS FET（SI4900DY）をスイッチングしており、2 A までの負荷を直接駆動する事が出来ます。それ以上の電流が流れる負荷を駆動する場合は、リレーを使用して下さい。

MUX Output の機能の選択方法は『**4.14 章 MUX output セッティング**』を参照して下さい。

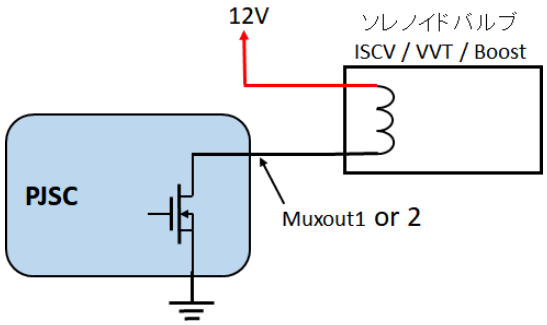
- 燃料ポンプ ON/OFF
- 冷却ファン ON/OFF

  燃料ポンプ及び冷却ファンは、下図の様にリレーを介して接続して下さい。



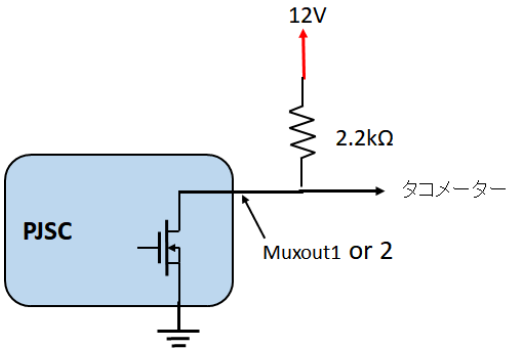
- アイドルスピードコントロールバルブ（PWM または ON/OFF）
- ブーストコントロール
- VVT コントロール

  ISCV、ブーストコントロール、VVT コントロール用のソレノイドバルブは下図の様に、直接 Muxout に接続して下さい。但しソレノイドバルブの抵抗値が 10Ω未満の場合は、Muxout の定格を超える電流が流れる可能性がありますので、直接接続しないで下さい。



- タコメーター信号出力

  タコメーター信号を出力する場合、下図の様にプルアップ抵抗 2.2kΩを介して 12V と Muxout を接続しタコメーターと接続して下さい。但し CDI 用タコメーターは-200V の点火信号を必要とする為、PJSC で駆動する事は出来ませんのでご注意願います。



2.2　PJSC ボード

2.2.1　概要

PJSC ボードは Speeduino v0.4 ボードをベースとして入出力をオートバイの燃調制御に必要な物だけに絞り、且つ点火出力も省く事で可能な限りコンパクトな筐体に収める事を目標として設計されました。

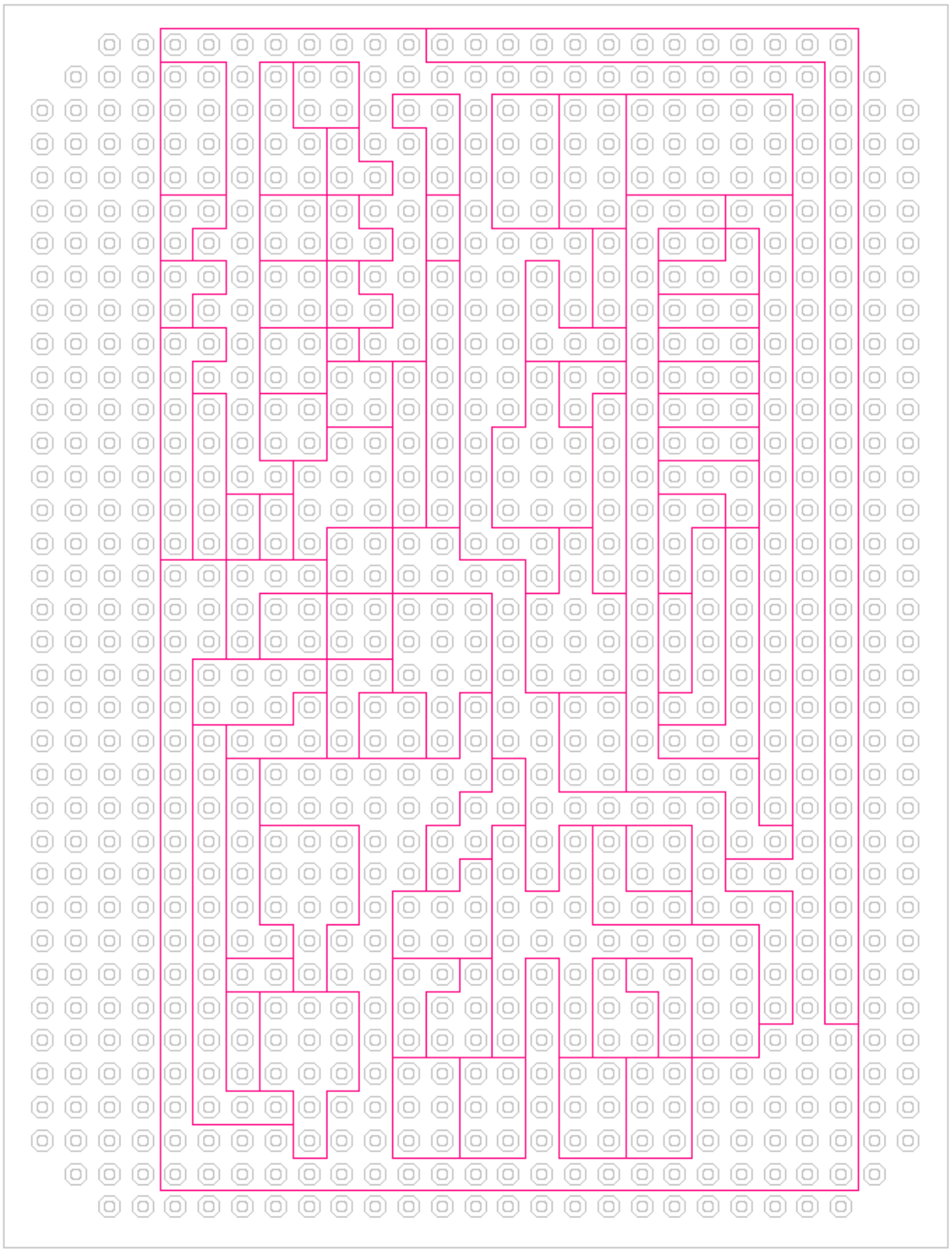
また小ロット生産に対応する為あえて専用基板を起こさず、十字ユニバーサル基板で回路基板を作成する事を前提に部品配置を決めています。十字ユニバーサル基板はユニバーサル基板の片面にスルーホールとスルーホールを繋ぐ格子状のパターンがあり、このパターンを目的の回路を構成する様にカットする事で容易に回路基板を作成出来るようにした物です。エッチング基板の様に薬剤を扱う必要が無いのが利点と言えます。しかし製作数が多くなるとパターンカットの手間が掛るので、極限られた数（数個から十数個程度）の回路の試作に向いています。

十字ユニバーサル基板の取り扱いは国内では秋月電子が行っており、PJSC はＢタイプ（95mm x 72mm）ガラスコンポジット基板を用いて製作出来るようにパターンが設計されています。

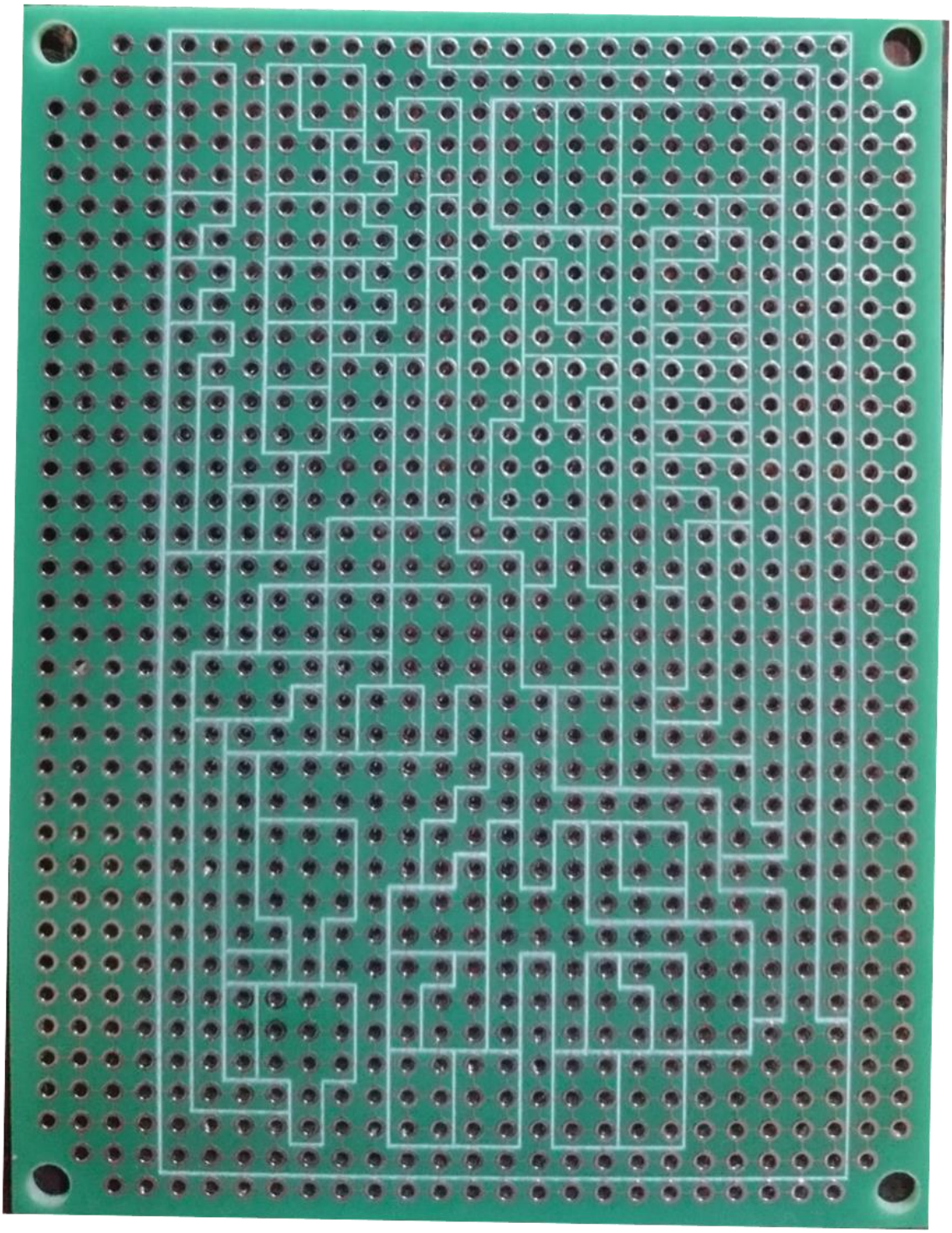
十字配線ユニバーサル基板　Ｂタイプ（９５ｘ７２ｍｍ）　ガラスコンポジット
http://akizukidenshi.com/catalog/g/gP-09794/

ソースコードと共に配布されている PJSC パターンファイル（PJSC.dxf）の『十字ユニバーサル基板カットパターン』レイヤーの図面が、カットするラインを表したものです。このパターンを十字ユニバーサル基板の格子状パターン面に印刷または転写し、カットラインをカッター等でカットする事で PJSC の回路基板が作成出来ます。カットした箇所は確実に格子パターンがカットされて導通しなくなっている事を、テスター等で確認して下さい。導通がある状態だと PJSC は正常に動作せず、最悪はエンジンを破損する事になりますのでくれぐれも注意して下さい。

PJSC 十字ユニバーサル基板カットパターン



カットパターンに倣ってカットした十字ユニバーサル基板



2.2.2 PJSC ボードの機能

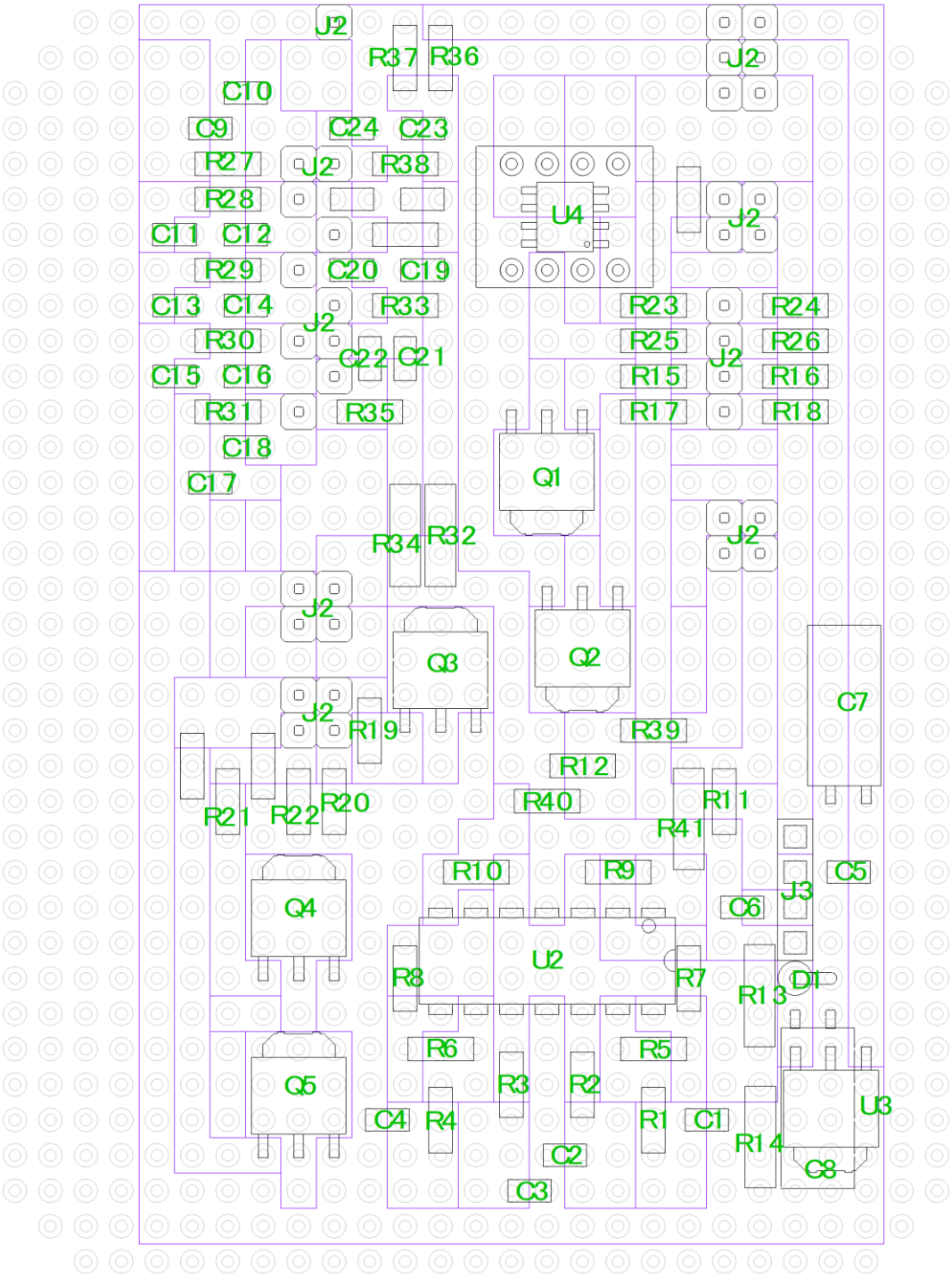
PJSC ver1.0 ボードは以下の機能が実装されています。

- インジェクター／ソレノイドドライバー 4 チャンネル
- CLT、IAT、MAP、TPS、BARO、O2、排気バルブポジションセンサー入力 7 チャンネル
- VR（ピックアップ）入力 2 チャンネル
- MUX 出力 2 チャンネル

2.2.3 部品配置

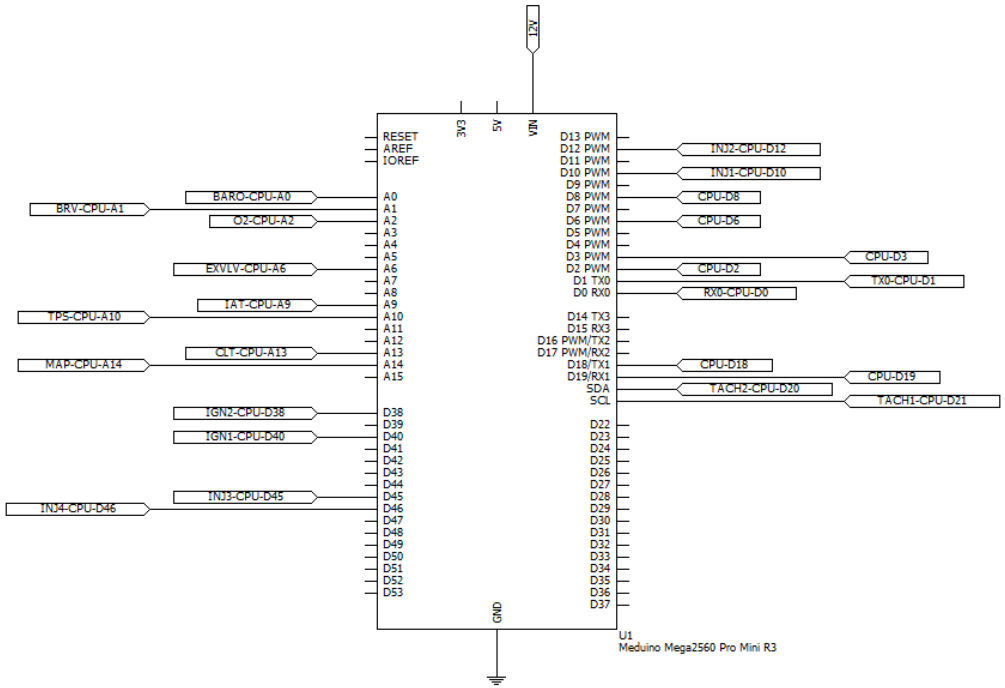
PJSC パターンファイルの『部品配置』レイヤーの図面が部品配置を表しています。これを参照し、必要な部品を十字ユニバーサル基板から作成した PJSC 基板に半田付けする事で PJSC ボードを作成出来ます。

部品配置図

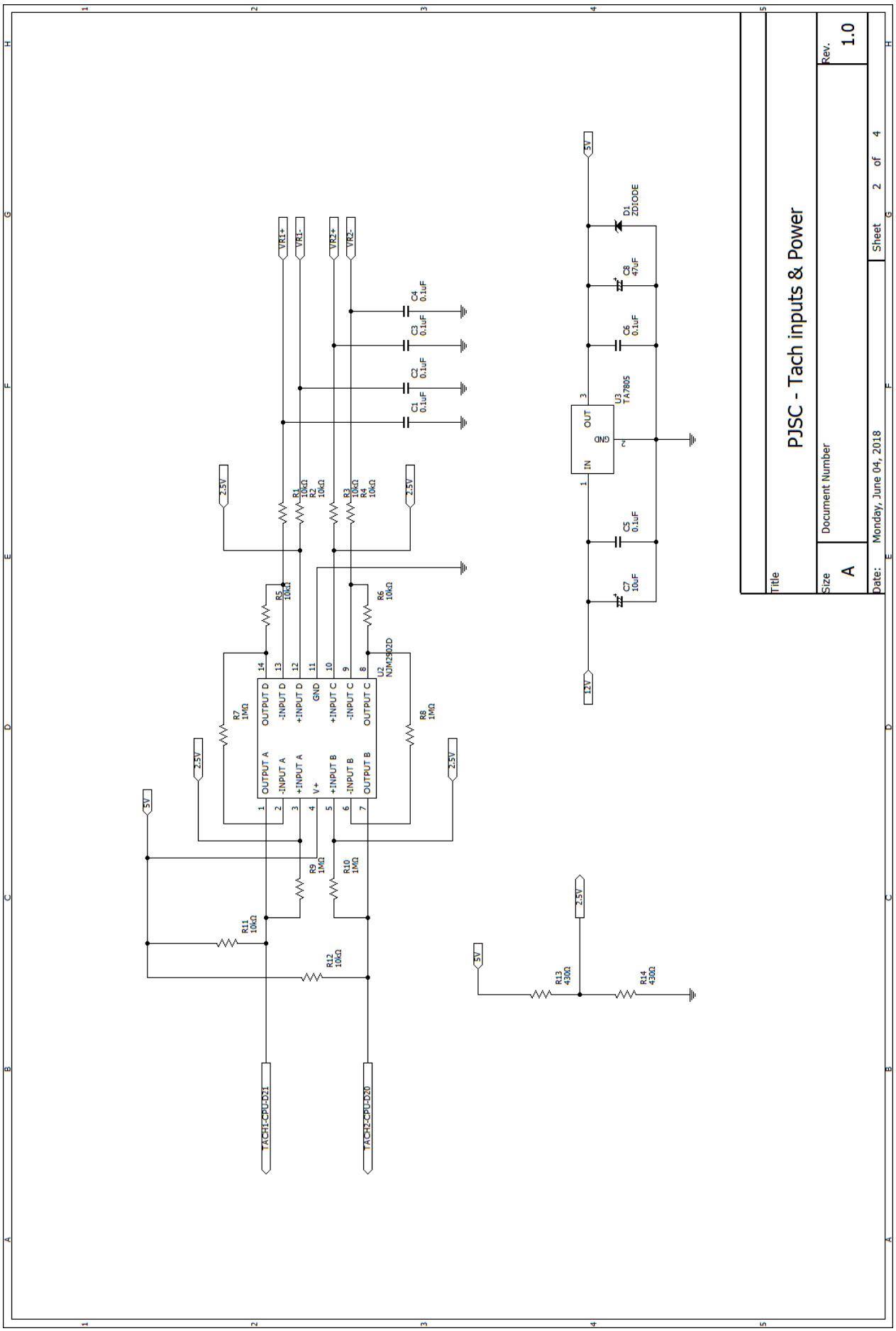


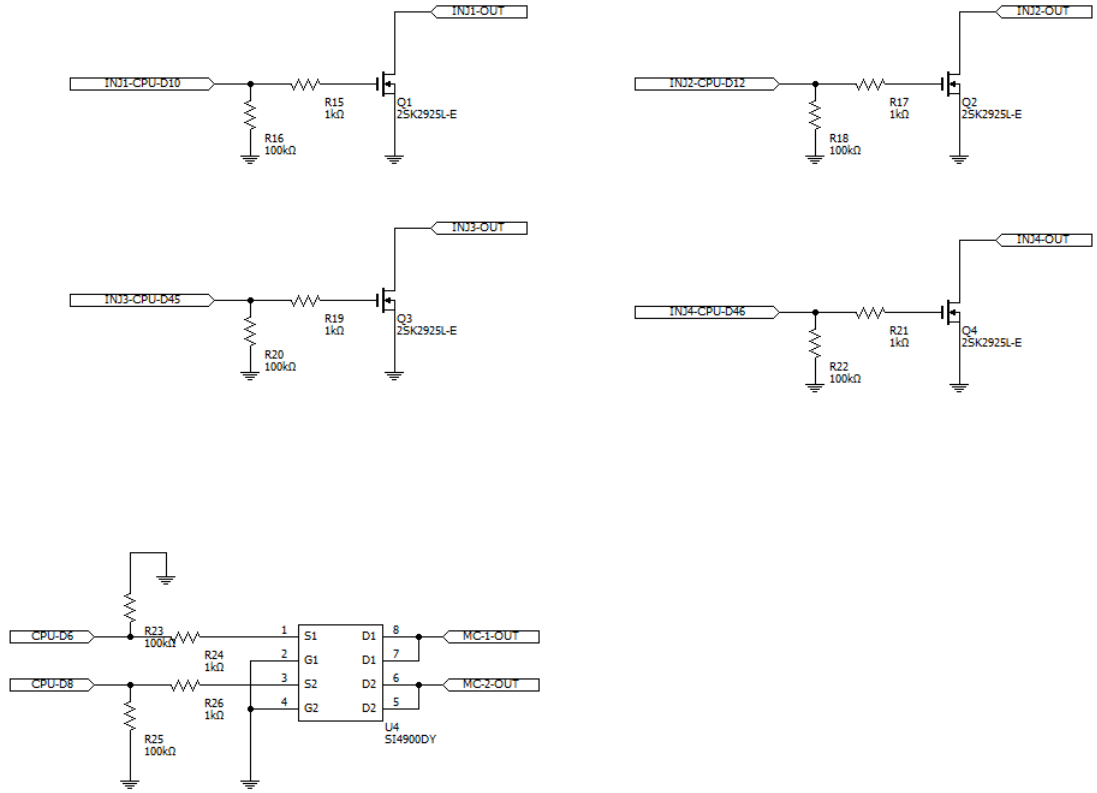
信号の入出力は基板上に直接リード線を半田付けします。ユニバーサル基板を使用した上で、基板、筐体を出来るだけ小さくする為の処置です。入出力に基板用コネクターを用いると、各入出力素子からコネクターまでのパターンが必要になり、ユニバーサル基板では基板面積を著しく大きくする要因となります。

筐体内での架空線の不要な揺動、接触や防水性、防塵性を確保する為に、基板を収めたケース内にはガラス樹脂等、絶縁性の樹脂を充填する事を推奨します。

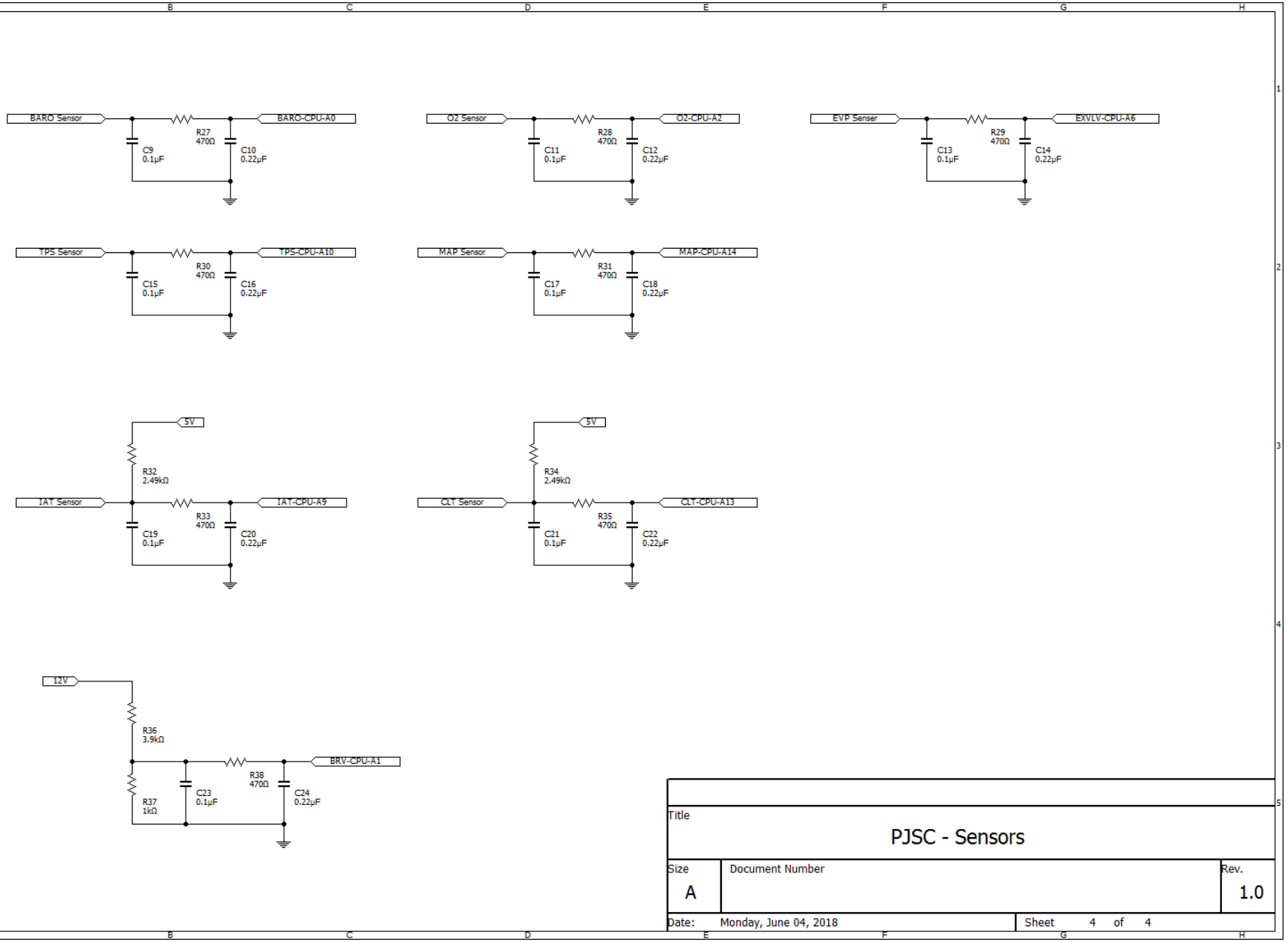


Title	PJSC - Meduino Mega2560 Pro Mini	
Size: A	Document Number	Rev.: 1.0
Date: Monday, June 04, 2018	Sheet 1 of 4	


PJSC - Tach inputs & Power
Title
Size
A
Document Number
Rev.
1.0
Date:
Monday, June 04, 2018
Sheet
2
of
4
VR1+
VR1-
VR2+
VR2-
C1 0.1uF
C2 0.1uF
C3 0.1uF
C4 0.1uF
R1 10kΩ
R2 10kΩ
R3 10kΩ
R4 10kΩ
R5 10kΩ
R6 10kΩ
R7 1MΩ
R8 1MΩ
R9 1MΩ
R10 1MΩ
R11 10kΩ
R12 10kΩ
2.5V
5V
U2 NJM2902D
OUTPUT D
-INPUT D
+INPUT D
GND
+INPUT C
-INPUT C
OUTPUT C
OUTPUT A
-INPUT A
+INPUT A
V+
+INPUT B
-INPUT B
OUTPUT B
TACH1-CPU-D21
TACH2-CPU-D20
12V
U3 TA7805
IN
GND
OUT
C5 0.1uF
C6 0.1uF
C7 10uF
C8 47uF
D1 2DIODE
R13 430Ω
R14 430Ω



Title	PJSC - Injector & Middle current general output	
Size: A	Document Number	Rev.: 1.0
Date: Monday, June 04, 2018	Sheet 3 of 4	


BARO Sensor
R27
470Ω
BARO-CPU-A0
C9
0.1µF
C10
0.22µF
O2 Sensor
R28
470Ω
O2-CPU-A2
C11
0.1µF
C12
0.22µF
EVP Senser
R29
470Ω
EXVLV-CPU-A6
C13
0.1µF
C14
0.22µF
TPS Sensor
R30
470Ω
TPS-CPU-A10
C15
0.1µF
C16
0.22µF
MAP Sensor
R31
470Ω
MAP-CPU-A14
C17
0.1µF
C18
0.22µF
5V
R32
2.49kΩ
IAT Sensor
R33
470Ω
IAT-CPU-A9
C19
0.1µF
C20
0.22µF
5V
R34
2.49kΩ
CLT Sensor
R35
470Ω
CLT-CPU-A13
C21
0.1µF
C22
0.22µF
12V
R36
3.9kΩ
R38
470Ω
BRV-CPU-A1
R37
1kΩ
C23
0.1µF
C24
0.22µF
Title
PJSC - Sensors
Size
A
Document Number
Rev.
1.0
Date:
Monday, June 04, 2018
Sheet
4
of
4

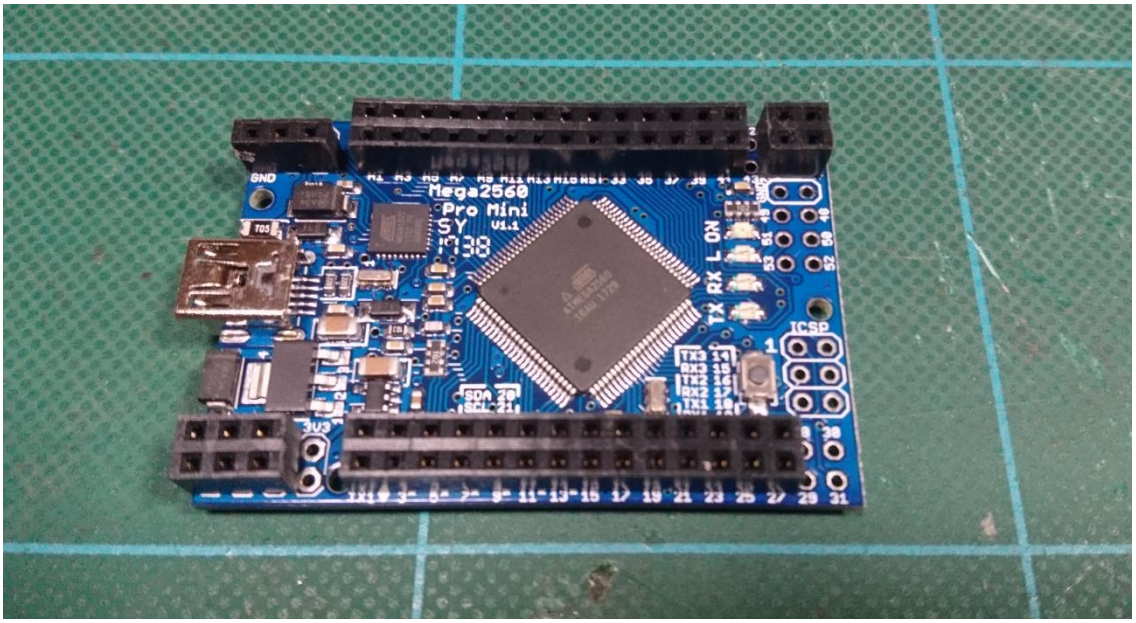
2.2.4　アッセンブリ

必要な部品と対応する部品番号は、BOM（Bill of Material）を参照して下さい。2.3.3 の部品配置図に記載されている部品番号を参照して、全ての部品を基板へ半田付けします。その際、部品のリード線を通すスルーホールの位置を間違えない様十分に注意して下さい。スルーホールの位置を間違えると、PJSC は正しく動作しません。

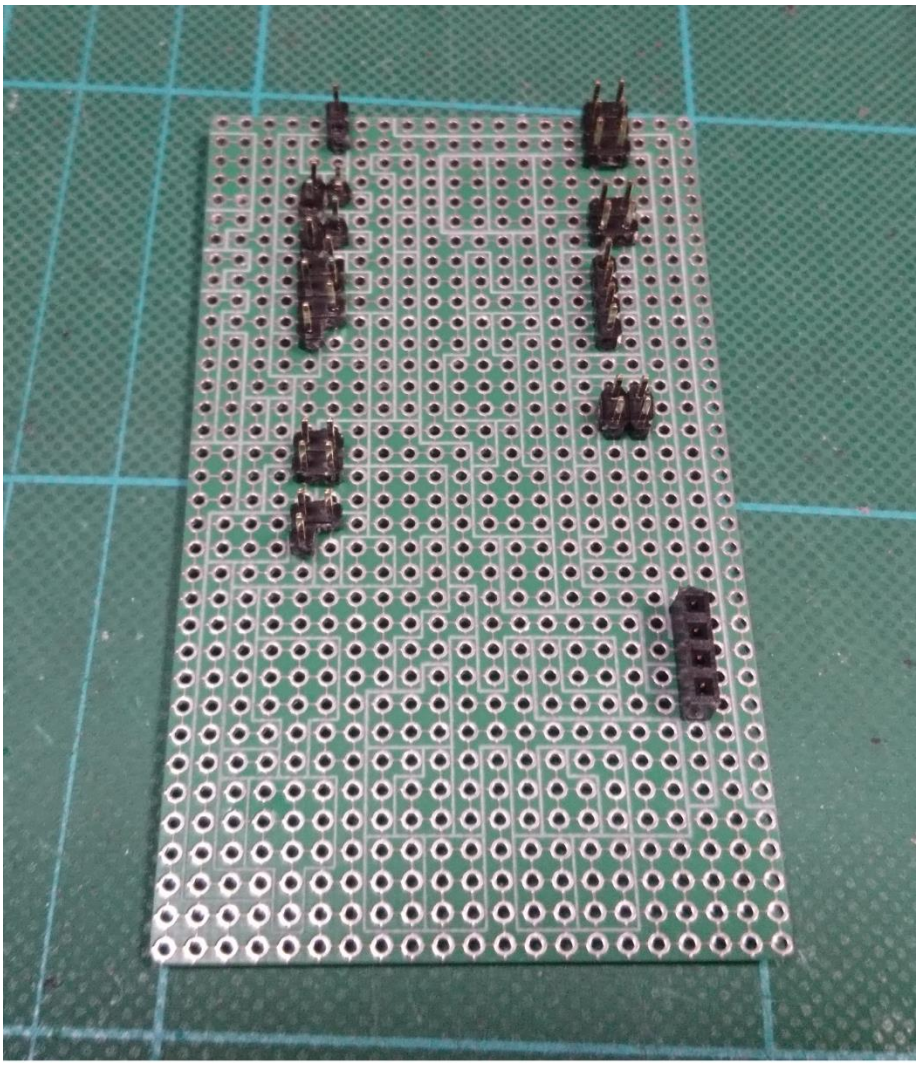
十字ユニバーサル基板は専用設計された PCB と異なり、部品配置を示すシルク印刷はありません。カットパターンと部品配置図の位置関係から、リード線を通すスルーホールの位置を確認して下さい。

部品の半田付けは、以下の順番に行う事を推奨します。

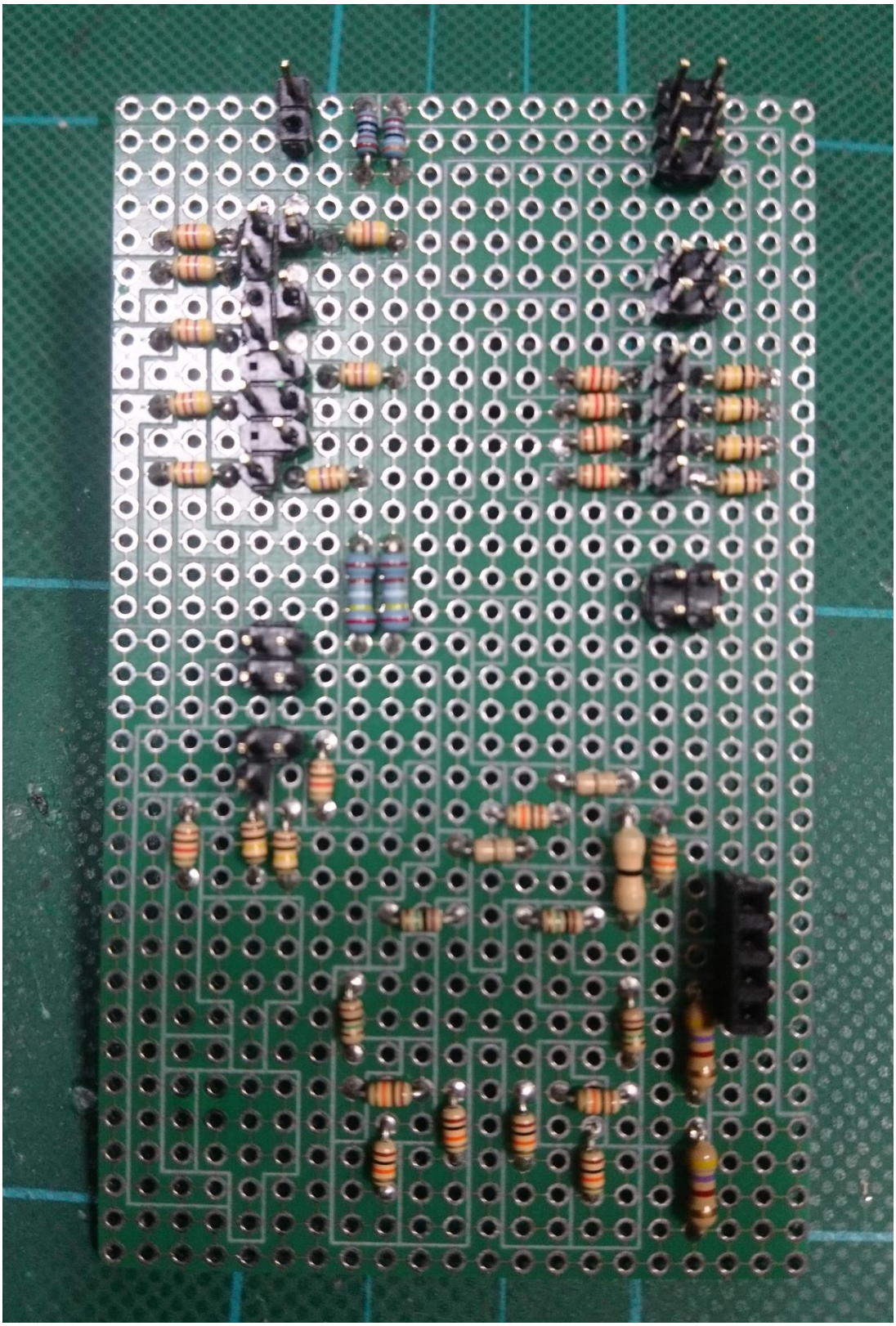
1. Meduino Mega2560 Pro Mini へピンソケット（J1）を半田付けします。ピンソケットは必要な長さにカットしてから、半田付けして下さい。



2. PJSC ボードにピンヘッダー（Meduino Mega2560 Pro Mini を接続する為のヘッダー、J2）、Bluetooth シリアルモジュール用のソケット（J3）を半田付けします。

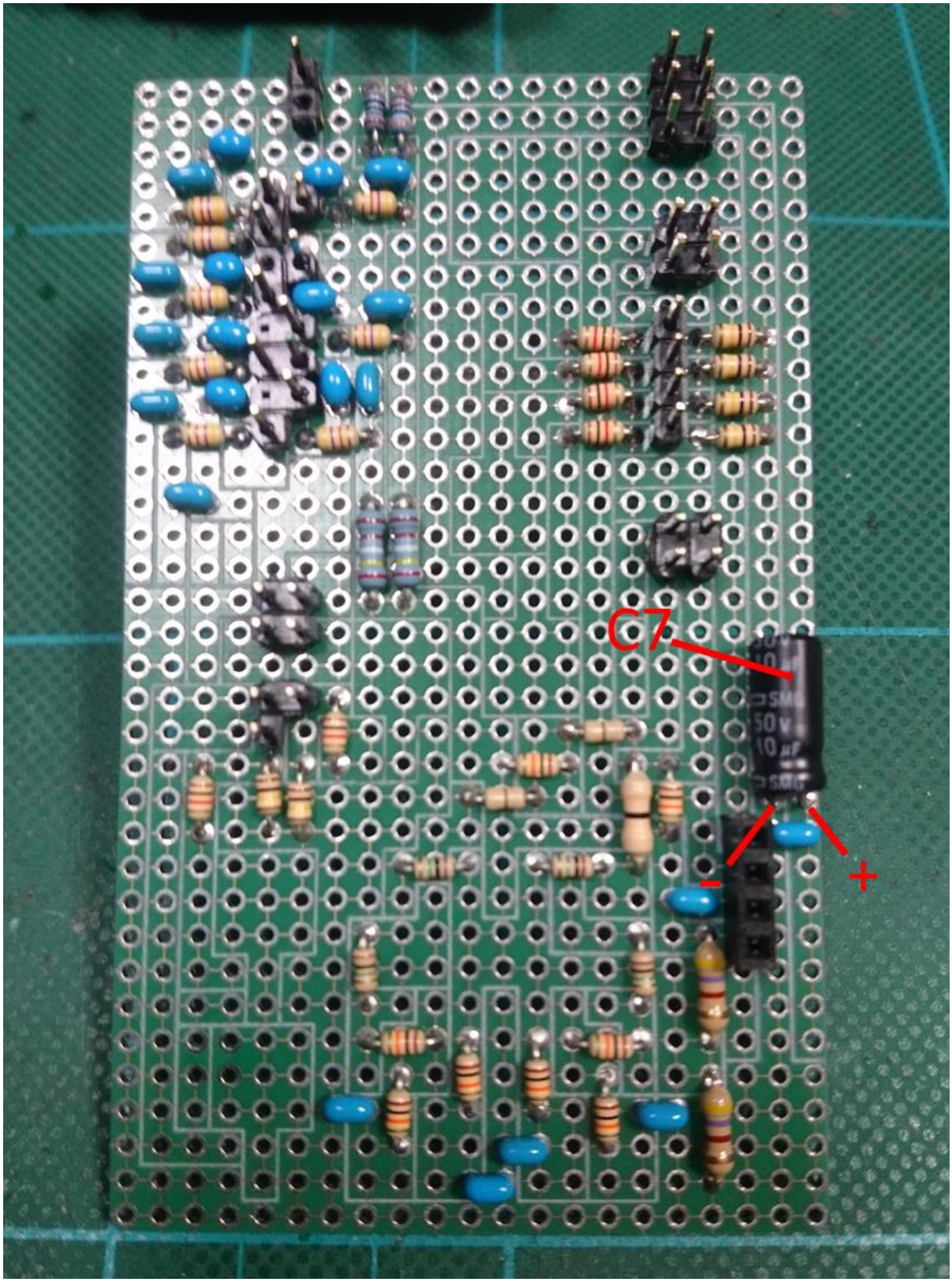


3. 抵抗



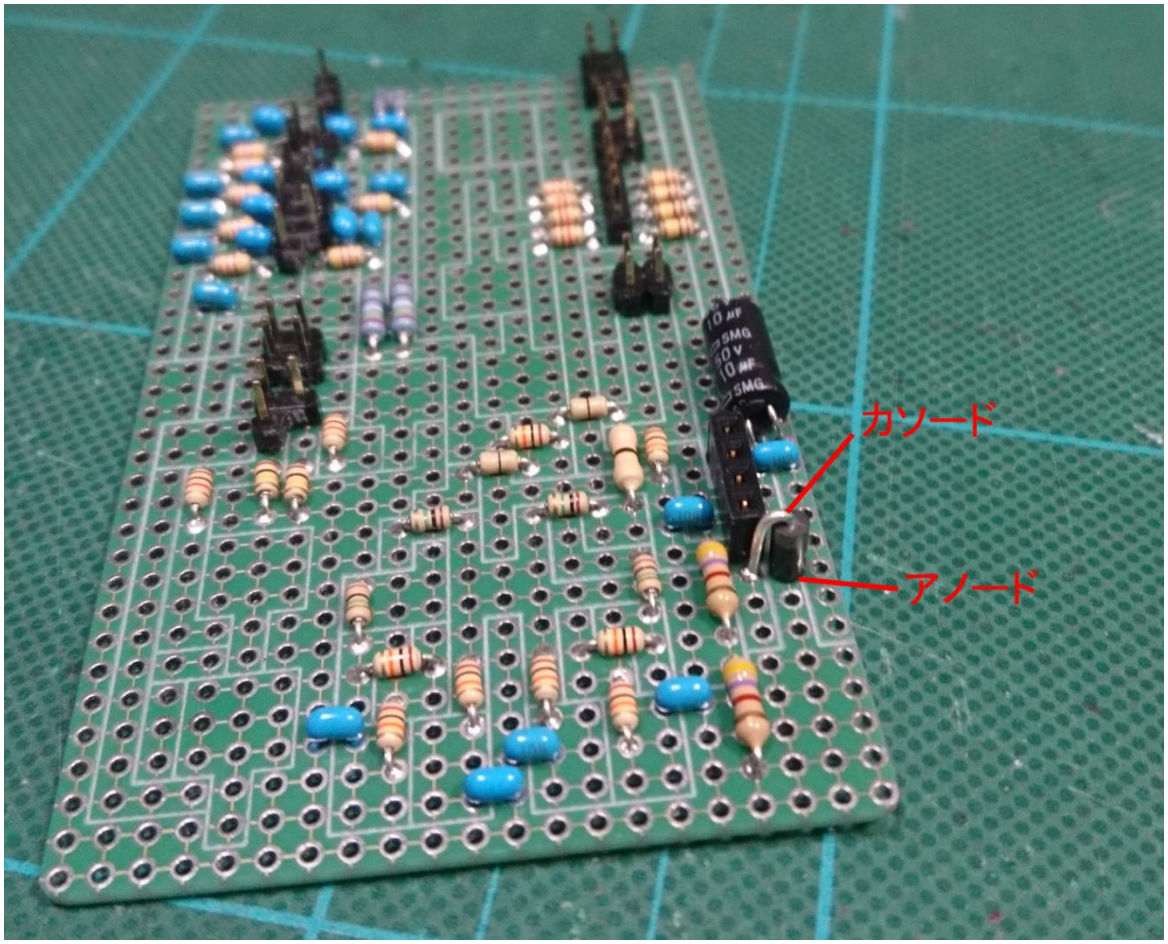
4. コンデンサー

(a) C7、C8 は電解コンデンサーで極性があります。パッケージに"-"が記載されている方のリード線を、グランドパターンに半田付けして下さい。但し、C8 はこの時点では半田付けしません。



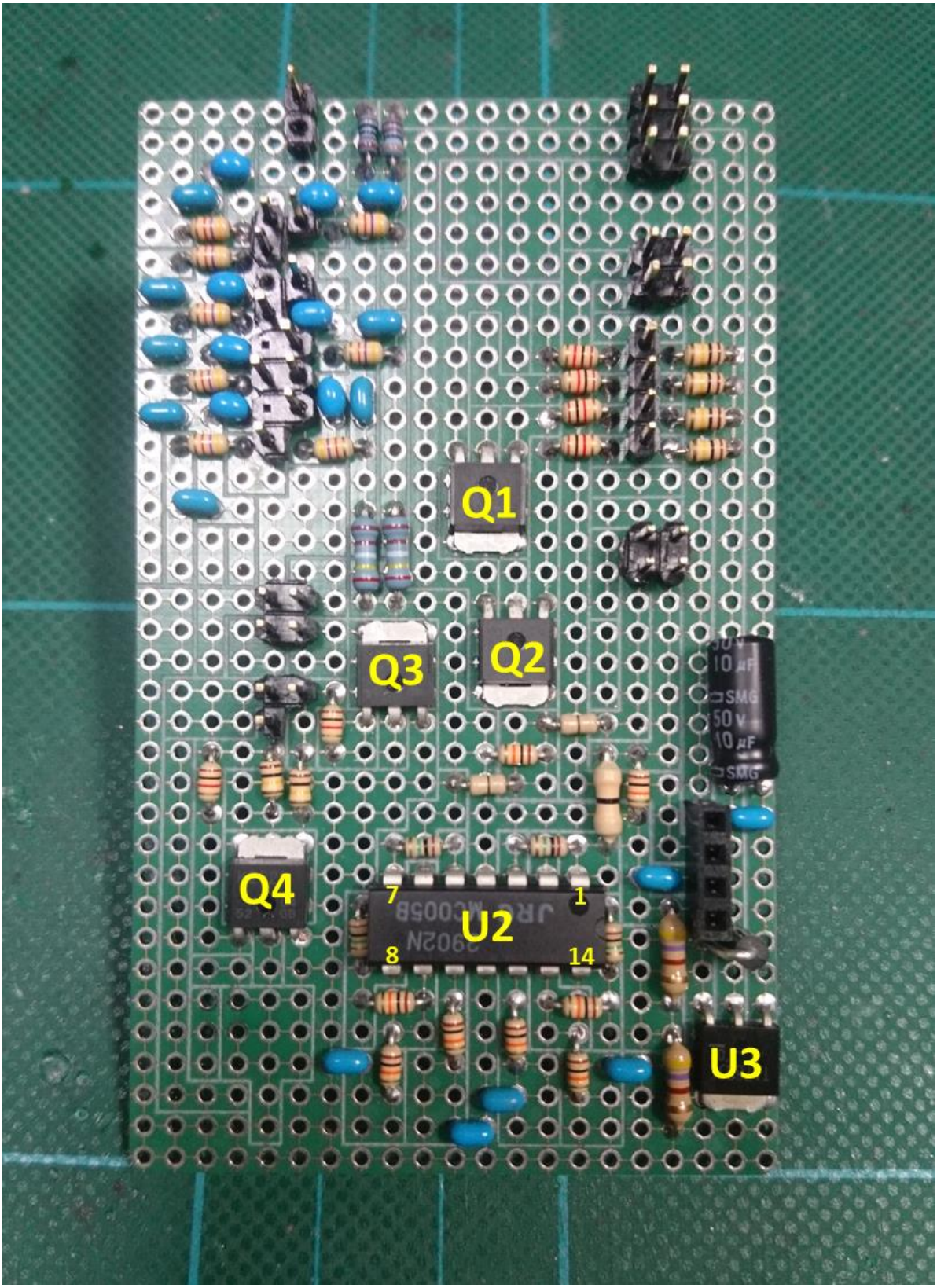
ダイオード

(b) ダイオード D1 には向きがあります。必ずカソード側を 5V 電源側、アノードをグランド側に接続して下さい。

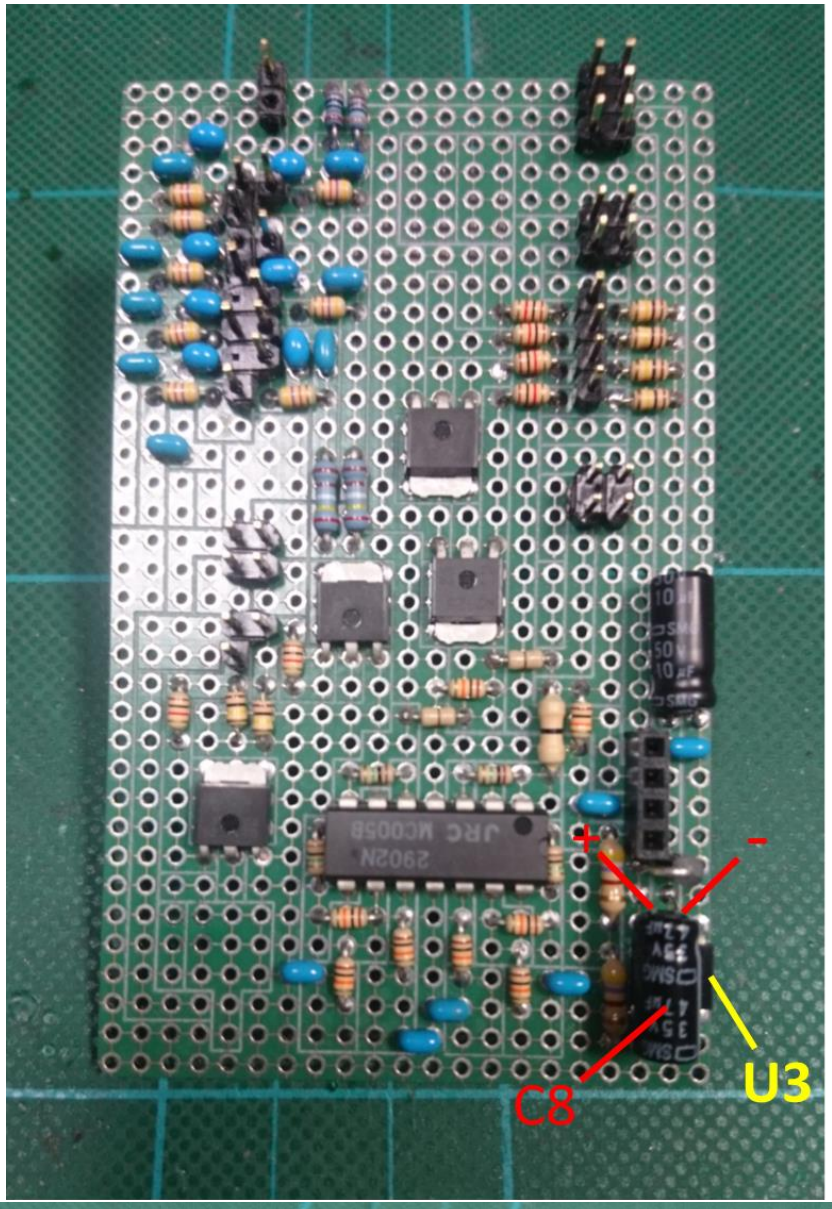


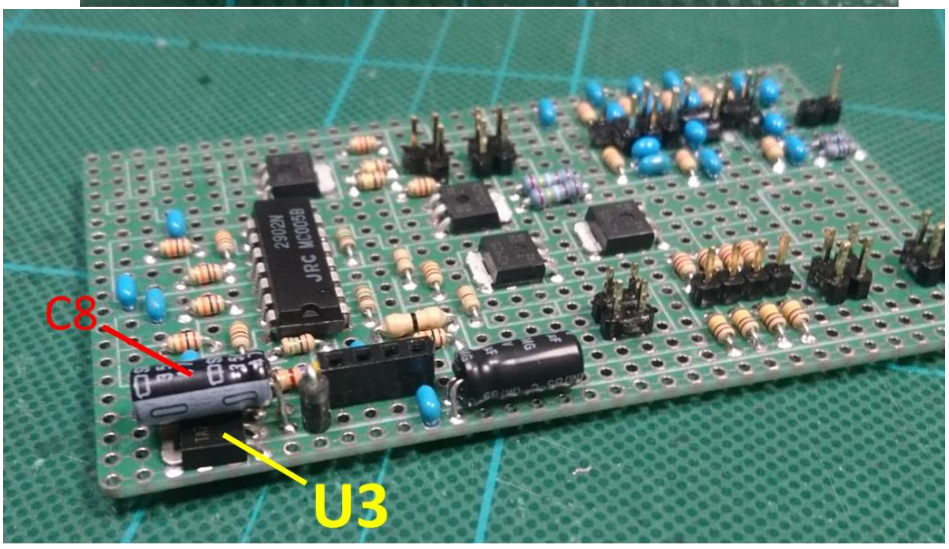
5. MOS-FET（Q1、Q2、Q3、Q4）、3 端子レギュレーター（U3）、オペアンプ（U2）

オペアンプ（U2）は下図の様に、1 番ピンが右上になるよう配置して下さい。



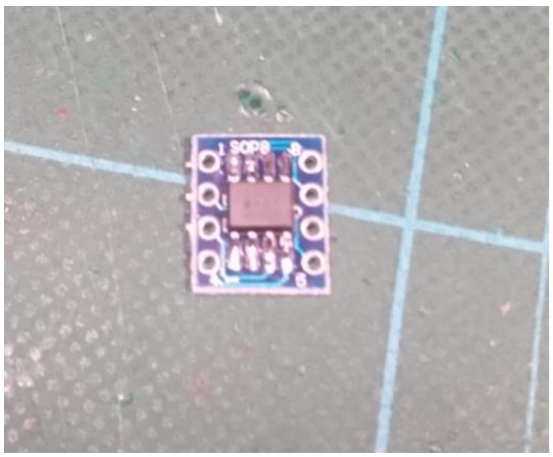
6. 電解コンデンサ（C8）



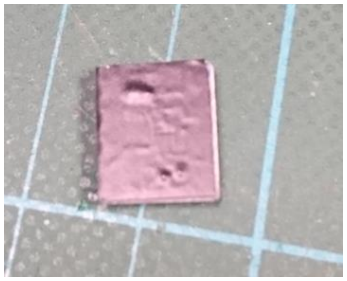


7. デュアル MOS-FET

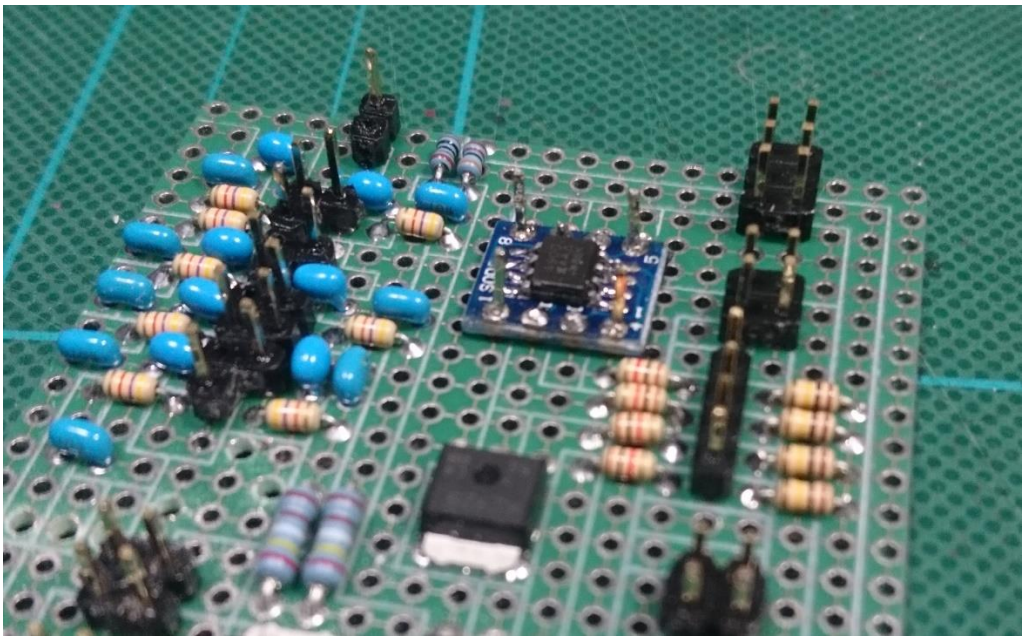
(a) デュアル MOS-FET（U4）をピンピッチ変換基板に半田付けします。



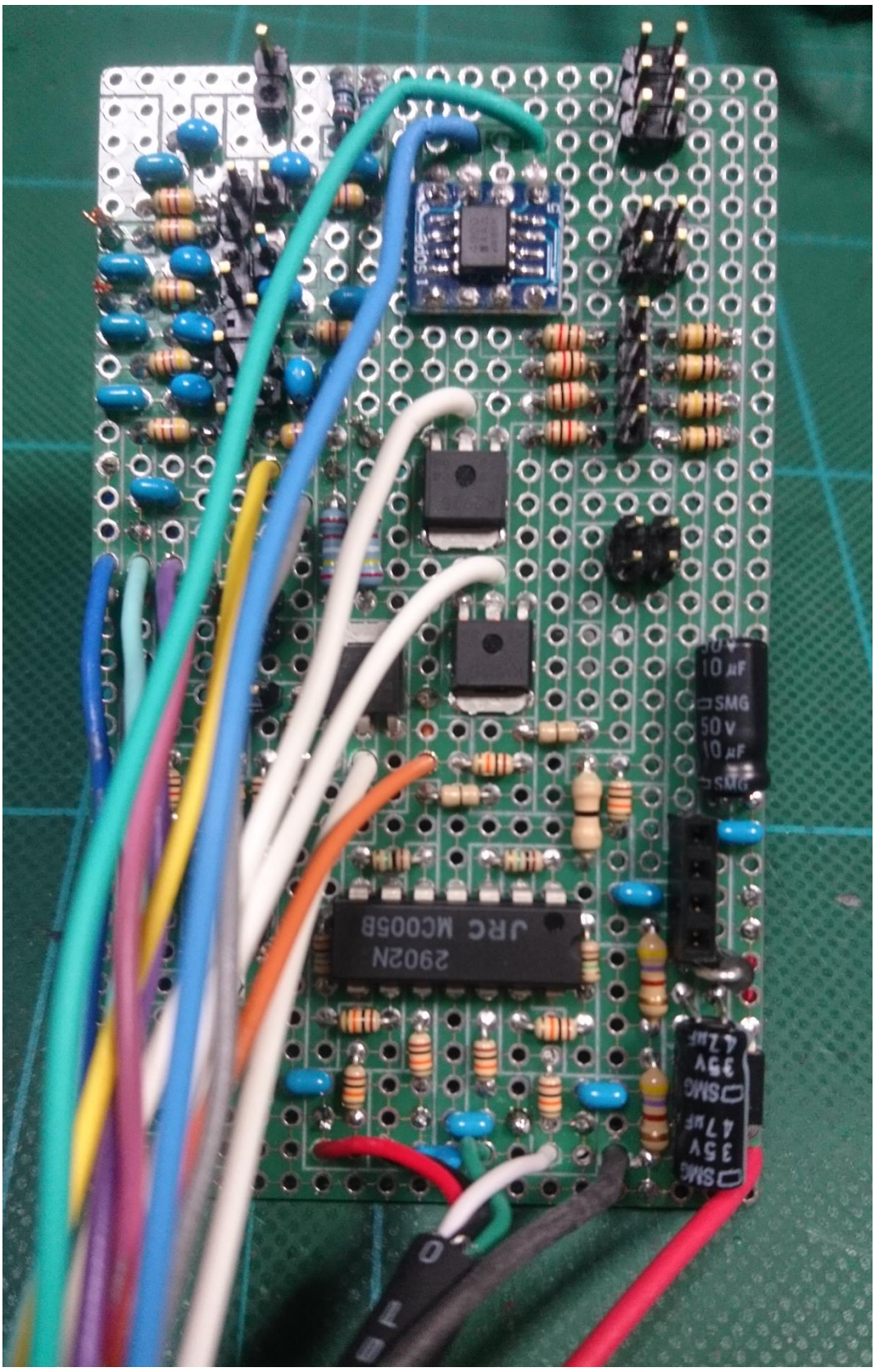
(b) ピンピッチ変換基板は裏側にもパッドがある為、そのまま PJSC ボードに装着するとデュアル MOS-FET の端子をショートしてしまいます。これを避ける為、ピンピッチ変換基板の裏側に絶縁テープを貼ります。

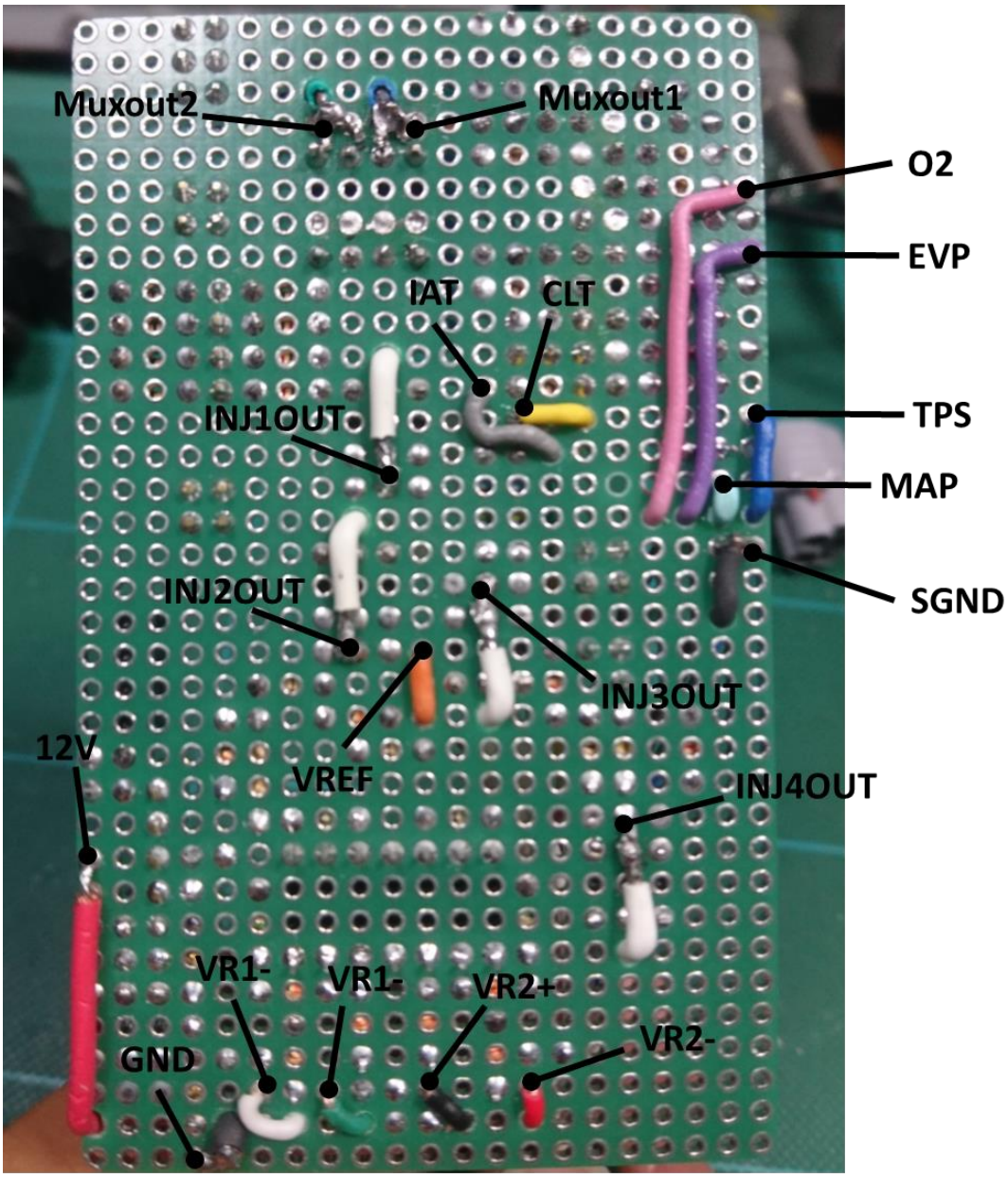


(c) ピンピッチ変換基板を所定の位置に設置し、ピンピッチ変換基板のスルーホールと PJSC ボードのスルーホールを貫通する様にリード線を刺し込み、スルーホールに半田を流し込みます。余分なリード線を切断します。



各入出力ピンにリード線を半田付けします。




Muxout2
Muxout1
O2
EVP
IAT
CLT
INJ1OUT
TPS
MAP
INJ2OUT
SGND
INJ3OUT
12V
VREF
INJ4OUT
VR1-
VR1-
VR2+
VR2-
GND

2.3　センサーキャリブレーション

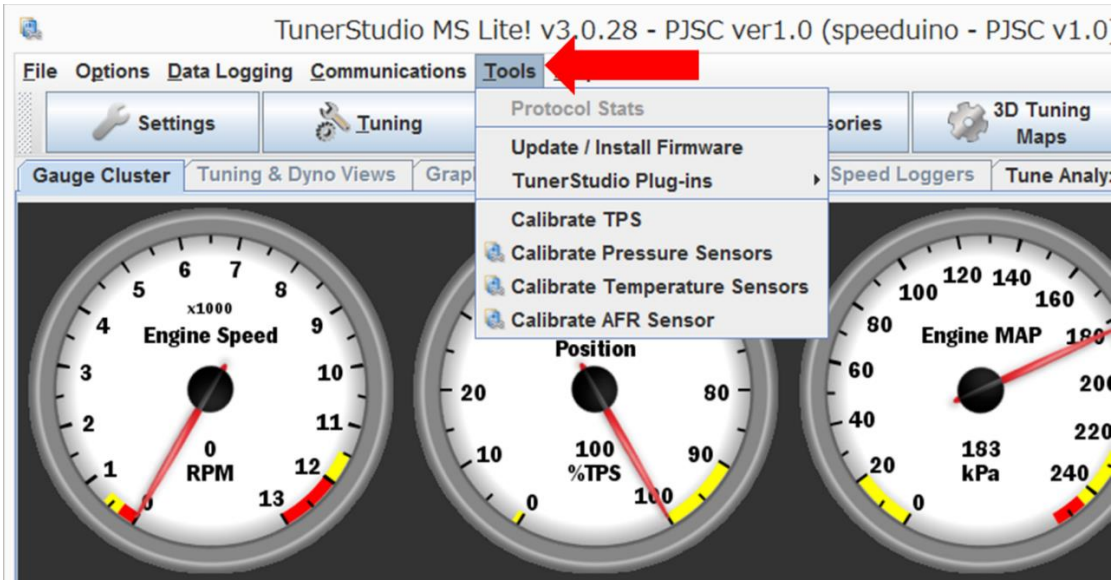
2.3.1　センサーキャリブレーション

PJSC でのチューニングを正しく行う為に、センサーのキャリブレーションは必須の作業です。配線図を参照してアッセンブリが完了した PJSC と車体配線を結線し車体に組込んだら、組み込んだ車両の為のプロジェクトを作成します（『1.6 プロジェクト作成』を参照）。プロジェクトを作成したら、最初にセンサーキャリブレーションを実施して下さい。

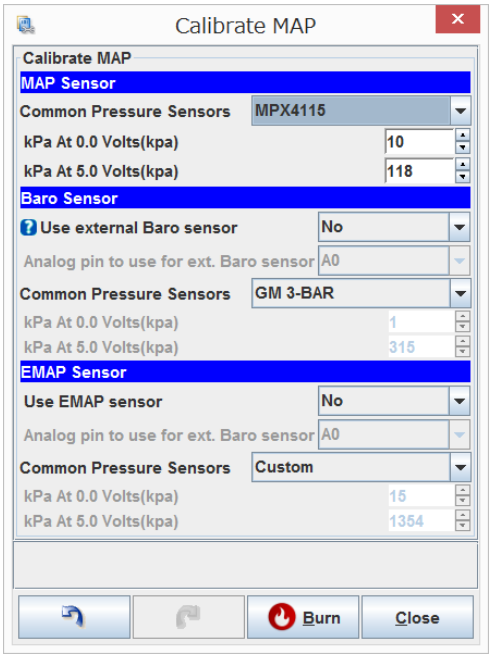
キャリブレーションを行うには Tuner Studio でセンサーの型番、或いは信号特性を入力するので、それらの情報が必要です。使用するセンサーの情報は各メーカーより入手するか、予め測定器を用いて計測しておいて下さい。

MAP センサー

メインメニューの Tools を開きます。



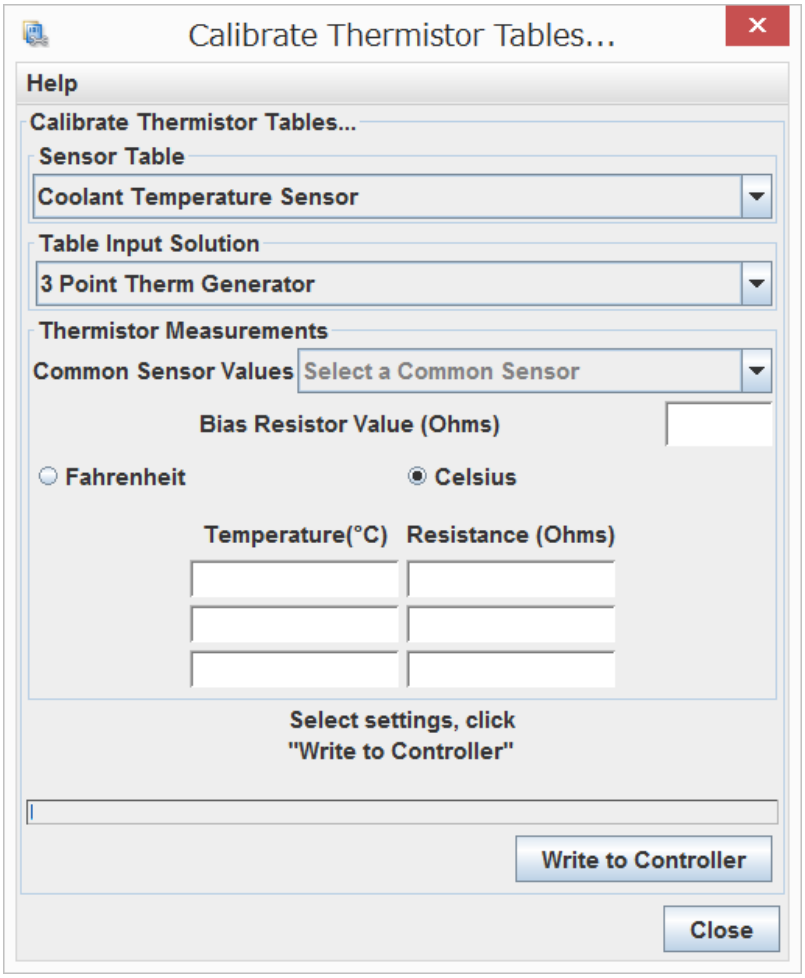
"Calibrate Pressure Sensors"を選択すると、以下の様なウィンドウが表示されます。



プルダウンメニューから使用する MAP センサーの型番を選択します。その後"Burn"ボタンをクリックすると、センサー情報が PJSC に書き込まれます。

CLT、IAT センサー

メインメニューの Tools から、"Calibrate Thermistor Table"を選択します。

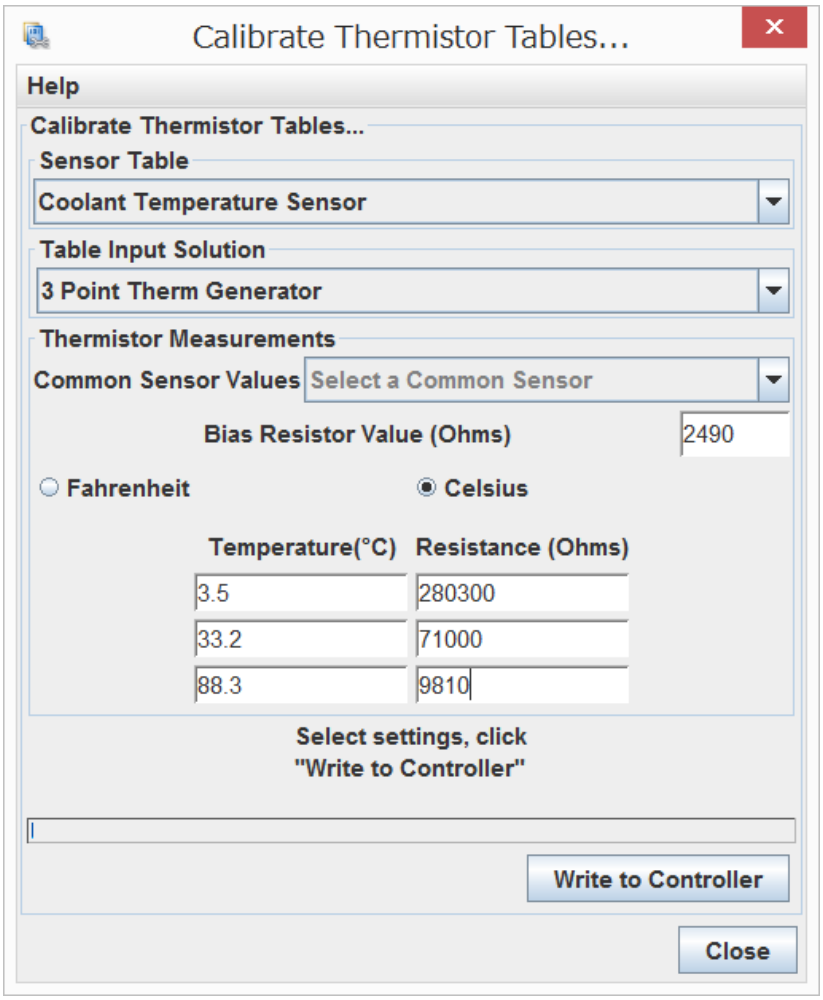


Sensor Table のプルダウンメニューで"Coolant Temperature Sensor"を選択します。

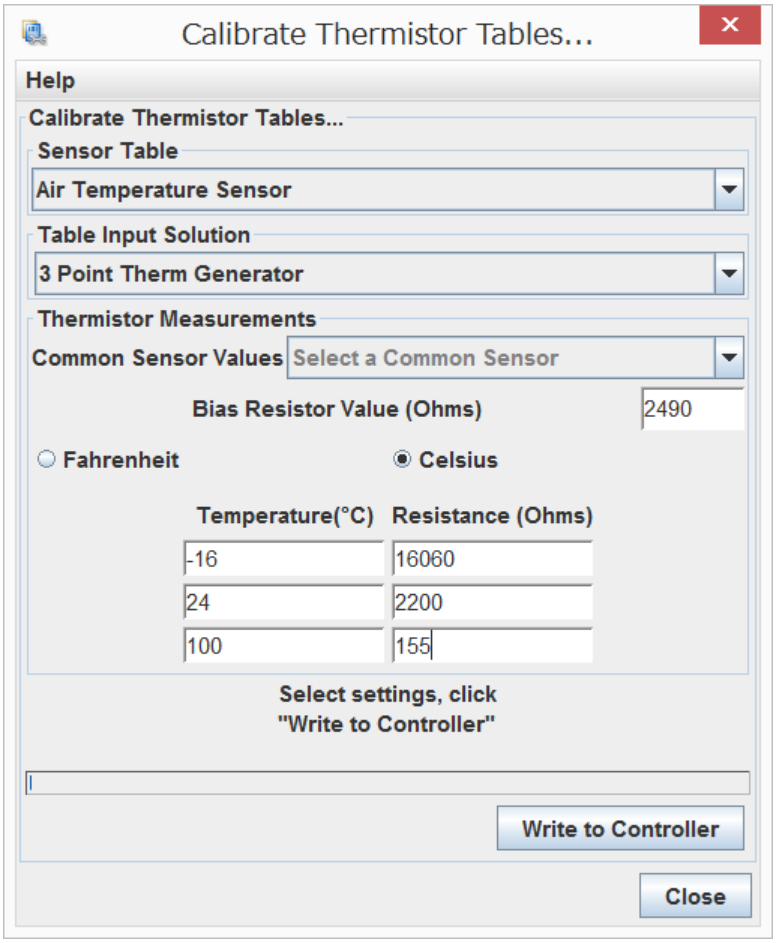
"Common Sensor Values"のプルダウンメニューから、使用する CLT（水温センサー）の型番を選択します。

"Bias Register Value"欄に 2490（Ω）を記入します。この値は PJSC 基板上のバイアス抵抗値なので、異なる抵抗値の抵抗に変更しない限り、値は変えないで下さい。

"Write to Controller"ボタンをクリックすると、CLT キャリブレーションテーブルが PJSC に書き込まれます。書き込みが完了したら"Close"をクリックしてウィンドウを閉じて下さい。



再度 Tools メニューから"Calibrate Thermistor Table"を選択して Calibrate Thermistor Tables ウィンドウを開きます。"Sensor Table"プルダウンメニューから"Air Temperature Sensoer"を選択します。



"Common Sensor Values"プルダウンメニューから、使用する IAT（吸気温度）センサーの型番を選択します。

"Bias Register Value"欄に 2490（Ω）を記入します。

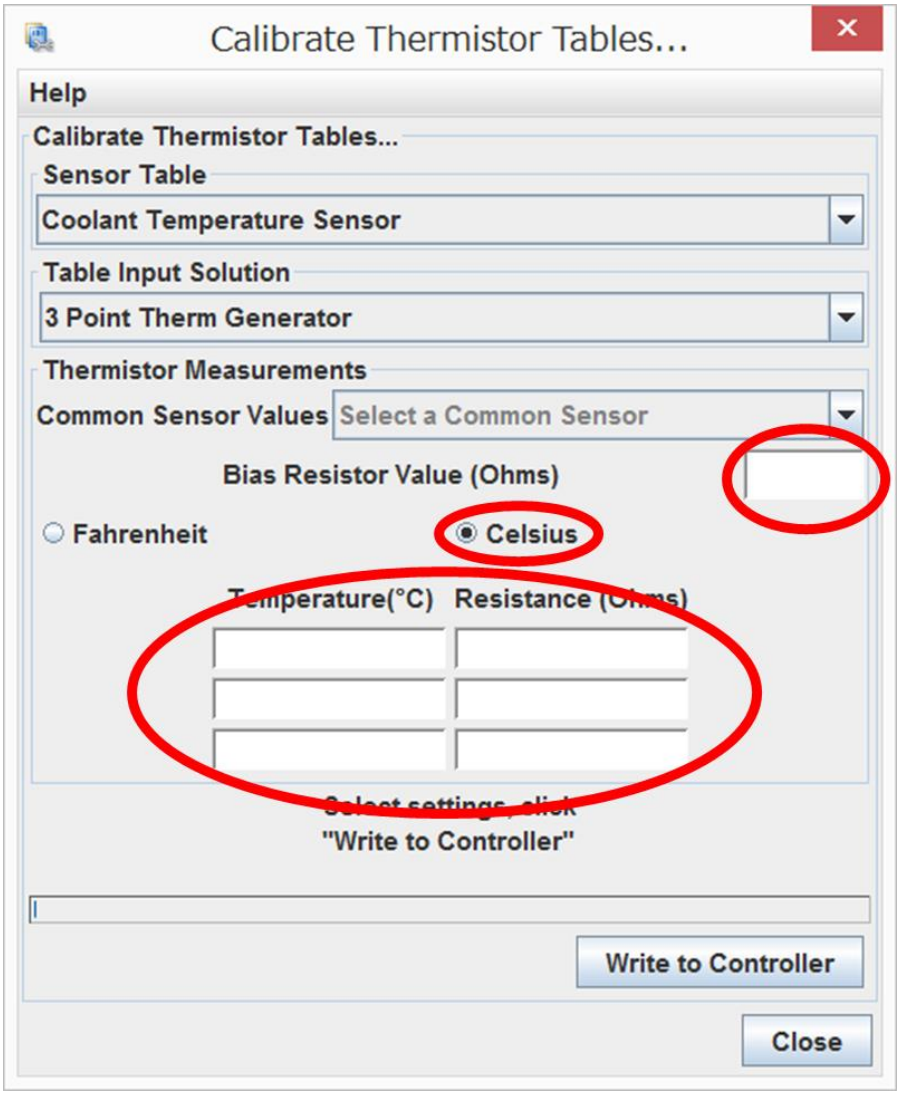
"Write to Controller"ボタンをクリックすると、IAT キャリブレーションテーブルが PJSC に書き込まれます。

リストに無いセンサーを使用する場合

"Common Sensor Values"のプルダウンメニューに使用するセンサーが無い場合、センサーの特性値を"Thermister MEasurements"フィールドに入力する事で使用する事が可能です。異なる 3 点の温度に対するセンサーの抵抗値を入力して下さい。

温度センサーは通常、センサー周囲の温度によって抵抗値が変わるサーミスターを使用しています。センサー温度が出来るだけ使用する温度範囲の最高温度、最低温度と中間の温度になる環境中に置き、その時の温度と抵抗値を測定して下さい。この時センサーを手で持って測定を行うと、体温がセンサーに伝わって正確な抵抗値が測れません。温度センサーに使用するサーミスターは温度変化に対して感度が高く、応答が早い為です。センサーに体温が伝わるのを避ける為に、予め配線をして配線の両端で抵抗値を測定すると良いでしょう。

3点の温度と抵抗値を入力して"Write to Controller"ボタンをクリックすると、Tuner Studioが入力された数値からキャリブレーションテーブルを計算しPJSCに書き込みます。



O2センサーキャリブレーション

メインメニューのToolsから"Calibrate AFR Table"を選択します。

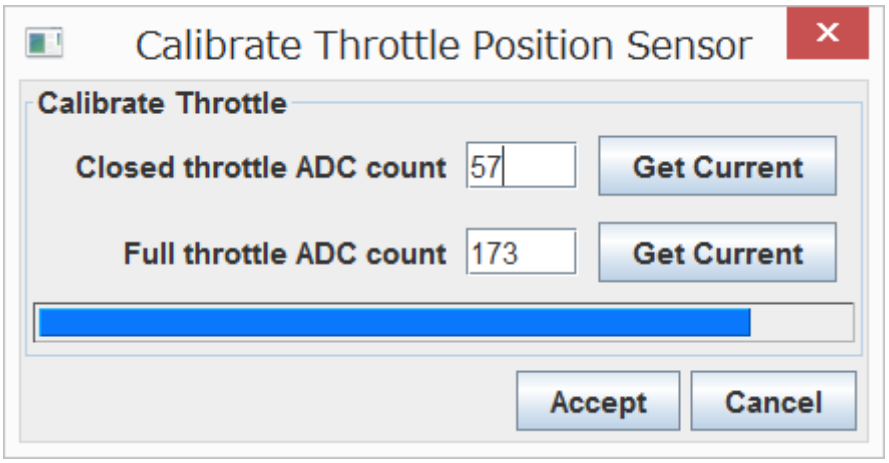
"Common Sensor Values"プルダウンメニューから、使用するO2センサーの型番を選択します。もし使用するセンサー型番がリストに無い場合"Custome Linear WB"を選択し、センサーのマニュアルに記載されているセンサー出力特性から任意の2点での電圧とAFR値を読み取って入力して下さい。

"Write to Controller"ボタンをクリックすると、O2センサーキャリブレーションテーブルがPJSCに書き込まれます。

TPS（スロットルポジションセンサー）

TPS 信号は PJSC で燃調を制御する為に重要な信号です。燃調方式にα－N を選択した場合、必ず TPS のキャリブレーションを行って下さい。

また TPS に印加するリファレンス電圧は 5V でなければなりません。PJSC を他の ECU と併用して TPS のリファレンス電圧を PJSC 以外から供給する場合、必ずこの電圧を確認して下さい。5V より高い電圧が供給されている場合、PJSC が破損する恐れがあります。その場合リファレンス電圧は PJSC から供給するか、TPS 信号が 5V 以下になるようにレベルシフト回路を介してから PJSC へ入力して下さい。



TPS にリファレンス電圧を供給し、信号を PJSC に入力した状態でメインメニューの Tools から"Calibrate TPS"を選択します。

スロットルを全閉状態にして、"Closed Throttle ADC count"欄横の"Get Current"ボタンをクリックします。"Closed Throttle ADC count"欄に TPS 信号を Arduino が ADC 変換した値が入力されます。

次にスロットルを全開状態で固定し、"Full Throttle ADC count"欄横の"Get Current"ボタンをクリックします。"Full Throttle ADC count"欄にスロットル全開時の ADC 値が入力されます。

"Accept"ボタンをクリックすると、TPS キャリブレーション値が PJSC に書き込まれます。

第３章　デコーダー

3.1　　ミッシングトゥース（欠歯）

3.1.1　概要

ミッシングトゥースクランクトリガーは、多くのメーカーの OEM が標準的に採用しているクランク角検出方式です。またアフターマーケットの ECU でも採用例が多いポピュラーな方式です。

これは一定数の等間隔のトリガー歯を有するクランクホイールと、1 つ以上の「欠歯」とで構成されます。 一般的なトリガー歯数と欠歯数の組合せは 60-2、36-1、24-1、12-1、4-1 です。

ミッシングトゥースではこの様に二つの数字を組み合わせ、歯数と欠歯の数を表現します。最初の数字は欠歯を含めた歯の総数で、ハイフンの後の数字が欠歯数を表します。

例えば「36-1」では実際の歯数は 35 で 10 度等間隔に並び、欠歯が一つという事を表しています（欠歯の前後のトリガー歯の間隔は 20 度）。「36-2」では 10 度間隔のトリガー歯が 34 と欠歯が 2 で、欠歯の前後のトリガー歯の間隔は 30 度となります。

註）第 3 の数字（例えば、36-1-1）がある場合欠歯は連続しておらず、欠歯と欠歯の間に一つ以上のトリガー歯がある方式となります。しかし PJSC ではこの様な欠歯が複数の方式はサポートしていません。

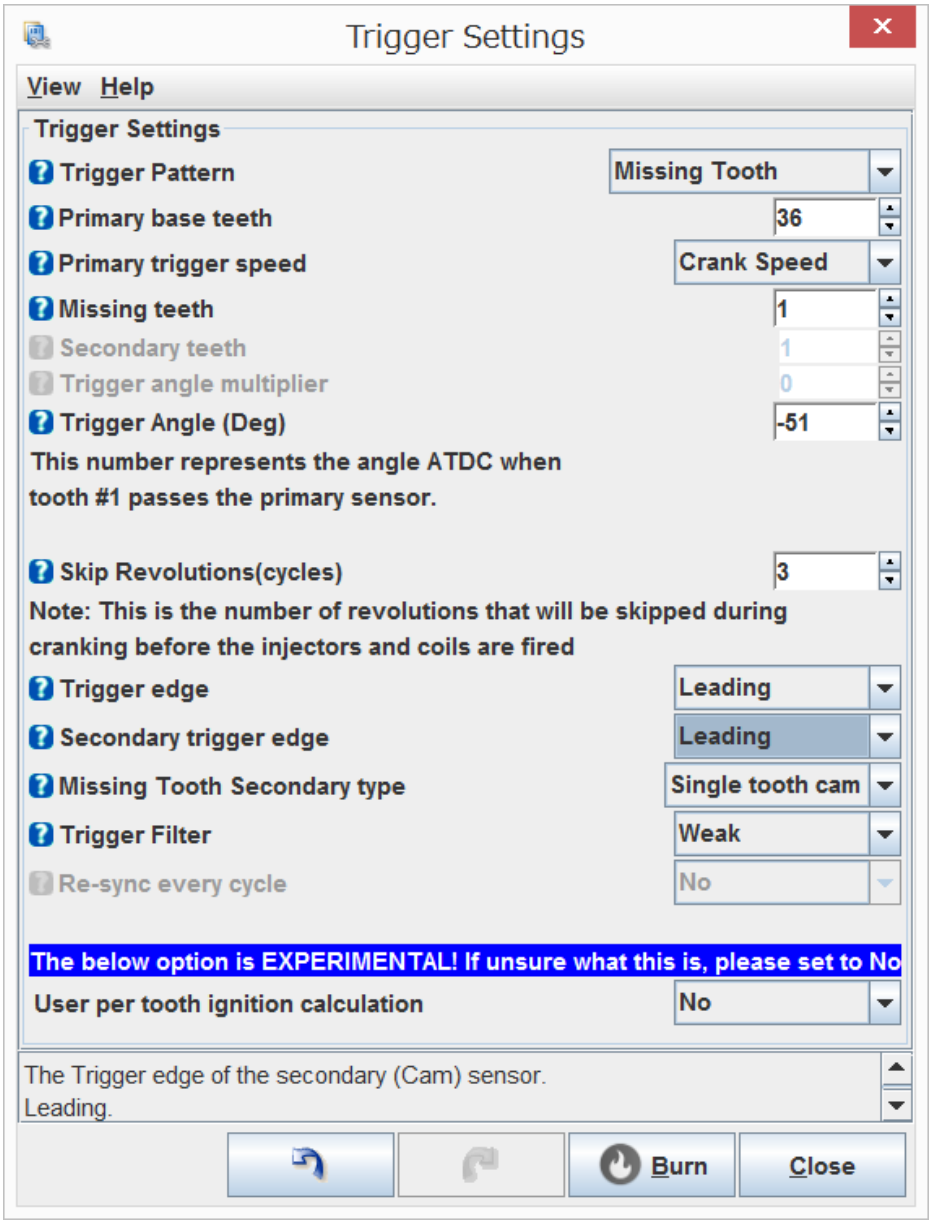
また「36/1」という表記の方式もあります。この場合スラッシュの後の数字はカムの歯を表しています。これは欠歯ではないので混同しないよう注意して下さい。

3.1.2　アプリケーション

ミッシングトゥースクランクホイールは事実上すべてのエンジンで使用可能で、アフターマーケット ECU では最もポピュラーな方式です。

トリガー歯数が多くなるほどクランク角検出の分解能が高くなり、CPU に高い負荷を掛けずに制御タイミングの精度を上げる事が出来ます。

3.1.3　Tuner Studio 設定



・**Primary base teeth**：プライマリーホイールの総歯数を入力します。これには欠歯も含みます。例えば 36-1 の場合、実際のトリガー歯は 35 本しかありませんが、このフィールドには 36 を入力します。

・**Missing Teeth**：これは欠歯の数－トリガー歯とトリガー歯のギャップ間の歯数を入力します。全ての欠歯は連続して一カ所に配置されていなければなりません。つまりギャップはホイール上に一カ所だけという事になります。

・**Trigger Angle**：欠歯－ギャップに続く最初のトリガー歯（インデックストリガー）が検出されるクランク角度を ATDC（After Top Dead Center）で入力します。例えばインデックストリガーが BTDC51 度で検出される場合、「-51」を入力します。

タイミング設定

シーケンシャル制御

ミッシングトゥースデコーダはカムセンサーも追加する事で、シーケンシャル制御を行う事が出来ます。燃料噴射タイミングにシーケンシャルモードが選択されている場合、PJSC はカム信号が入力される事を前提にタイミングを決めます。よってカムセンサーが無ければ正しく同期出来ません。

カムセンサー信号は 1 サイクル当たり 1 のパルスの信号でなければなりません。カム歯が極短い（狭い）物か半月型の場合、電気的に 1 サイクル当たり 1 つの立ち上がり（または立ち下がり）エッジしか出力されない事があり得ます。

3.2 カムミッシングトゥース

3.2.1 概要

カムミッシングトゥースデコーダーは、シングルホイールでありながらデュアルホイールと同様の機能を実現出来ます。カムまたはディストリビューターにミッシングトゥースホイールを装着し、位相を同期させてシーケンシャル制御を可能にします。

カムミッシングトゥースの動作はミッシングトゥース（クランク装着）とデュアルホイールの両方に共通しています。 最初にこれらのセクションを読んで、理解することをお勧めします。本セクションでは、ミッシングトゥース（クランク装着）とデュアルホイールとの違いのみ解説します。

このデコーダはクランク装着ミッシングトゥースと同様に、シングルカムホイールで構成されています。トリガー歯の数は 720° を均等に割り切れる数である必要があります。 カムホイールはクランクホイールの半分の角速度で回転し、センサーはクランク 1 回転（360 度）でカムホイールの半分のトリガー歯を検出し、次の 1 回転で残り半分のトリガー歯を検知します。

ミッシングトゥースはクランク 2 回転につき 1 回検出され、デュアルホイールデコーダーのカム信号と同様に位相を同期させる為に使用されます。

3.2.2 アプリケーション

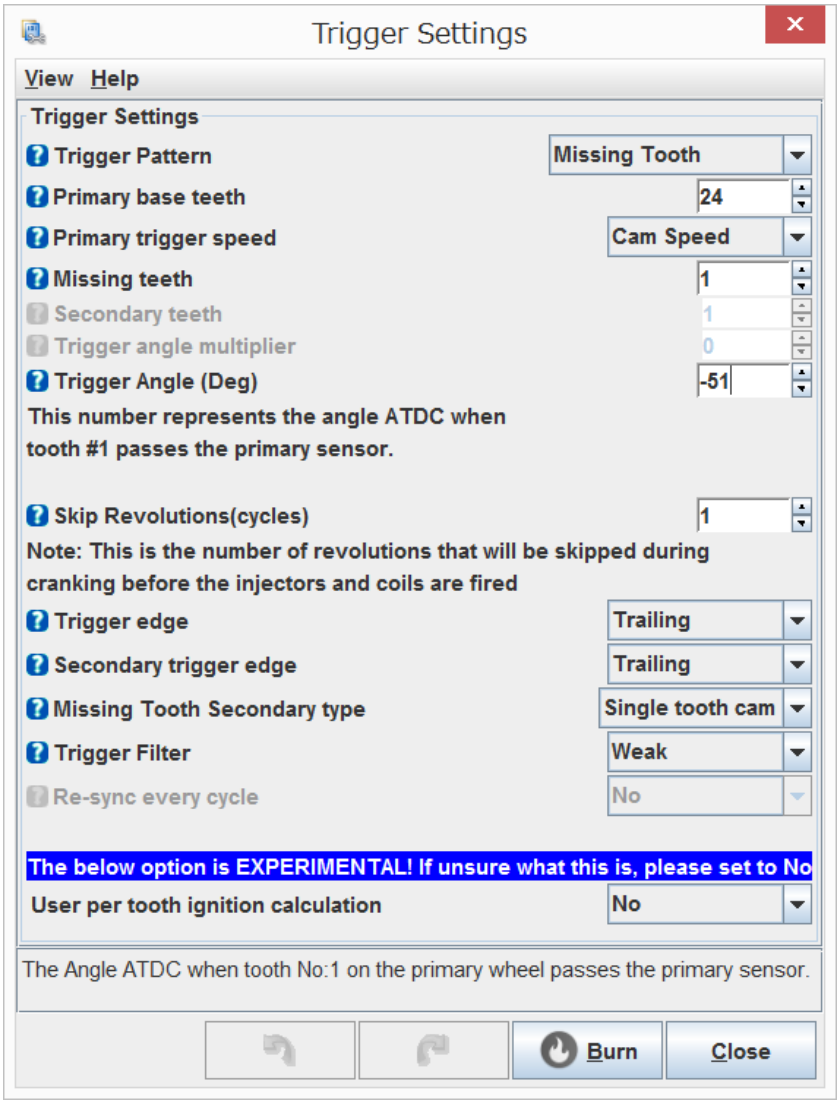
カムホイールには、最低でもシリンダー数と同数のトリガー歯がなければなりません（欠歯を除く）。一般的にはシリンダ数の 2 倍以上のトリガー歯数を必要とします。クランク角検出の分解能が高くなるよう、出来るだけ多くのトリガー歯を設置する事を推奨します。スペースの問題で一般的に、クランクホイールに

比べてカムに設置するホイールは直径が小さくなります。この為、クランクホール方式と比べて、センサーはより小さいトリガー歯または近接したトリガー歯を確実に読み取れる性能が必要になります。

カムホイールでは、クランク１回転でも半分のトリガー歯しか読み取られません。またクランクの角速度は常に一定ではなく１回転する間にも角速度が変動します。これらの要因の為、カムホイール方式はクランクホイール方式に比べて角度検出、タイミング検出の精度が劣ります。

１回転中の角速度変動率はエンジンの仕様によって異なる為、どの程度の誤差が生じるかは一概には言えません。

3.2.3 Tuner Studio 設定



・**Primary base teeth**：プライマリーホイールの総歯数を入力します。これには欠歯も含みます。例えば 36-1 の場合、実際のトリガー歯は 35 本しかありませんが、このフィールドには 36 を入力します。

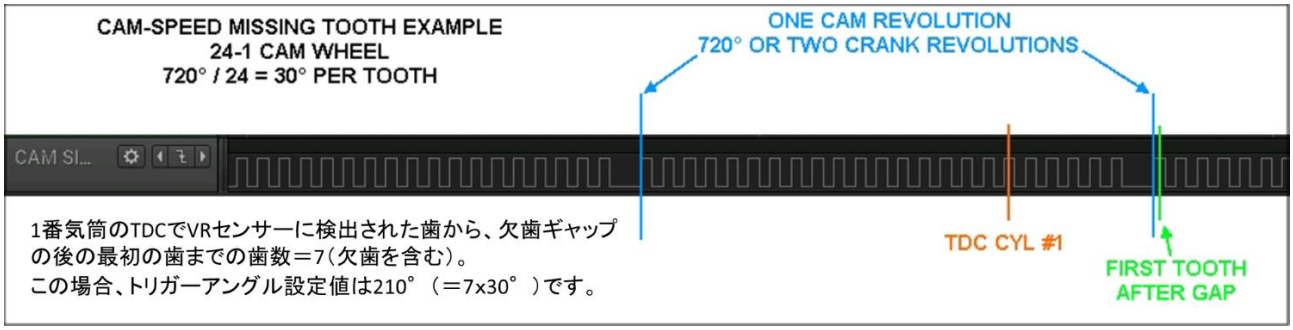
・**Primary trigger speed**：Cam Speed を選択して下さい。

・**Missing Teeth**：これは欠歯－トリガー歯のとトリガー歯のギャップ間の歯数を入力します。全ての欠歯は連続して一カ所に配置されていなければなりません。つまりギャップはホイール上に一カ所しだけという事になります。

・**Trigger Angle**：欠歯－ギャップに続く最初のトリガー歯（インデックストリガー）が検出されるクランク角度を ATDC（After Top Dead Center）で入力します。例えばインデックストリガーが BTDC51 度で検出される場合、「-51」を入力します。

タイミング設定

3.2.4　トリガーパターン



3.3　デュアルホイール

3.3.1　概要

このデコーダは 2 つのホイールがある場合に使用されます。 プライマリホイールの回転速度はクランクの回転速度と同じでなければならず、またミッシングトゥースではない必要があります。セカンダリーホイールはクランクまたはカム何れに装着しても良く、歯数は 1 つだけでなければなりません。センサーがセカンダリーホイールの歯を検出して出力するパルスはカム位相と同期する為の物で、トリガーと区別する為にシンクパルス（sync pulse）と呼びます。

この方式では、シンクパルスが検出された後に最初に検出されるプライマリーホイール上のトリガー歯がインデックストリガーと定義されます。 プライマリホイールがクランクではなくカムに装着されている場合、Tuner Studio のトリガー設定でプライマリーホイールの歯数を 2 つに分割してクランク速度を取得します。例えば歯数 24 のプライマリーホイールがカムに装着されている場合、Primary base teeth に 12 を入力します。

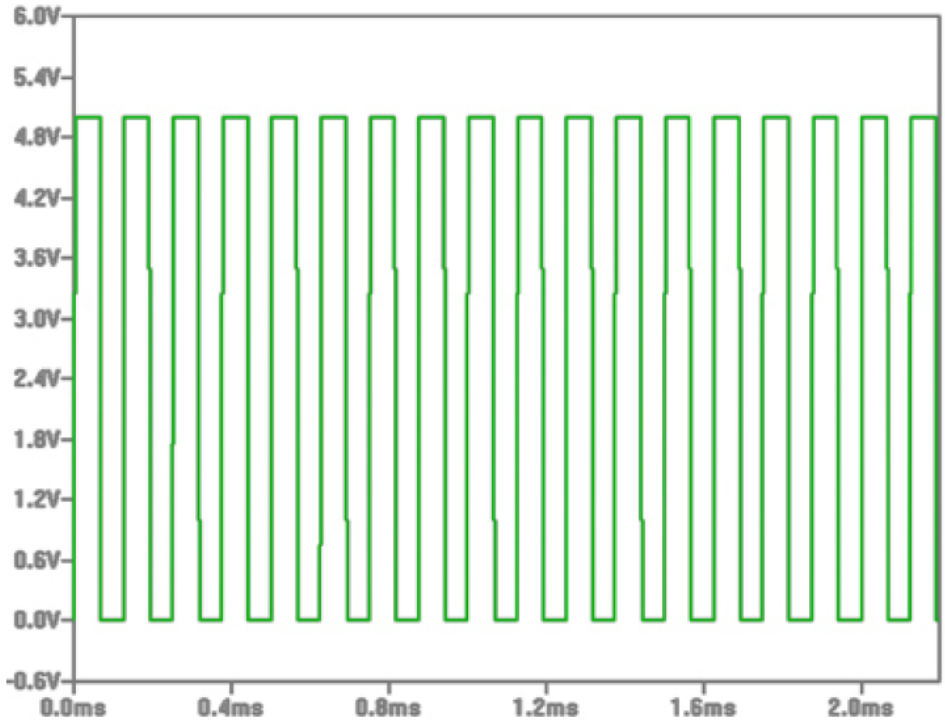
3.4　ベーシックディストリビューター

3.4.1　概要

これにはシリンダ毎の行程を同期する信号がありません。 ミッシングトゥース或いはシンクパルスが無い場合、PJSC はクランク角、サイクル位相、またはシリンダ割り当てを計算出来ません。点火信号を適切なシリンダーに送るために、ディストリビューターを必要とします。

この信号は機械式接点、機械式ディストリビューターが使用されていた車両の様に、クランクシャフト 1 回転につき 1 パルスの非常にシンプルなもので構いません。

3.4.2　トリガー信号



3.5　GM7X

3.5.1　概要

このデコーダーは GM、で多く採用されている方式の一つです。この方式では、6 つの等間隔のトリガー歯と 1 つの不等間隔のトリガー歯を持つトリガーホイールを使用しています。不等間隔のトリガーは合計 7 つのトリガーの内の、3 番目のトリガーとして識別されます。

3.6 4G63

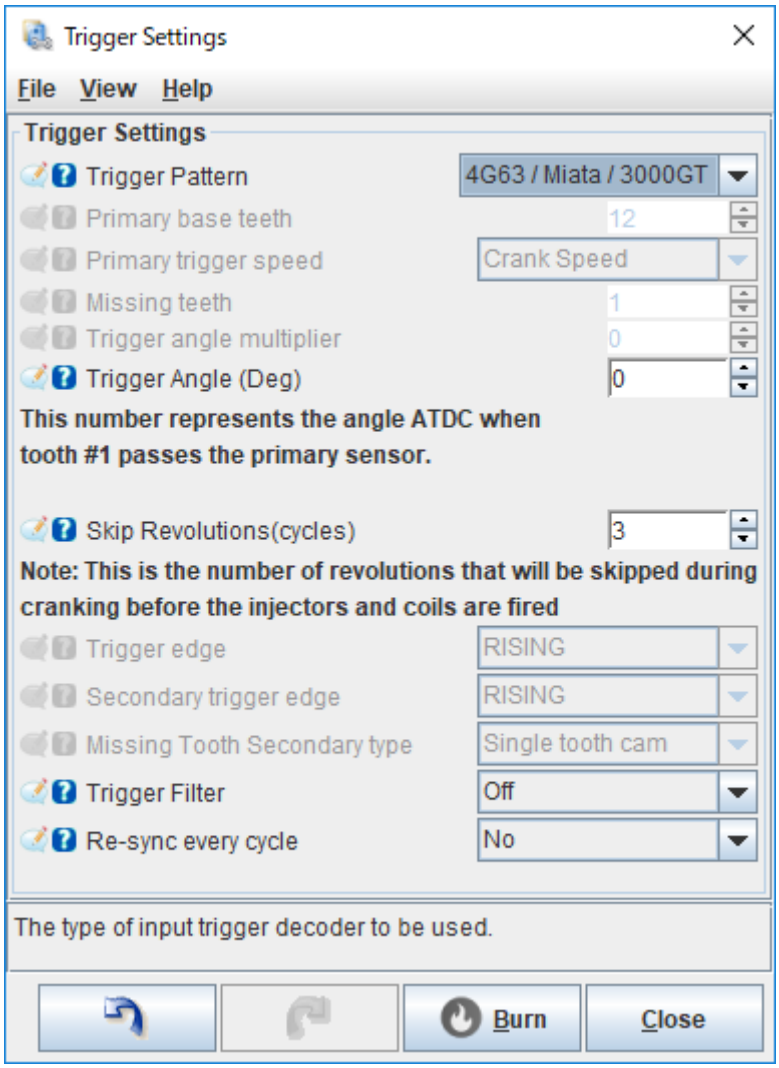
3.6.1 概要

4G63 トリガーは三菱やマツダの 4 気筒エンジンで使用されています。この方式はクランクとカムの二つトリガーを、ホールセンサ或いは光学センサによって検出しています。

3.6.2 適用車種

- 三菱ランサー
- NA/NB Miata / MX-5

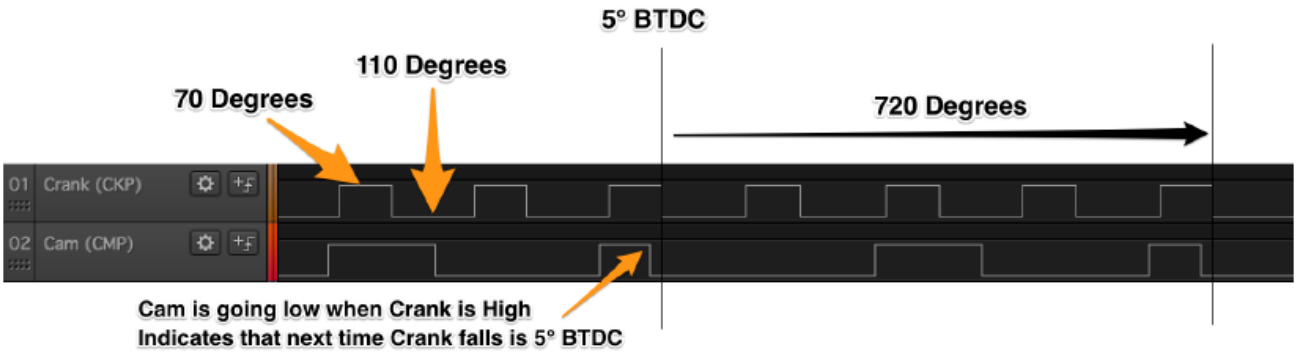
3.6.3 Tuner Studio 設定



3.6.4　タイミング補正

通常はトリガーアングルの変更は不要（Trigger Angle = 0 度）ですが、純正のトリガーホイールにはトリガーとトリガーの間隔に若干のバラつきがあります。バラつきが大きい場合、Trigger Angle に値を入力して、タイミング補正を補正する必要があります。

3.6.5　トリガーパターン

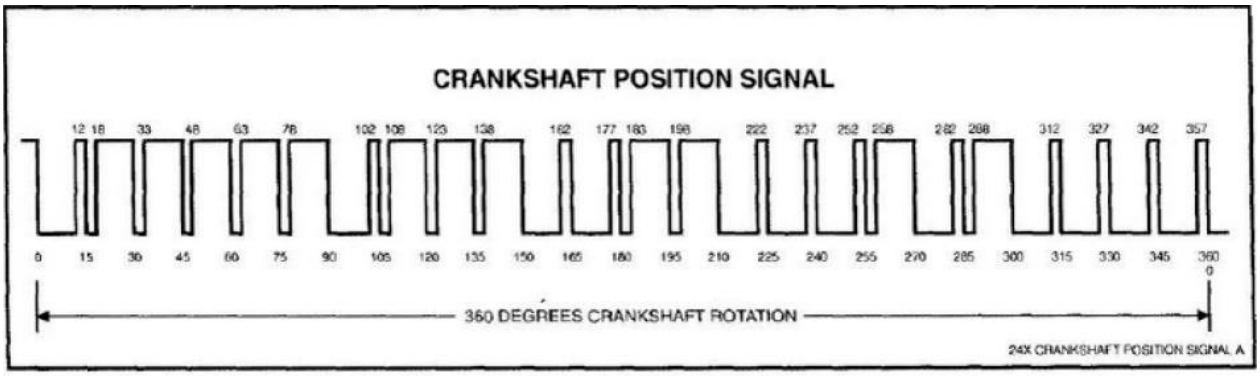


3.7　GM24X

3.7.1　概要

この方式のトリガーホイールは、は広いトリガー歯と狭いトリガー歯 12 本ずつ、合計 24 本のトリガー歯を持っています。狭いトリガー歯の幅は 3 度、広いトリガー歯の幅は 12 度あります。全ての立ち下がりエッジの間隔は 15 度となります。 このデコーダは立ち下がりエッジでトリガーを検出する為、クランク角を識別するためにカム信号が必要です。

3.7.2　トリガー信号



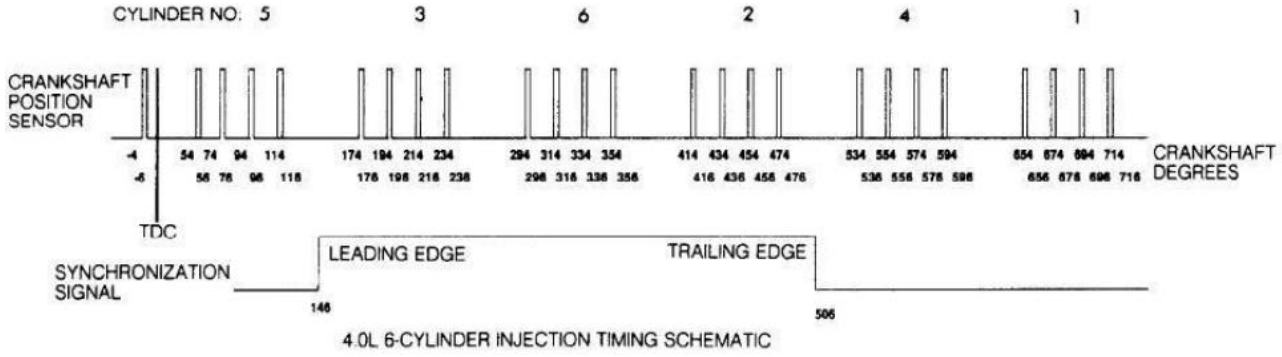
Crankshaft Position Sensor Signal

3.8 Jeep2000

3.8.1 概要

クランクホイールとカムホイールから信号を得る方式です。クランクホイールセンサーは 120 度毎に 4 つのパルスを出力します。4 つのパルスの間隔は 20 度です。カムホイールはクランク角 360 度または 720 度で 1 回回転し、パルスは 180 度または 360 度の間 Hi レベルになります。

3.8.2 トリガー信号



4.0L 6-CYLINDER INJECTION TIMING SCHEMATIC

3.9 アウディ 135

アウディ 135 デコーダーはアウディ V 6、I 5、I 4 T など、1 回転あたり 135 パルスのトリガー信号を使用するエンジンの為のデコーダーです。

3.10 ホンダ D17

3.10.1 概要

ホンダ D17 デコーダーは、12 + 1 クランクホイールを使用するホンダ系エンジンに使用されているデコーダーです。4 + 1 カム信号は現在 PJSC ではサポートされていないため、セミシーケンシャルおよびグループ噴射のみ使用可能です。

3.10.2 アプリケーション

・TBA

3.10.3　Tuner Studio 設定

3.10.4　タイミング調整

ほとんどの場合トリガーアングルの変更は必要ありませんが、OEM バージョンのトリガー間隔には若干のばらつきがあるため調整が必要になる場合があります。エンジンが始動したら点火タイミングを固定にして角度を設定し、タイミングライトで点火タイミングを確認して下さい。点火タイミングがずれている場合、トリガーアングルを調整して下さい。

3.10.5　トリガーパターン

クランクホイールは等間隔に配置された 12 個のトリガー歯と、インデックストリガーの位置情報を与える 13 番目のトリガー歯で構成されています。この 13 番目のトリガー歯の後に来る最初のトリガー歯が #1（インデックストリガー）として識別されます。

3.11　Miata 99

3.11.1　概要

MY99 以降、Miatas は新しいトリガーパターンに移行しました。これは 4g63 で使用されていたものと似ていますが、よりノイズ耐性が向上しています。また可変カムタイミングが採用されたエンジンでは、クランク信号に対してカム信号の位相も可変となります。この方式は可変カムの位相変化にかかわらず、同期信号（Sync）を識別出来るので可変カムタイミングが採用されているエンジンに適しています。

トリガーは、クランクシャフトに配置された 4 歯ホイールとカムに配置された 3 歯ホイールで構成されています。二つのホイールの歯は不均等な間隔で配置されています。

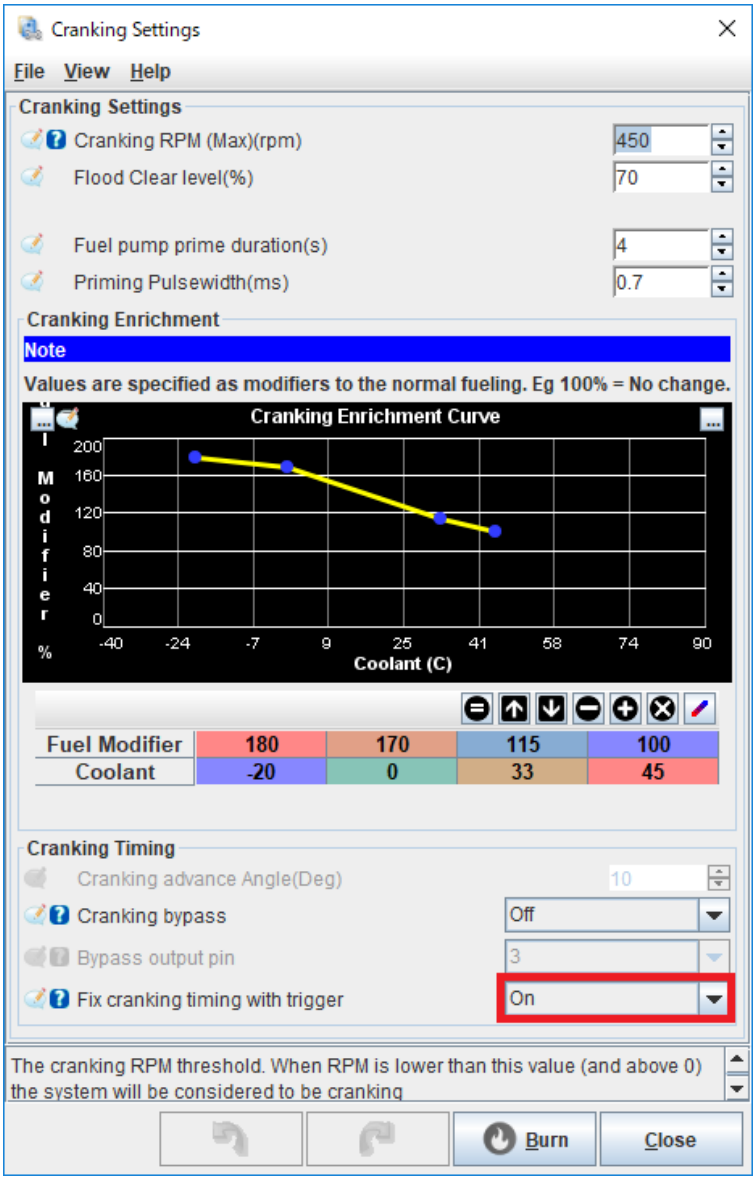
3.11.2　アプリケーション

1999 年から 2005 年までの NB Miatas に適用されます。

3.11.3　Tuner Studio 設定

このデコーダーではトリガーアングルを変更する必要はありません。また殆どの場合、トリガフィルタリングはオフまたは弱に設定して下さい。

Starting/Idle メニューの Cranking Settings でクランキング設定ダイアログを開き、Fix cranking timing with trigger オプションを On に設定します。

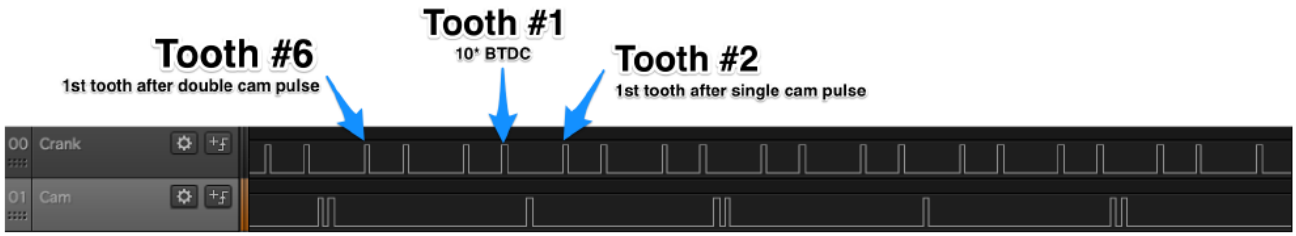


3.11.4　トリガーパターン

クランクホイールには、クランク角 70 度と 110 度で分割された 4 つのトリガー歯があります。

同期パルス（Sync pulse）は 1 サイクル当たり 2 つあり、分割されたクランクパルス（Primary pulse）の間に発生するカムパルス（Secondary pulse）の数によって識別されます。 2 パルスのカムパルスの後の最初に発生するクランクパルスは 6 番目のトリガー歯（Tooth #6）として識別され、 1 パルスのカムパルスの後の最初に発生するクランクパルスは 2 番目のトリガー歯（Tooth＃2）として識別される。Tooth #1 は BTDC10 度の位置にあり、Tooth＃2 と Tooth＃6 のみでは直接識別する事は出来ません。VVT によってカ

ムシャフトタイミングが変動しても、カムパルスによる同期パルス識別ウィンドウは機能します。よって VVT 値に関係なく、同期パルスはすべての負荷と速度で識別可能です。



3.12 Non-360

これはトリガー歯数が 360 度を整数で割り切れないデュアルホイールで使用するデコーダです。このデコーダは特定のブランドやエンジンシリーズに固有のものである為、以前は『Audi 135 デコーダ』等、タイプを識別する為の名称で呼ばれていました。このデコーダは様々な歯数で使用出来ますが、すべての歯数の組合せをこのデコーダーでカバー出来る訳ではありません。

3.13 Nissan 360

3.13.1 概要

Nissan 360 CAS トリガーは、日産の 4 気筒あるいは 6 気筒エンジンで使用されているデコーダーです。

カムに同期して回転するホイールに 360 のウィンドウを持ち、光学センサーでトリガーを読み取ります。

各ウィンドウ幅はクランク角度 2 度となります。位置情報については、シリンダー数に等しいウィンドウの内側リングもあります（4 気筒エンジンでは 4 ウィンドウ、6 気筒エンジンでは 6 ウィンドウ）。

注）4 気筒 CAS には複数のバージョンがあり、現在、全てがサポートされている訳ではありません。既知の各バージョンについて以下の通りです。

1. パターン 1：単一のユニークなインナーウィンドウがあるパターン。現在サポートされていません。
2. パターン 2：ユニークなスロットサイズは対になっているパターン。これは部分的にサポートされています。
3. パターン 3：各インナーウィンドウはそれぞれ異なるサイズのパターン。通常 4 気筒エンジンでは、4、8、12、16、6 気筒エンジンでは 4、8、12、16、20、24 です。これはサポートされています。

3.13.2 アプリケーション

- CA18：パターン 3
- SRxx Redtop：パターン 1
- SRxx Blacktop（early）：パターン 1
- SRxx ブラックトップ（ノッチ）：パターン 1
- FJ20：パターン 1
- RB30：パターン 1
- RB25/26：パターン 3

3.14 Daihatsu +1

3.14.1 概要

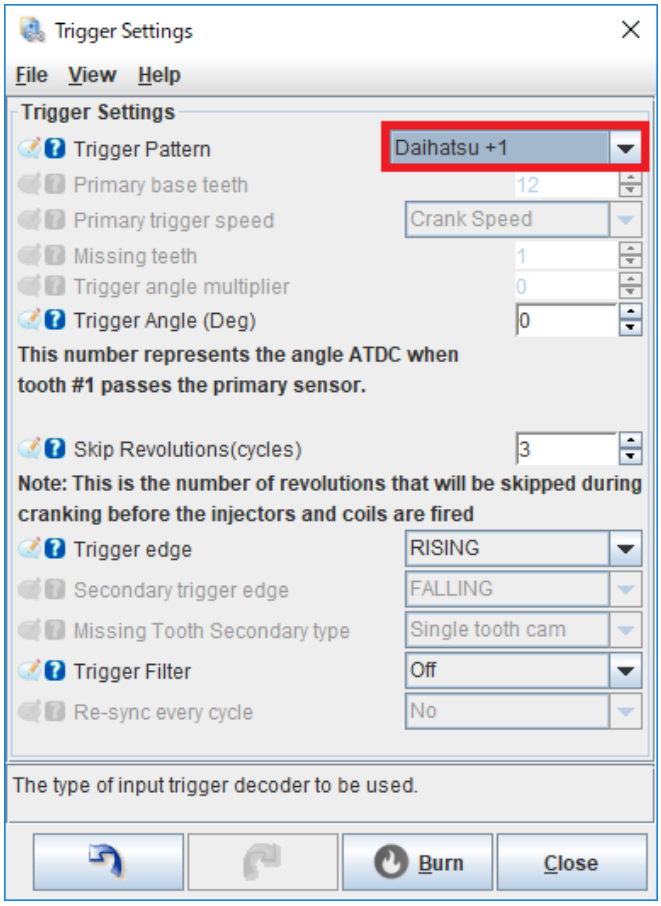
Daihatsu +1 のデコーダーは、ダイハツの 3 気筒エンジンと 4 気筒エンジンで使用されています。この方式はホールセンサと単一のカムホイールで構成されています。この信号は PJSC の VR1 ピンに入力します。

3.14.2 アプリケーション

- TBA（3 気筒）
- TBA（4 気筒）

3.14.3 Tuner Studio 設定

Trigger Settings ダイアログの Trigger Pattern プルダウンメニューにて、『Daihatsu +1』を選択して下さい。それ以外の項目の設定は不要です。



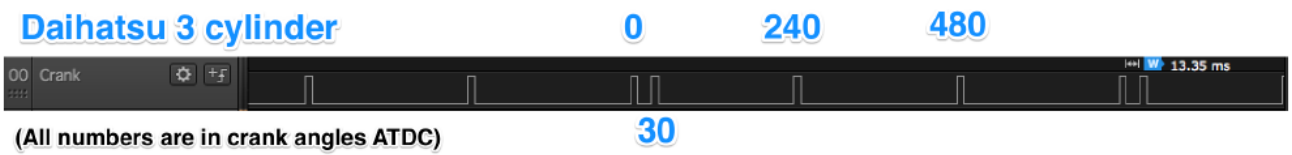
3.14.4　タイミング調整

ほとんどの場合トリガーアングルの変更は必要ありませんが、OEM バージョンのトリガー間隔には若干のばらつきがあるため調整が必要になる場合があります。エンジンが始動したら点火タイミングを固定にして角度を設定し、タイミングライトで点火タイミングを確認して下さい。点火タイミングがずれている場合、トリガーアングルを調整して下さい。

3.14.5　トリガーパターン

3 気筒エンジンではクランク角が 0、240、480 度の等間隔で 3 つのトリガー歯があり、更にクランク角 30 度の位置にインデックストリガーを識別する為のトリガー歯が一つあります。

4 気筒エンジンは等間隔のトリガーが 4 本である事以外、3 気筒エンジンと同様です。クランク角 0、30、180、360 度の等間隔で 4 つのトリガー歯と 540 度に一つのトリガー歯があります。



3.15　スバル 36-2-2-2

3.15.1　概要

36-2-2-2 のデコーダーは、2000 年以降の多くのスバル 4 気筒エンジン共通の方式です。

この方式のクランクホイールは、クランク角 10 度毎の 36 個のトリガー歯と 2 個の欠歯が 3 か所あります。これら 3 つの欠歯グループは、クランク 1/2 回転以内に同期（Sync）を識別可能にします。

初期の方式は VR センサーによりトリガー信号を出力していましたが、スバルが可変バルブタイミングを採用してからセンサーもホールセンサーに変更されました。

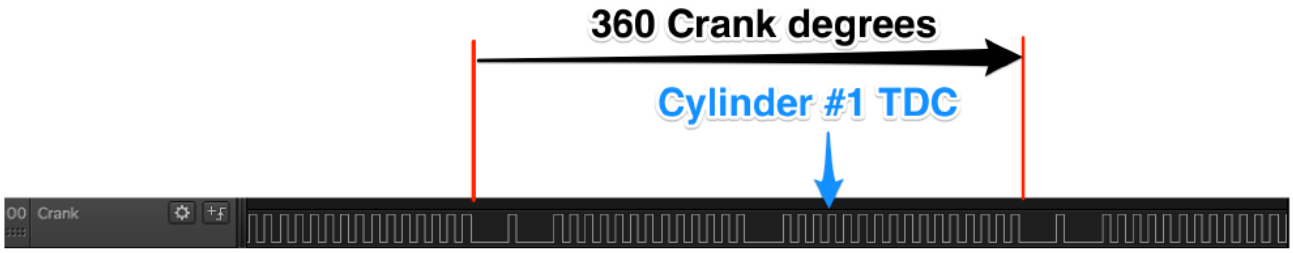
殆どの場合一つか二つの 4-1 カムセンサーと組合せて使用されますが、PJSC では同期の為にカムセンサーは必要ありません。

3.15.2　トリガーパターン

2 つの欠歯の 3 組の内、2 組は 1 本のトリガー歯を挟んで隣接して配置され、残りの 1 組は 180 度反対側に配置されています。同期パルスの検出アルゴリズムは、2 組の欠歯を検出して次に検出された 1 組の欠歯の直後のトリガー歯をインデクストリガーと識別する事で同期（Sync）をとります。

シリンダー#1 の TDC、1 組の欠歯から 4 番目のトリガー歯のタイミングでとなります。PJSC は欠歯の周期を監視し、予測出来るウィンドウの期間内に次の欠歯が続くかどうかを確認します。従って、同期は 1 回のクランク回転において 2 箇所で識別する事ができる。

注）オンラインで入手できる図やトリガーホイールの画像の多くはホイールを裏側から見たもので、反時計回りに回転しているように見えるので注意が必要です。歯の期間は、その後、それが別のものが続くかどうかを確認するために待機します。したがって、同期はこのようにして決定できます。



第 4 章　設定

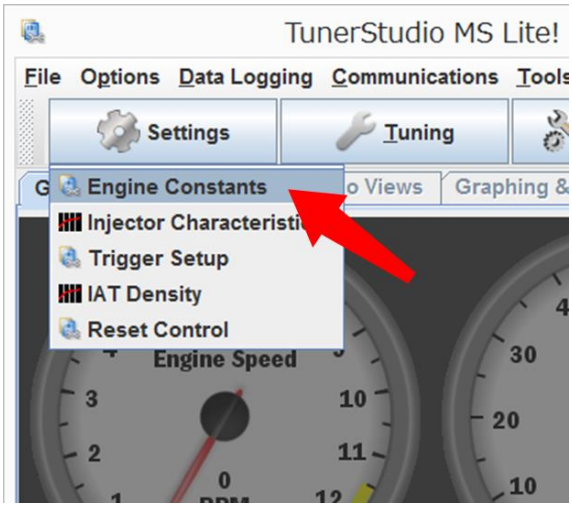
この章では、Tuner Studio を用いて PJSC の設定の方法を解説します。Tuner Studio で PJSC の設定を行う前に、以下の準備を行って下さい。

(a) PJSC とエンジン、センサーを正しく配線、接続して下さい。
(b) Windows または Linux がインストールされた PC、または OS X がインストールされた Mac を用意して下さい。
(c) Tuner Studio をインストールする PC に Java ランタイムライブラリをインストールして下さい。ランタイムライブラリは、http://www.java.com からダウンロード出来ます。
(d) EFI Analytics HP（http://www.tunerstudio.com/index.php/downloads）から最新バージョンの TunerStudio MS および MegaLogViewer MS をダウンロードして下さい。
(e) OSDN の PJSC Personal Forge（https://osdn.net/users/maharu/pf/PJSC/wiki/FrontPage）から PJSC の最新版ファームウェアをダウンロードし、ファームウェアをアップデートして下さい。

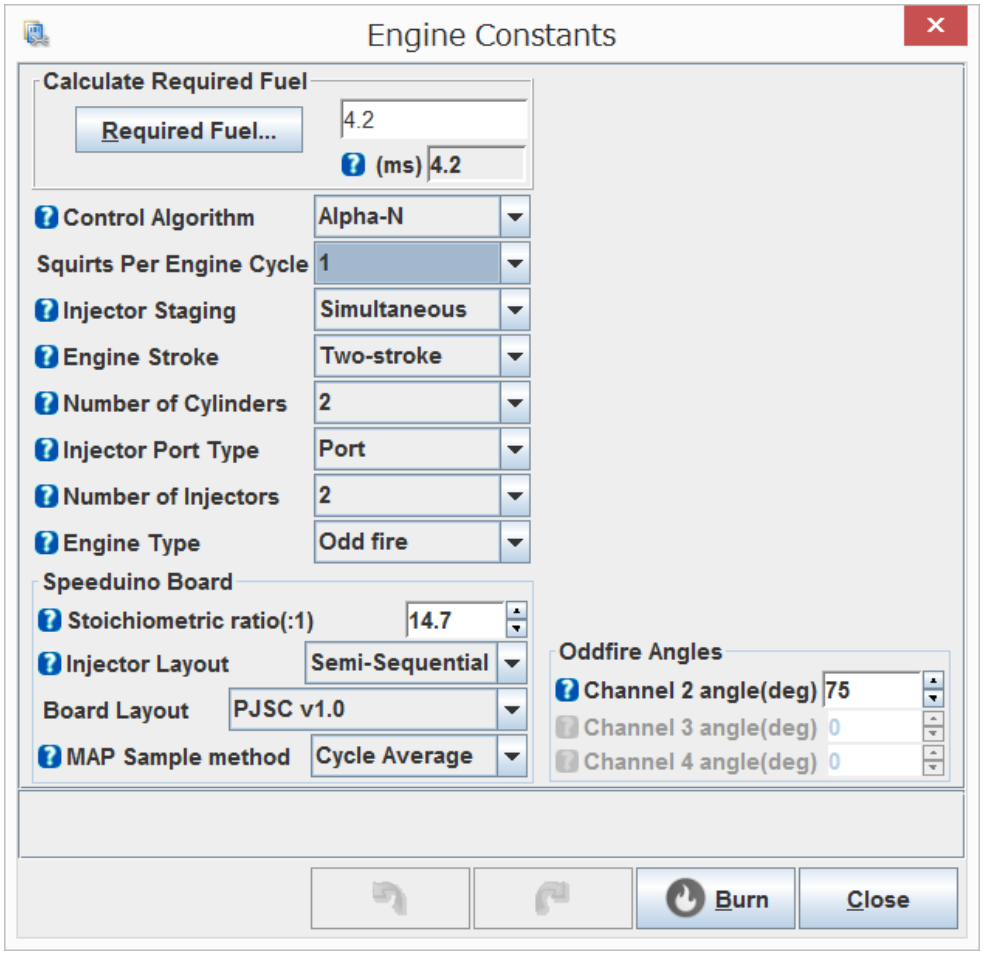
4.2　エンジン設定

4.2.1　概要

Settings メニューの"Engine Constants"を選択すると、Engine Constants ウィンドウが開きます。このウィンドウではエンジン形式、燃料噴射方式によって決まる項目を入力します。



4.2.2　設定



・**Injector Staging**：インジェクターの噴射順序を指定します。

・**Alternating（推奨）**：インジェクターの噴射タイミングを各気筒毎に TDC を基準とした同じピストン位置にします。インジェクターを閉じるタイミングは、Injector Characteristics ダイアログで指定します。

・**Simultaneous**：全てのインジェクターは同時に噴射します。噴射タイミングの基準は 1 番気筒の TDC となります。

・**Engine stroke**：2 ストロークか 4 ストロークで、使用するエンジンが該当する方を選択します

・**Number of cylinders**：使用するエンジンの気筒数を指定します。ロータリーエンジンの場合は 4 を選択します。

・**Injector Port Type**：インジェクター設置位置を指定します。但し、現行のファームウェアではこのパラメーターを使用していませんので、使用するエンジンと異なるポートタイプを選択してもエンジン制御に支障はありません。

・**Number of Injectors**：使用するエンジンに設置されているインジェクターの数を指定します。

・**Engine Type**：噴射タイミング（クランク角）の間隔が全てのシリンダーで等間隔の場合は"Even Fire" を選択、不等間隔の場合は"Odd Fire"を選択します。Odd Fire に該当するエンジン（例えばV型エンジンの多くが該当）を使用する場合、各出力チャンネル毎にの 1 番気筒を基準とした噴射タイミングオフセットをク

ランク角で指定します。例：V バンク角が 90 度の V 型 2 気筒 2 ストロークエンジンの場合、1 番気筒に対して 2 番気筒の噴射タイミングは 90 度オフセットとなります。この場合 Engine type は"Odd Fire"を選択し、"Odd Fire Angles" の "Channel 2 Angle(deg)" に 90 を入力します。

・**Injector Layout**：インジェクターへの結線方法に応じて、以下から選択します。

・**Paired**：1 出力当り、2 本のインジェクターを並列に接続します。全てのチャンネルの噴射タイミングは同時です（グループ噴射）。

・**Semi-Sequential**：インジェクター出力の Ch1/Ch4 と Ch2/Ch3 がそれぞれペアとなって同じタイミングで噴射されます。Ch1/Ch4 と Ch2/Ch3 はタイミングが 180 度（或いは 360 度）反転します。このモードは 4 気筒以下でのみ有効です。

・**Sequential**：1 チャンネル当たり 1 インジェクター或いは 1 気筒に対して噴射します。1 サイクル内で各気筒に順番に噴射を行います。Sequential は 4 気筒以下のエンジンで使用可能です。

・**Boad Layout**：Arduino と組み合わせて使用するボードの種類を指定します。ボード指定に応じて、Arduino の入出力ピン割り当てが変更されます。具体的なピン割り当ては utils.ino ファイルを参照して下さい。

・**MAP Sample Method**：

・**Instantaneous**：MAP センサー信号を Arduino が読み込む度、MAP センサー値をそのまま負荷値として使用します。MAP センサー値は非常に不安定なので燃調制御も不安定になってしまいます。マニホールド内圧力変化のデータを採りたい場合は有効です。

・**Cycle Average**：クランク角 720 度（4 ストロークエンジンの 1 サイクル）毎に MAP センサー値を平均した値を負荷値として使用します。シリンダー数が 4 気筒以上のエンジンに対して推奨です。

・**Cycle Minimum**：クランク角 720 度（4 ストロークエンジンの 1 サイクル）毎の MAP センサー値の最小値を負荷値として使用します。シリンダー数が 4 気筒未満のエンジン、或いは燃調制御方式に ITB を選択した場合、この方式を推奨します。

4.2.3 Required Fuel

PJSC で燃調セッティングを行うには、主に VE テーブルを編集して燃料噴射量を調整する事になります（詳細は第 4.9 章を参照して下さい）。

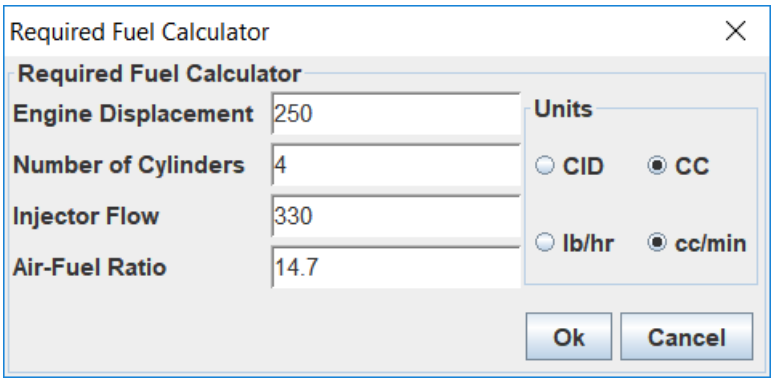
古い ECU の中には、燃調テーブルにインジェクターのバルブを開いている時間を指定する物があります。この方式ではインジェクターを容量が異なる物に変更した場合、燃調セッティングの為に燃調テーブルの値全てを入力し直す必要があります。

PJSC では、この様な場合でも燃調テーブルの値を入力し直さずにインジェクター変更前の燃調セッティング継承出来る VE 方式を採用しています。

VE は体積効率（Volumetric Efficiency）の略で、シリンダー容積に対する新気の充填効率を意味します。エンジン回転数と負荷（TPS/MAP）に対して新気の充填効率をテーブルで指定する事で、適切な燃料噴射量を計算によって求めます。この計算を行う為には、予めシリンダー容積とインジェクター容量、目標空燃比を指定しておく必要があります。以下はこれらのパラメーターを Tuner Studio から入力する方法の説明です。

4.2.4 Required Fuel Calculator

Engine Constants ダイアログ内の”Required Fuel”ボタンをクリックすると、以下の様な Required Fuel Calculator ダイアログが表示されます。



・**Engine Displacement**：エンジンの総排気量を入力して下さい。

・**Number of Cylinders**：エンジンの気筒数を入力して下さい。PJSC はここに入力された値と総排気量から単室容積を計算します。

・**Injector Flow**：インジェクター容量を入力して下さい。

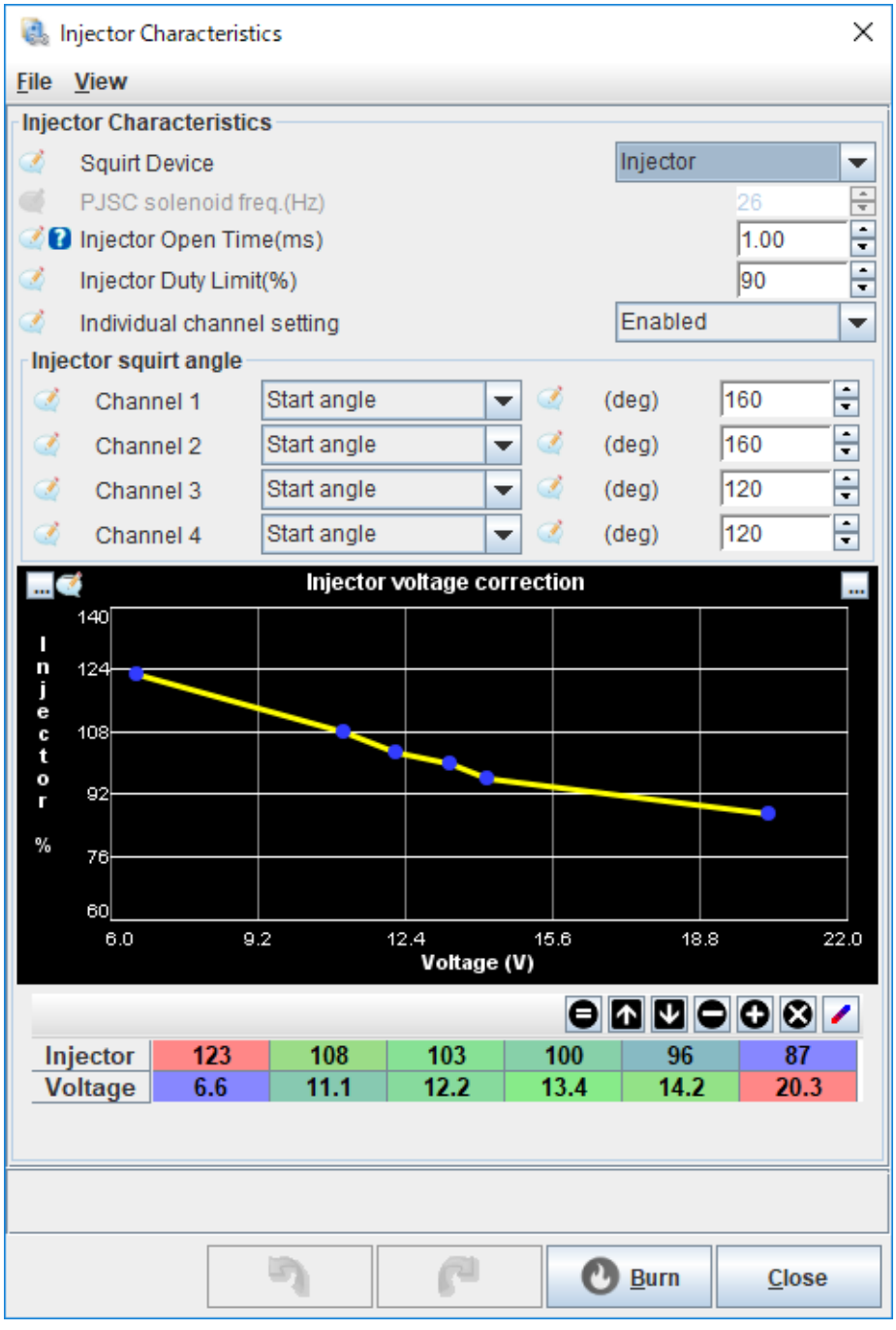
・**Air-Fuel Ratio**：目標空燃比を入力して下さい。通常は 14.7 を入力します。

4.3 Injector Characteristics

4.3.1 概要

Settings メニューの"Injector Characteristics"を選択すると、Injector Characteristics ウィンドウが開きます。このウィンドウでは燃料噴射デバイスの選択、無効噴射時間等の設定を入力します。

4.3.2 設定



・**Squirt Device**：使用する燃料噴射デバイス（インジェクターまたはソレノイドバルブ）を選択します。

・**PJSC solenoid freq.(Hz)**：ソレノイドバルブを燃料噴射デバイスとして選択した場合、周波数固定の PWM 信号でソレノイドバルブを駆動します。PWM 周波数を入力します。

・**Injector Open Time(ms)**：インジェクターを燃料噴射デバイスとして選択した場合、無効噴射時間を入力します。

・**Injector Duty Limit(%)**：インジェクターの噴射信号のデューティ比をこの値（%）に制限します。インジェクター噴射時間は様々な補正を行って算出されますが、計算後のデューティ比がこの値を超えた場合噴射時間をリミット値に変更します。

・**Individual channel setting**：この設定を Enable にすると、4 チャンネルあるインジェクター出力に対して噴射タイミングを個別に設定する事が出来ます。

・**Injector squirt angle**：インジェクター噴射タイミングはクランク角で指定します。指定したクランク角で噴射開始とする場合は“tart angle”を選択し、噴射終了とする場合は“End angle”を選択します。

・**Injector voltage correction curve**：インジェクターは供給される電圧によって開弁時間変わってきます。その結果燃料噴射量も変わってしまう為、電圧に応じて噴射量補正します。その補正値を Injector Voltage correction curve で任意に設定する事が可能です。

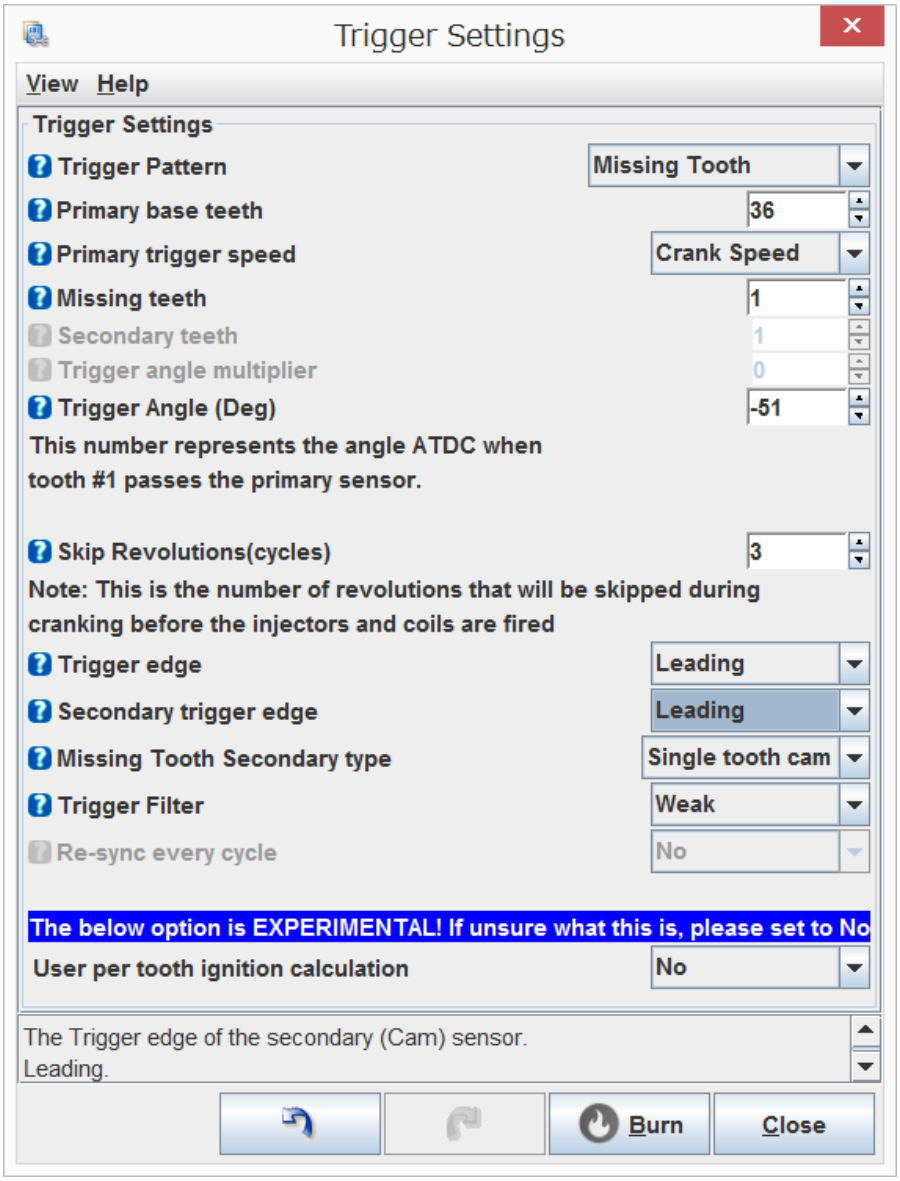
4.4　　トリガー設定

4.4.1　概要

クランク角センサー（CAS）とセンサー信号の識別方法は、EFI チューニングを行う上で最も重要な要素です。Trigger settings ダイアログ内の項目で、トリガー信号を識別する為に必要なトリガーの仕様を指定します。これらの項目に間違った値が指定されていると sync（クランク角同期信号）と回転数を誤って識別し、最悪のケースではエンジンを損壊してしまいます。エンジンを始動する前に、必ず正しい設定を行って下さい。

註）Trigger Settings ダイアログには多くの設定項目がありますが、トリガーの仕様に無関係な項目はグレーアウトされ編集出来なくなります。

4.4.2　Trigger Settings



・**Trigger Pattern**：クランク/カムセンサーのパターンを指定します。サポートされているパターンについては、『第３章 デコーダー』を参照して下さい。

・**Primary Base teeth**：プライマリーホイール上の歯数を指定します。ミッシングトゥースホイールの場合、欠歯の数も加えて Primary Base teeth とします（欠歯が無い状態のホイールとして歯数をカウントする）。例えば、12-1 のミッシングトゥースの場合、11（実際にホイール上にある歯数）＋1（欠歯の数）＝12 が Primary Base teeth となります。

・**Primary trigger speed**：プライマリーホイールがクランクと同期して回転している場合は"Crank Speed"を、カムと同期して回転している場合は"Cam Speed"を選択します。この項目は Primary Base teeth で指定した歯数がクランクが 1 回転する度にセンサーを通過する歯数なのか、或いはカムが 1 回転する度にセンサーを通過する歯数なのかを指定します。

・**Missing teeth**：トリガーにミッシングトゥースパターンを使用している場合、欠歯の歯数を指定します。例えば 36-1 の場合は"1"を、60-2 の場合は"2"を入力します。欠歯の配置はホイール一周中に一カ所のみをサポートしています。 1 周中に 2 カ所以上欠歯があるパターンには対応していません。

・**Secondary teeth**：Trigger Pattern で Dual wheel を指定した場合、セカンダリーホイールの歯数を入力します。通常はカムホイールをセカンダリーホイールとします。

・**Trigger angle multiplier**：

・**Trigger angle**：プライマリーホイールの最初の歯（例えばミッシングトゥースであれば欠歯の後の最初の歯－インデックストリガー）が、センサーを通過して信号が生成される時のクランク角を ATDC で入力します。例えば ATDC5 度でインデックストリガーが検出される場合は"5"を入力します。BTDC の場合はマイナスになります。例えば、インデックストリガーが検出されるクランク角が BTDC30 度の場合"-30"を入力します。

・**Skip revolutions(cycles)**：クランキング初期はクランク回転速度が安定していない為、トリガー信号の検出、同期が困難な場合があります。その様な場合、数回転分のトリガー信号を無視して回転が安定してから回転速度計算を行うとエンジンが始動し易くなります。Skip revolutions(cycles)に 1 以上の数値を指定すると、クランキングで最初にトリガー信号が発生してから指定された数値（回転）分のトリガー信号を無視します。

・**Trigger edge**：トリガーを信号の立ち上がりエッジで検出するか、立下りエッジで検出するか選択します。'RISING'/'LEADING'を選択すると信号の立ち上がりエッジで、'FALLING'/'TRAILING'を選択すると立下りエッジでトリガーを検出します。

・**Secondary trigger edge**：Trigger Pattern で Dual wheel を指定した場合、セカンダリーホイールの信号の立ち上がりでトリガーを検出するか、立下りで検出するかを選択します。

・**Missingtooth Secondary type**：Missingtooth を選択して Missingtooth ホイールをクランクに装着し、且つカムにもホイールを装着した場合、セカンダリーホイールのトリガー型式をここで選択します。セカンダリーカムホイールには 4-1Missingtooth ホイールか、シングルトリガーのホイールを選択出来ます。セカンダ

リーホイールで検出されたトリガーは、4サイクルエンジンのサイクルスタートタイミングの基準となる「シンク（Sync）」パルスとなります。

・**Trigger Filter：**トリガー信号にノイズが乗ってトリガーの誤検出が発生するのを防ぐ為、トリガー信号はADC ポートで Arduino に取り込んだ後ソフトウェアによるフィルターを設定する事が出来ます。このフィルターの強弱をここで選択します。「OFF」はフィルタリングを行わず、「Aggressive」が最も強くフィルタリングを行います。フィルターの強度を上げるとノイズによる誤検出を減らせますが、本来のトリガーの信号レベルも下がって感度が低下し、クランキングや低回転でトリガーを検出し難くなってしまいます。トリガー信号の仕様に合わせて適切なフィルタリングレベルを選択して下さい。

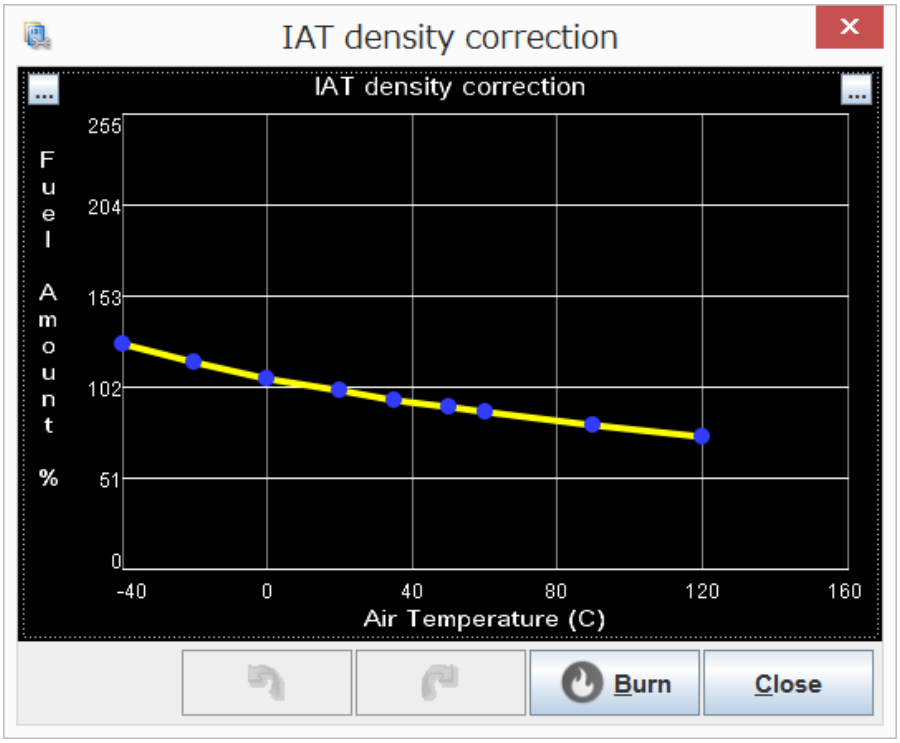
・**Re-sync every cycle：**これを有効にする（「Yes」を選択する）と、シンクパルスの検出ウィンドウを常に開きます。これによってシンクパルスの検出感度が高くなりますが、シンク信号ラインにノイズが乗っていると誤検出もし易くなってしまいます。「No」を選択した場合はシンクパルスを検出した際、その時のエンジン回転数から次のサイクルのシンクパルスが発生するタイミングを予測してシンクパルス検出ウィンドウを開くタイミングを決定します。通常は「No」を選択する事を推奨します。

4.5　IAT Density（吸気酸素密度）

4.5.1　概要

IAT Density カーブは、吸気温度による酸素密度の変化を示しています。デフォルトカーブはボイルシャルルの法則に則った数値であり、通常はこのまま使用する事が出来ます。しかしエンジンルーム内が高温になる場合や、過給によって吸気温度が高温になりボイルシャルルの法則から外れる様な場合は、IAT Density カーブを修正する必要があります。

4.5.2　セッティング



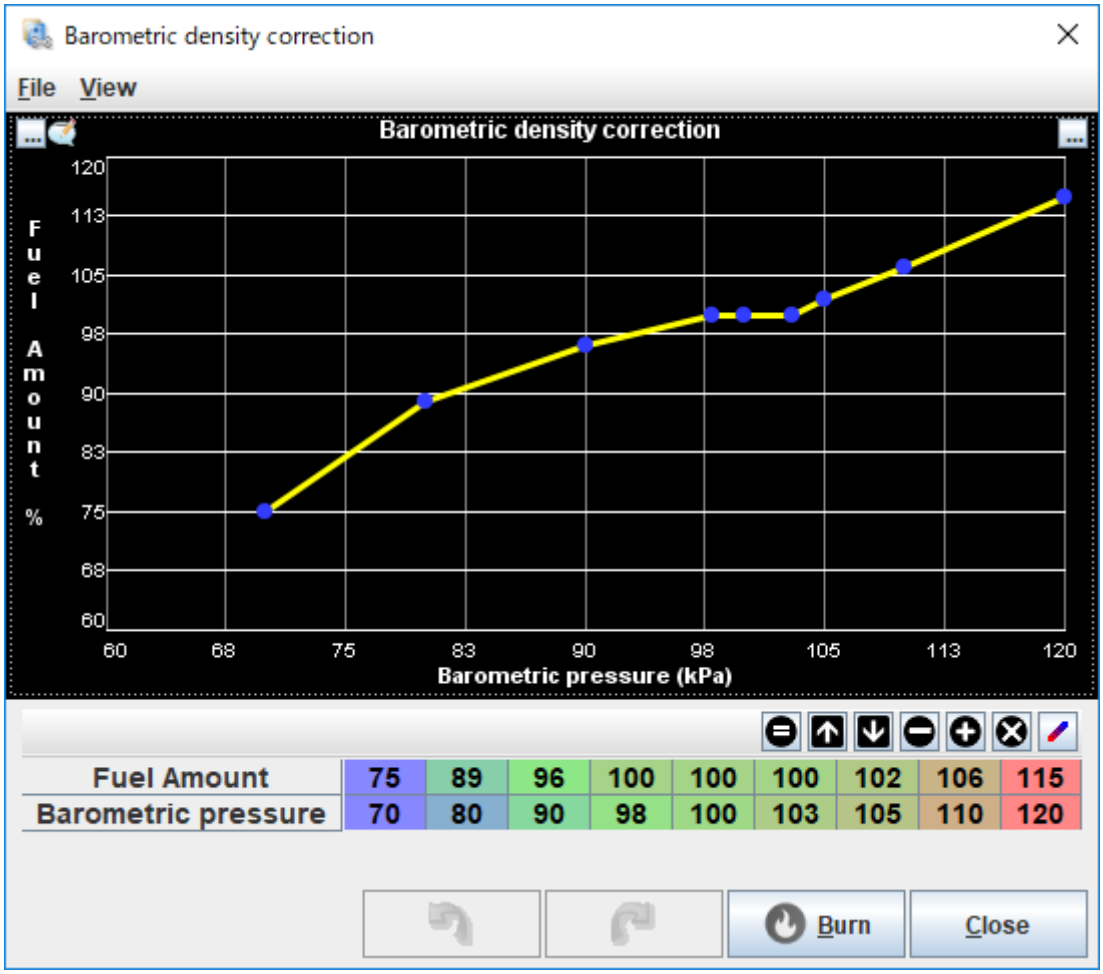
4.6　Barometric correction curve（大気圧補正カーブ）

4.6.1　概要

PJSC では大気圧に応じた燃料噴射量の補正方法が二通りあります。一つは大気圧（Baro）センサーの出力値と、インテークマニホールド圧力（MAP）センサーの出力値の比で補正値を計算する方法です。これは VE テーブルコンフィギュレーションダイアログ内の“Multiply VE value by MAP:Baro ratio”で有効にする事が出来ます（4.10 VE Table configration を参照）。

もう一つは大気圧（Baro）センサーの出力値に応じて補正値を決める方法で、以下の Barometric correction カーブで任意に設定する事が出来ます。

4.6.2 セッティング



4.7 加速補正

4.7.1 概要

加速補正（Acceleration Enrichment - AE）はスロットルを短時間で急に開いた場合、燃料を増量する補正です。この機能は負荷が急に増えると燃料噴射量を増やして、キャブレターの加速ポンプと同等の機能を再現します。

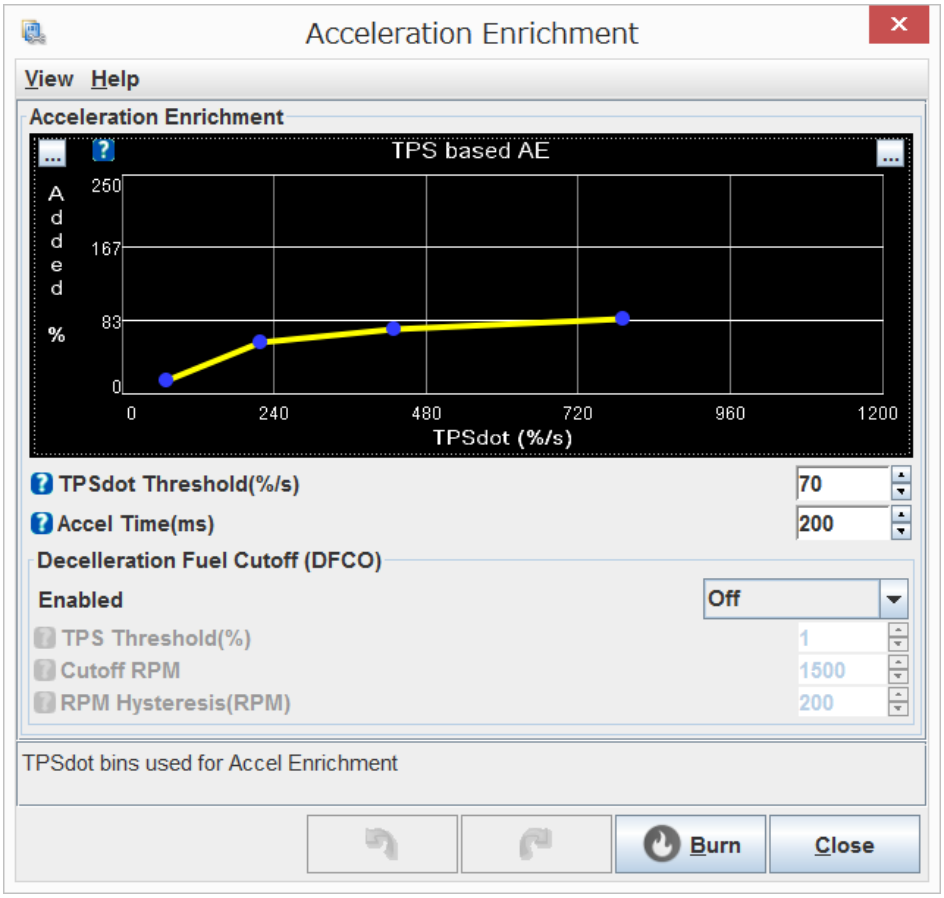
この機能を使用するには、予め TPS（スロットルポジションセンサー）を装着してキャリブレーションを実施しておく必要があります。

4.7.2　理論

加速補正はスロットルポジションの変化率－TPSdot（TPS delta over time）に基づいて算出します。TPSdot の単位は[%/s]で、値が大きい程速いスピードでスロットルが開かれている事になります。通常のエンジンであれば 50[%/s]から 1000[%/s]に収まります。

・100[%/s]＝1 秒で TPS が 0%から 100%になるまでスロットルが開かれた場合
・1000[%/s]＝0.1 秒で TPS が 0%から 100%になるまでスロットルが開かれた場合

加速補正カーブの X 軸は TPSdot を示し、Y 軸は燃料補正率（%）を示します。例えば補正率が 100%の場合、燃料噴射量が 2 倍になります。



・**TPSdot Threshold(%/s)：**加速補正の有効／無効を切り替える TPSdot の値を入力します。TPSdot がこの値より大きいと加速補正は有効となり、TPSdot がこの値以下の時は加速補正は無効となります。

・**Accel Time(ms)：**加速補正が有効になった時に、噴射増量が継続される時間を入力します。

・**Decelleration Fuel Cutoff(DFCO)：**アクセルを閉じた時に燃料噴射をカットする条件を指定します。

　・**Enable：**On にすると燃料噴射カット機能が有効になります。

Tuning

デフォルト設定ファイルに含まれている加速補正カーブは大抵のエンジンに適用出来るカーブとなっていますが、インジェクターのサイズやスロットル径に応じて修正する事を推奨します。

加速補正カーブのチューニングはダイノベンチ上あるいは実走しながら行うのがベストですが、車両停止状態の空ぶかしでも可能です。その場合 AF センサー装着し、AE ダイアログを開いて AF 値と AE 値の変化を確認しながら行うと良いでしょう。AE ダイアログ上のグラフでは TPAdot と AF 値を併せてリアルタイムでモニター出来、また加速補正カーブ上のどのポイントが参照されているかが判るので修正が必要な箇所が判別し易いでしょう。

定常状態（スロットルを急操作していない状態）で AF 値が適正でありながらスロットル急開で一時的に AF が薄くなる場合、Accel Time を 10-20ms ずつ増やして AF が薄くならないポイントを探すと良いでしょう。

誤検出

TPS 信号にノイズが乗っている場合、スパイクノイズによってアクセルが急開されたと誤検出して加速補正により燃料が増量されてしまう場合があります。これはログを見るか、TunerStudio のリアルタイムモニター画面の"TPS Accel"で確認する事が出来ます。この誤検出が生じた場合、TPSdot Threshold の値を増やす事で誤検出を回避する事が出来ます。TPSdot Threshold を増やす時は 1 回につき 5%以下に止め、誤検出が発生するかモニターします。これを繰り返して誤検出しなくなるポイントを探します。

4.8 AFR/O2

4.8.1 概要

PJSC は O2 センサーが検知する排気中に含まれる酸素濃度に応じて、燃料のクローズドループ制御を行う事が出来ます。AFR（Air Fuel Ratio）/O2 ダイアログはクローズドループ制御の設定を行う事が出来ます。AFR クローズドループシステムは AFR テーブルの値を参照してターゲット値とし、O2 センサーの値がターゲットと一致する様に燃料噴射量のフィードバック制御を行います。

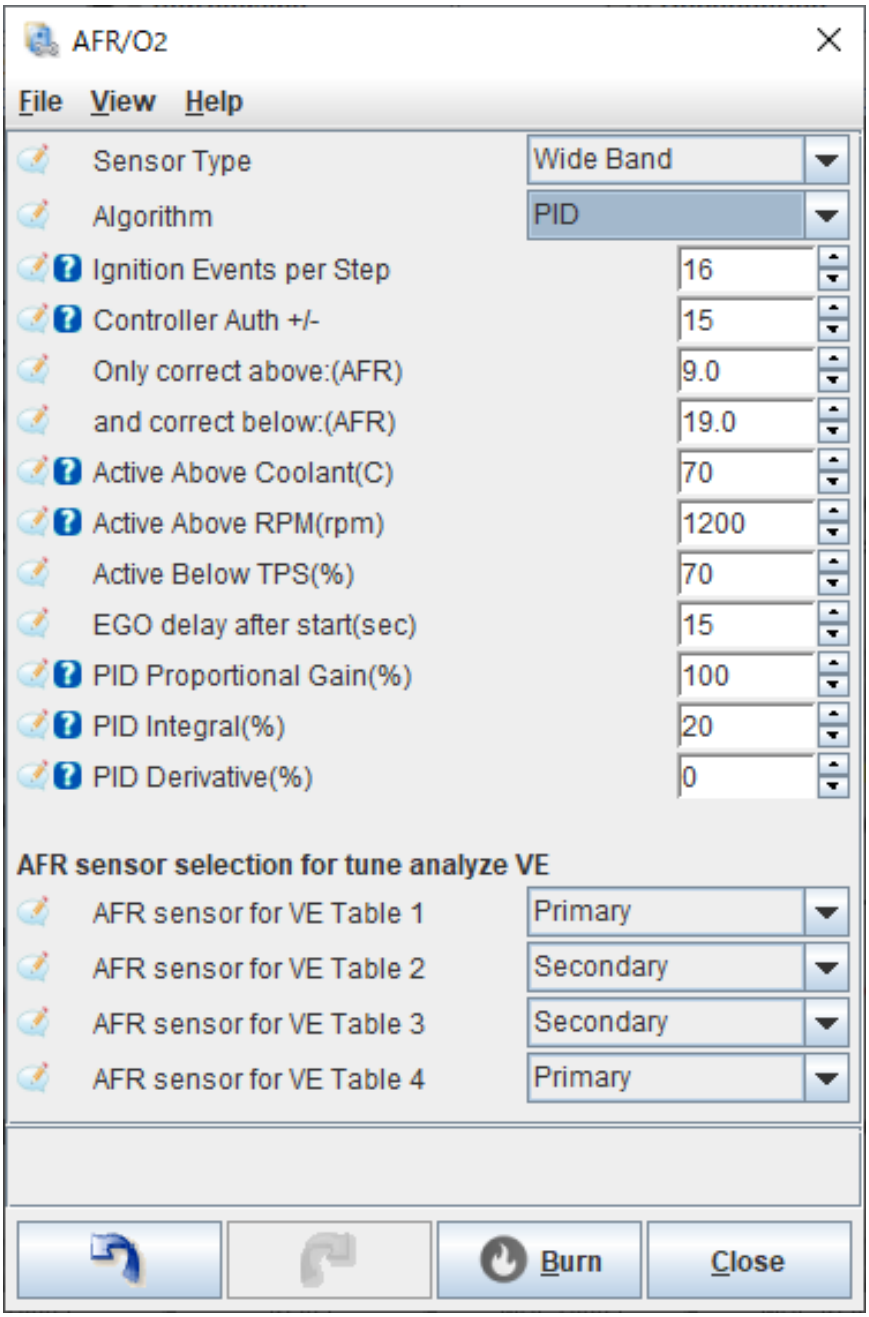
AFR クローズドループ制御機能を使用する場合、O2 センサーはワイドバンドセンサーを用いる事を推奨します。ナローバンドセンサーでもクローズドループ制御機能は可能ですが、制御の精度は低くなってしまいます。

註）AFR クローズドループ制御は、燃調設定が適切ではない状態を補償するものではありません。適切な燃調設定がされたシステムでは AFR クローズドループ制御は使われない、或いは僅かな補正としてのみ働きます。

4.8.2 セッティング

PJSC は 2 つのクローズドループアルゴリズムをサポートしています。

1. Simple：これは時系列ターゲット AFR 追従型のアルゴリズムで、ターゲット AFR に対して燃調が濃い時間、或いは薄い時間に応じて燃料を増減し、O2 センサー値がターゲット AFR 値に近づくように制御します。このアルゴリズムは、燃調が濃いか薄いかのみを判別するナローバンドセンサーを使用する場合に適しています。しかし燃調マップ（VE マップ）が適切にチューニングされないままこのアルゴリズムを有効にすると、正しく制御出来ない場合があります。その様な場合インジェクター駆動信号、或いは AFR 値が発振してしまいます。燃調マップのチューニングが完了するまで、この機能は無効にしておくべきです。
2. PID：クローズドループ制御を行う場合、ワイドバンドセンサーと PID 制御アルゴリズムを組み合わせた方がより良い結果を得られます、



・**Sensor type**：使用している O2 センサーに合わせて、"Narrowband"または"Wideband"を選択します。ナローバンドセンサーは出力が 0-1V の物を、ワイドバンドセンサーは 0-5V の信号の物を使います。ワイドバンドセンサーを使用する場合 TunerStudio メインメニューから Tools->Calibrate ARR Table ダイアログを表示し、キャリブレーションを実施します。

・**Algorithm**：Simple または PID を選択すると燃料クローズドループ制御が有効になります。

・**Ignition event per step**：AFR 補正計算は点火サイクルごとに実行されます。通常クローズドループの補正による O2 センサー出力の変化にはタイムラグがあります。この値を大きくすると、タイムラグの分フィードバックを遅らせて誤った補正が生じるのを防ぐ事が出来ます。通常この値は 2-5 程度です。

・Controller step size

・Controller Auth：クローズドループ補正出来るインジェクター出力パルス幅の最大値（%）です。補正値がこの値より大きくなった場合、補正値はこの値に制限されます。この値は 20%以下が推奨です。

・Correct above/below AFR：クローズドループによる AFR 補正が適用される、AFR の範囲です。AFR がこの範囲に収まっていない場合、AFR 補正は行われません。O2 センサーとコントローラーには正確に AFR を計測出来る範囲があり、この範囲外では誤って補正されてしまうので AFR 補正が有効になる範囲を制限します。

・Active above Coolant：クローズドループ補正は、エンジンの温度が適正な状態で行われなければなりません。これは正常な燃焼状態が維持されている時のみ補正が機能するようにする為です。この値は、エンジンの適温を指定します。

・Active above RPM：クローズドループ補正は通常、アイドリング時は無効にします。この値は補正が有効になるエンジン回転数の下限を指定します。回転がこの値以下の時は、補正は行われません。

・Active below TPS：TPS の値がこの値以上の時、AFR 補正は無効になります。

・EGO delay after start：O2 センサーは電源投入後にウォームアップが必要で、この間は AFR を測定出来ません。その間 AFR 補正を無効にする為に、この値としてウォームアップに必要な時間を入力します。ウォームアップ時間はセンサー、コントローラーによって様々なので実際にウォームアップに掛る時間を測定し、その値に 5s を足した値を入力する事を推奨します。

PID 制御パラメーター

・P/I/D：PID 制御のフィードバックゲインを指定します。（PID Proportional Gain, Integral and Derivative percentages.）

AFR sensor selection for tune analyze VE （PJSC ver1.02 追加）

PJSC ver1.02 でセカンダリーO2 センサー（O2_2）をサポートし、二つの O2 センサーを同時に使用する事が可能になりました。二つの O2 センサー（プライマリ／セカンダリ）を使用して VE テーブルオートチューンを行う場合、VE テーブル毎にプライマリ／セカンダリ何れのセンサー信号で学習するか選択します。

・AFR sensor for VE Table 1：VE テーブル 1 の学習を行う際、プライマリ／セカンダリ何れの O2 センサーを使用するか指定します。

・AFR sensor for VE Table 2：VE テーブル 2 の学習を行う際、プライマリ／セカンダリ何れの O2 センサーを使用するか指定します。

・AFR sensor for VE Table 3：VE テーブル 3 の学習を行う際、プライマリ／セカンダリ何れの O2 センサーを使用するか指定します。

・AFR sensor for VE Table 4：VE テーブル 4 の学習を行う際、プライマリ／セカンダリ何れの O2 センサーを使用するか指定します。

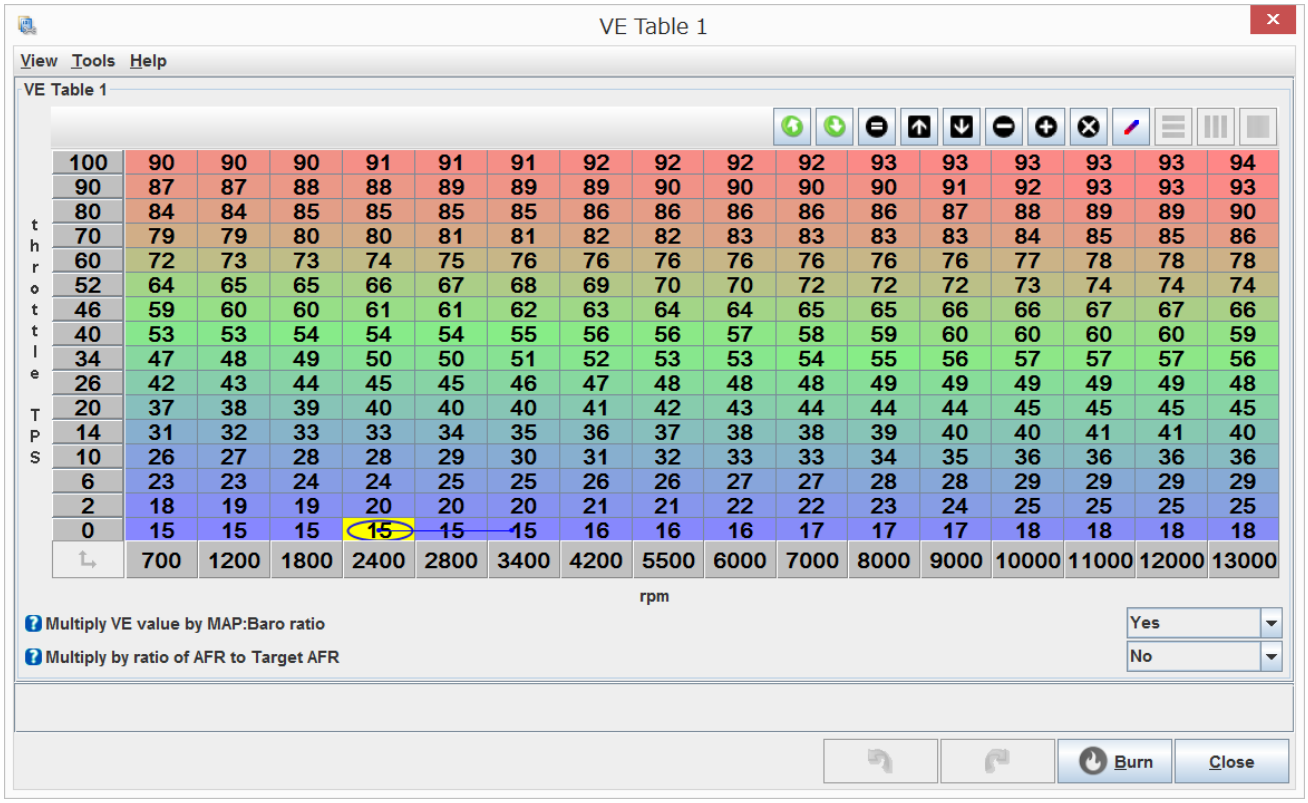
4.9　VE テーブル

4.9.1　概要

PJSC は VE（Volumetric Efficiency、体積効率）を元にインジェクターの燃料噴射量（開弁時間）を計算します。VE はシリンダー容積に対して、実際に新気が吸入される割合（%）を示した値です。VE テーブルはこの値をエンジン回転数とエンジン負荷（TPS または MAP）毎に指定する事で、回転数と負荷に応じた燃料噴射量を計算出来る様にするテーブルです。

4.9.2　セッティング

Tuning メニューの"VE Table"を選択すると、以下の様な VE テーブルダイアログが表示されます。



・**Multiply VE value by MAP:Baro ratio**：MAP センサーを使用していて且つこの機能を有効にすると、噴射時間を計算する際 VE 値に大気圧（Baro）に対する MAP 値の割合（%）を掛けます。燃調アルゴリズムにスピードデンシティを選択した場合、自動的に大気圧補正が掛る事になります。

・**Multiply by ratio of AFR to Target AFR**：この機能を有効にする（"Yes"を選択）と、AFR テーブルで設定した値をターゲット AFR として噴射時間にフィードバック補正を加えます。この機能はワイドバンドセンサーでの EGO フィードバックを有効にすると選択出来ます。

4.10　VE テーブルセレクション

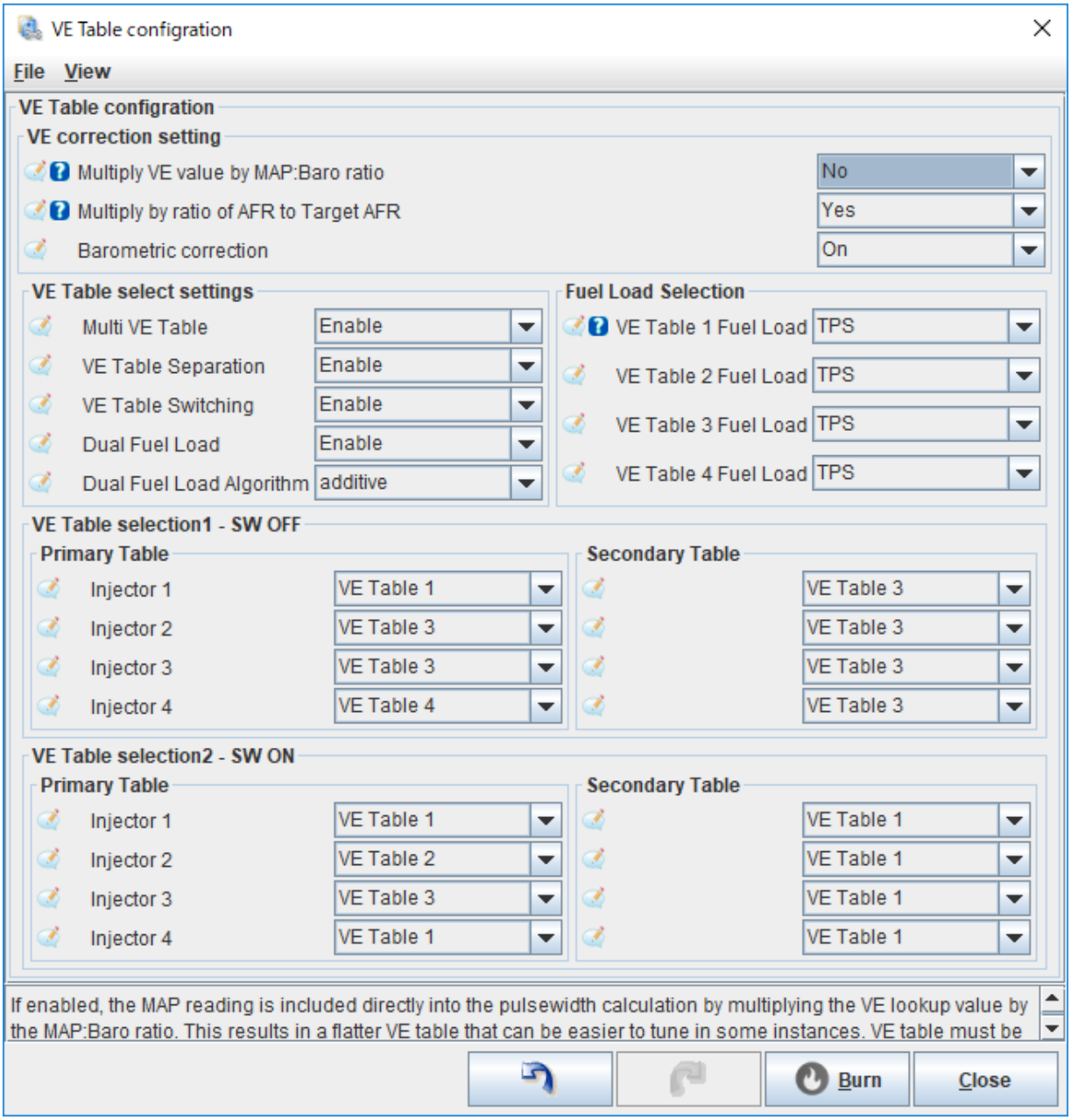
4.10.1　概要

PJSC は VE テーブルを最大で 4 テーブル記憶する事が出来ます（ver1.01 で機能追加）。4 テーブルを任意のインジェクター出力に割当てたり、２種類の異なる負荷軸（例えば TPS と MAP）のテーブルを組み合わせて両負荷特性に対応した VE テーブルを作成する事が可能です。

4 テーブルの使用方法は VE Table configration ダイアログにて設定します。

4.10.2　セッティング

Tuning メニューの"VE Table configration"を選択すると、以下の様な VE テーブルコンフィギュレーションダイアログが表示されます。



・**Multiply VE value by MAP:Baro ratio**：MAP センサーを使用していて且つこの機能を有効にすると、噴射時間を計算する際 VE 値に大気圧（Baro）に対する MAP 値の割合（%）を掛けます。燃調アルゴリズムにスピードデンシティを選択した場合、自動的に大気圧補正が掛る事になります。

・**Multiply by ratio of AFR to Target AFR**：この機能を有効にする（”Yes”を選択）と、AFR テーブルで設定した値をターゲット AFR として噴射時間にフィードバック補正を加えます。この機能はワイドバンドセンサーでの EGO フィードバックを有効にすると選択出来ます。

・**Barometric correction**：大気圧（Baro）センサーに応じて噴射量に補正をかけたい場合、”On”を選択してこの機能を有効にします。

・**VE Table select setting**：VE テーブルの割当て方を指定します。

・**Multi VE Table**：この機能を有効（Enable）にすると、VE テーブル 2～4 が使用可能になります。無効（Disable）を選択した場合、VE テーブル 1 のみ使用可能となり、後述のインジェクター毎の VE テーブル割当てや Dual Fuel Load 機能は使用不可となります。

・**VE Table switching**：この機能を有効（Enable）にすると、入力 2（Spare Input port2）に接続した外部スイッチによって VE テーブルの切り替えが可能となります。（注）PJSC ver1.0 及び ver1.01 ボードでは、この機能はサポートされていません。

・**Dual Fuel Load**：この機能を有効（Enable）にすると、VE テーブルを 2 つ使用して複数の負荷軸から燃料噴射量を計算する事が可能となります。インジェクター出力に対して 2 つの VE テーブルを割当て、片方の負荷軸を TPS、もう片方の負荷軸を MAP として二つの VE テーブルの値を加算或いは乗算して VE 値を求めます。過給器を装着した車両で、α－N を基本としながらインテークマニホールド圧力に応じて噴射量を補正したい場合等で有効です。

・**Dual Fuel Load Algorithm**：Dual Fuel Load を有効にした場合、 2 つの VE テーブルを加算（Additive）するか乗算（Multiply）するか選択します。

・**Fuel Load Selection**：各 VE テーブルの負荷軸を選択します。

・**VE Table Selection1 - SW OFF**：VE テーブル切替スイッチが OFF の時に、各インジェクター出力に割当てられる VE テーブルを選択します。

・**Primary Table**：各インジェクター出力の基本となる主 VE テーブルを選択します。Dual Fuel Load 機能が無効の場合、Primary Table で指定した VE テーブルのみから VE 値を参照します。

・**Secoundary Table**：各インジェクター出力の副 VE テーブルを選択します。Dual Fuel Load 機能が有効な場合、主 VE テーブルと副 VE テーブルを加算あるいは乗算して VE 値を算出します。

・**VE Table Selection2 - SW ON**：VE テーブル切替スイッチが ON の時に、各インジェクター出力に割当てられる VE テーブルを選択します。

・**Primary Table**：各インジェクター出力の基本となる主 VE テーブルを選択します。Dual Fuel Load 機能が無効の場合、Primary Table で指定した VE テーブルのみから VE 値を参照します。

・**Secoundary Table**：各インジェクター出力の副 VE テーブルを選択します。Dual Fuel Load 機能が有効な場合、主 VE テーブルと副 VE テーブルを加算あるいは乗算して VE 値を算出します。

4.11　ステージドインジェクション

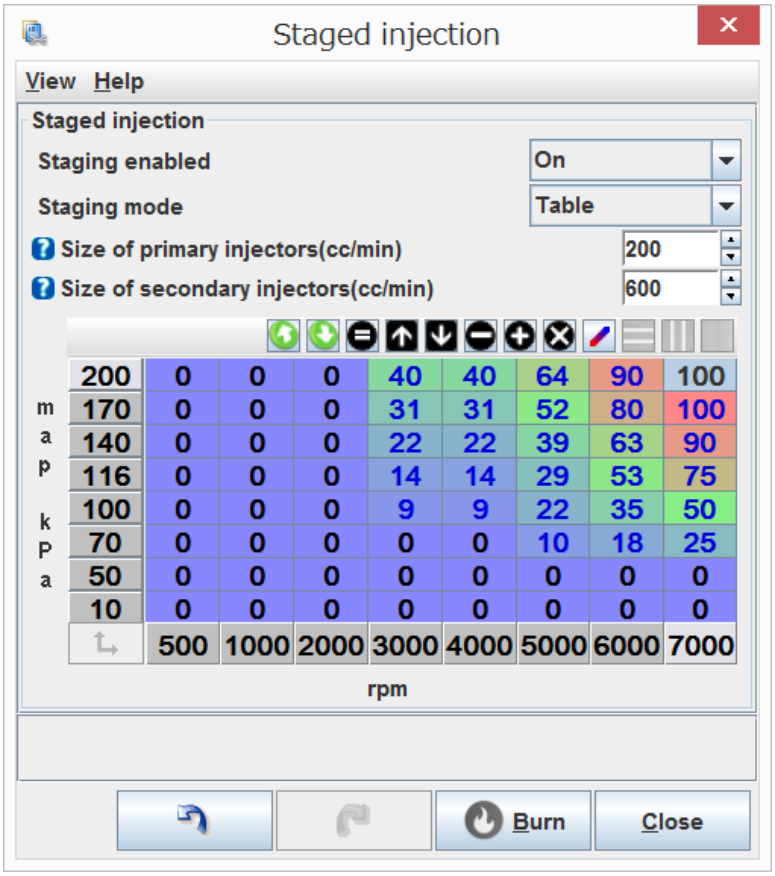
4.11.1　概要

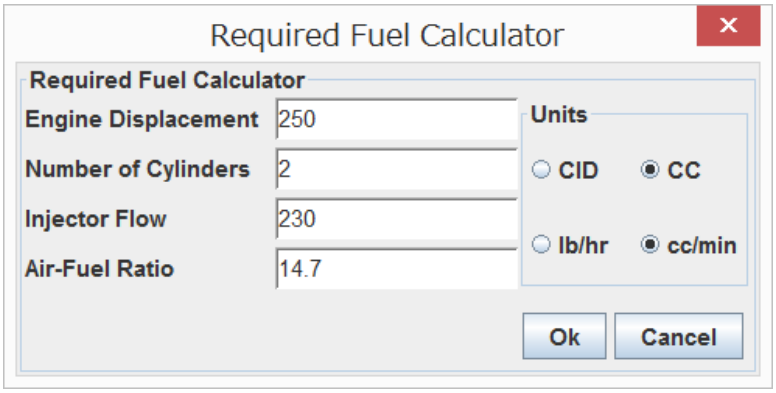
PJSC はツインインジェクターの 2 ステージ制御が可能です。セカンダリインジェクタを装備しているエンジンでは、高回転で大容量の燃料噴射能力と低回転での良好な霧化特性を両立する為にステージドインジェクション制御が用いられます。大容量のインジェクターは高回転で多くの燃料が要求される場合でも十分な燃料を供給出来ますが、低回転で噴射量が少ないと良好な霧化特性が得られないというデメリットがあります。これを補う為に、低回転では少量の噴射でも良好な霧化特性が得られる小容量のインジェクターで燃料を供給し、高回転で多くの燃料が要求される時は小容量と大容量の二つのインジェクターから燃料を供給する方式がステージドインジェクションです。

[重要] ステージドインジェクションを有効に切り替えた場合、必ず Engine Constants ダイアログの req-Fuel を更新して下さい。ステージドインジェクションが有効な場合、要求燃料を算出する為のパラメーターに入力する値は、プライマリーおよびセカンダリーインジェクタサイズの合計に等しくなるようにして下さい。

これらの値を正しく設定しないと、燃調が過剰に濃いまたは薄い状態になります。

例：
一次インジェクター：300cc
二次インジェクター：700cc
req_fuel 計算機に入力する値：1000cc





制御方式

PJSC はステージドインジェクションの制御方式として 2 つの方式を持っており、それぞれに長所と短所があります。通常は VE テーブルのチューニングのみで済む自動モード（Staging Mode で"Automatic"を選択します）から始めることを推奨します。 Automatic モードで適切な燃料配分が出来ない場合、手動で Fuel Staging Table を編集する手動モード（Staging Mode で"Manual"を選択します）に切替えて下さい。

Table Control

テーブルコントロールを使用すると、マニュアル

テーブルコントロールでは手動で入力する 8x8 マップ（Fuel Staging Table）を参照して、負荷に応じてプライマリーインジェクターとセカンダリーインジェクターの噴射割合を変更します。

0%=セカンダリインジェクタ無効
100%=プライマリインジェクタが無効

Fuel Staging Table の値は、デューティサイクルまたはパルス幅の分割に直接対応していないことに注意が必要です。この表の値は燃料総量に対し、セカンダリーインジェクターが噴射するパーセンテージを表します。この値によって変わるパルス幅は、プライマリーインジェクタ容量とセカンダリーインジェクタ容量の比率に依ります。
テーブルコントロールのデメリットは、プライマリーインジェクタおよびセカンダリーインジェクタの全噴射を同時に使用することが出来ない事です。テーブルの値は総燃料負荷を分割するので片方のインジェクターの噴射量が増えると、もう片方のインジェクターの噴射量は減少します 。

Automatic Staging

PJSC にはプライマリーインジェクターとセカンダリーインジェクターの合計容量から、自動的に分割割合を計算する Automatic Staging モードがあります。Automatic Staging モードでは、ユーザーはシングルインジェクター使用時と同様に VE テーブルの変更のみで燃調をチューニングする事が出来ます。

このモードでは、Injecgtor Characteristics ダイアログで設定されている『Injector Duty Limit』までプライマリーインジェクターを使用します。ステージングを使用している場合、Injector Duty Limit は 85%以下にする事を推奨します。プライマリーインジェクターが Injector Duty Limit に達すると、それ以上燃料噴射量を増やす場合はセカンダリーインジェクターから供給します。このように、システムが燃料噴射量を各インジェクターに割り当てるのに参照するのは VE テーブルのみとなります。
註）プライマリーインジェクターとセカンダリーインジェクターの無効噴射時間が同じであると仮定されている事に注意して下さい。

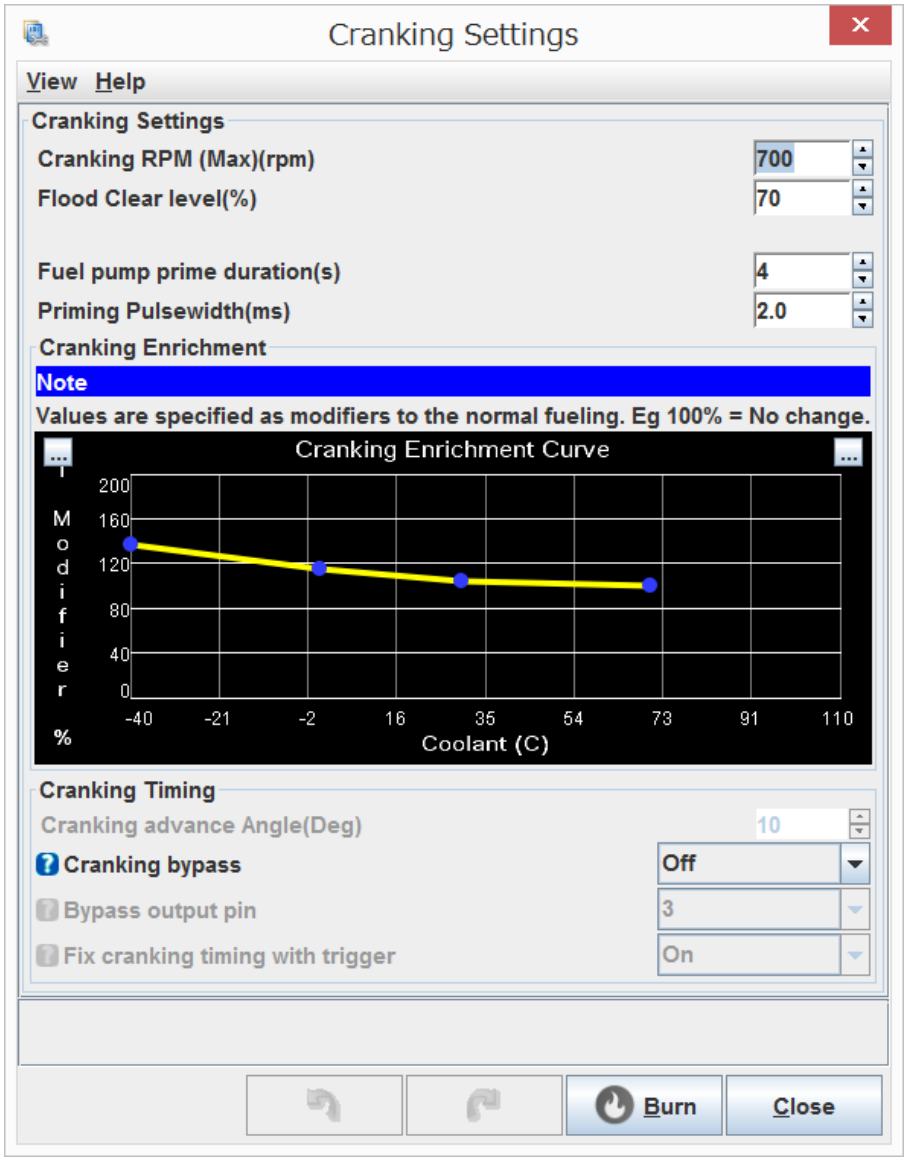
4.12　クランキング/始動補正

4.12.1　概要

通常、エンジン始動時には定常運転時よりも多くの燃料が要求されます。PJSC は指定された条件でエンジン始動時（クランキング）を識別すると、噴射燃料の増減を行う始動補正機能を持っています。
クランキング設定ダイアログ（Cranking Settings Dialog）では PJSC がエンジン始動時か否かを識別する為の条件と、始動時の燃料補正値を入力します。

4.12.2　設定

ツールバーメニューの Starting/Idel ＞ Cranking で以下の様なクランキング設定ダイアログが表示されます。



・**Cranking RPM：**エンジン回転数がこの数値以下の時、PJSC はエンジンは始動時状態であると識別します。エンジン回転数がこれを超えると、エンジン始動は完了して通常運転に切り替わったと識別します。エンジン始動時状態と認識されると、全ての始動時補正が適用されます。
スパイクノイズによる誤検出を避ける為に、実際の始動時クランキング回転数より約 100rpm 程上回る値に設定する事を推奨します。

・**Flood Clear Level：**シリンダーに余分な燃料が入ってプラグが被った状態を解消する為の Flood Clear（被り防止）機能を有効にする条件（スロットル開度%）を入力します。回転数が Cranking RPM 以下で且つ TPS が Flood Clear Level を超えている時、被り防止機能が有効になります。被り防止機能有効時、全ての燃料噴射が停止されプラグが被るリスク無しにクランキングする事が出来ます。

・**Fuel pump prime Duration：**PJSC の電源が OFF から ON になった時、燃料ラインの燃圧を規定の値にする為に数秒間燃料ポンプを駆動する必要があります。そこで電源 ON 時に燃料ポンプは駆動され、この時間が経過すると停止します。もしこの間にエンジンを始動すると、燃料ポンプは停止しないで稼動し続けます。
註）これによる燃料ポンプの駆動は、システムの電源投入時にのみ発生する事に注意して下さい。USB 接続をしている場合、PJSC は車両から 12V 電源が供給されていなくても稼働状態になります。よって USB 接続されたままだと、車両側 12V の電源を OFF から ON に切替えてもこの機能による燃料ポンプの駆動は実行されません。

・**Priming Pulsewidth：**電源投入時、PJSC はここに指定された時間だけ全てのインジェクターのバルブを開きます。このパルスは燃料ラインに入った空気を取り除くためのものであり、始動時燃料増量とは異なる事に注意して下さい。この時間が必要以上に長いと、容易にプラグが被ってしまうので出来るだけ短くしておく必要があります。

・**Cranking enrichment：**エンジン回転数が Cranking RPM 以下の時、この始動補正カーブの値（パーセンテージ）だけ燃料噴射量を増量します。補正量は補正カーブ上のポイントをクリックして移動させるか、始動補正テーブルの値をキーボードから入力する事で任意に変更可能です。

　註）この始動時補正は他の全ての補正（例えば Warmup Enrichment、暖機補正）が加えられた後に、その値に対するパーセンテージだけ加えられます。

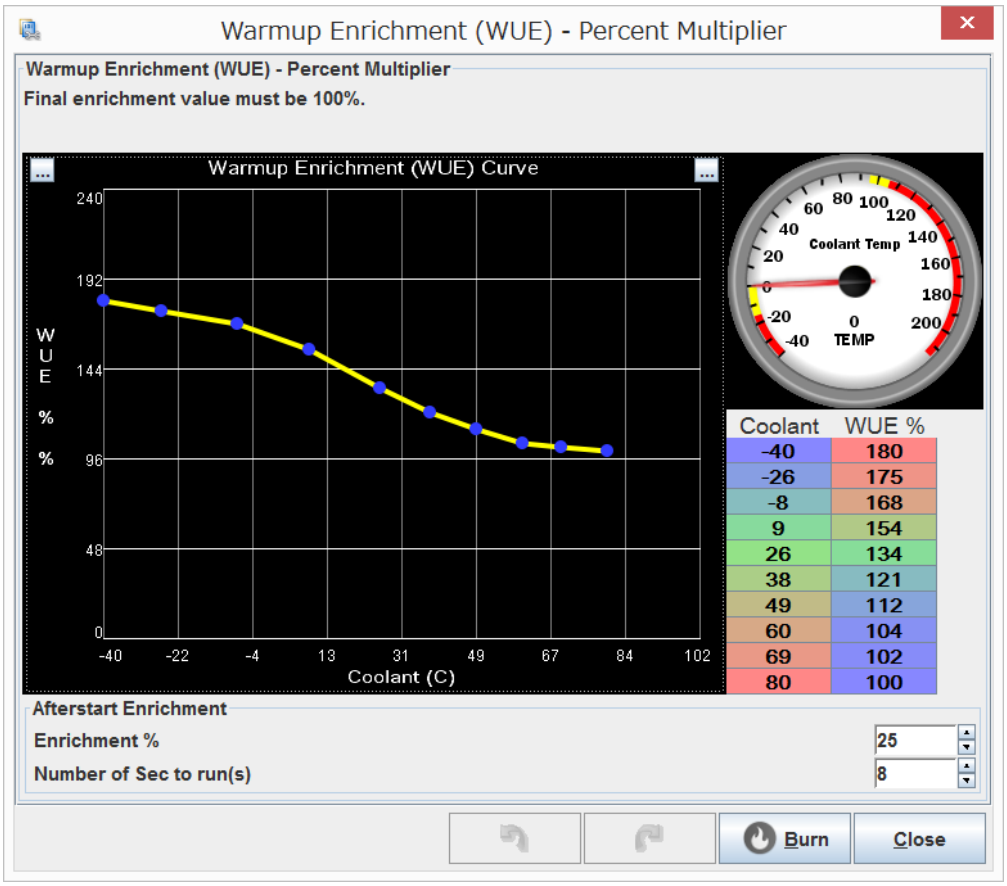
4.13　暖機補正/始動後補正

4.13.1　概要

PJSC はエンジン始動後、水温をモニターして温度に応じて噴射燃料を増減させる暖機補正（WarmUp Enrichment, WUE）機能を持っています。水温に対する燃料補正量を決める暖機補正カーブ（WUE curve）は任意に設定できます。

4.13.2　設定

ツールバーメニューの Starting/Ide＞ Warmup Enrichment で以下の暖機補正ダイアログ（Warmup Enrichment Dialog）が表示されます。



・**暖機補正カーブ：**カーブ上のポイントをクリックして移動させるか、暖機補正テーブルの値をキーボードから入力する事で補正値を任意に設定可能です。暖機補正テーブルの左の列は水温、右の列は補正係数（%）を表しています。補正係数は燃料噴射出力のデューティ比に対して掛けるパーセンテージで、100（%）で補正 0 を意味します。水温センサーを装着しない、或いは暖機補正を無効にする場合は全ての補正係数を 100（%）にして下さい。

・**Afterstart Enrichment：**エンジン始動後の一定時間、無条件で燃料増量を行う始動後補正を設定します。

・**Enrichment %：**始動後補正で増量する燃料のパーセンテージを入力します。この値は係数ではなく、加算される値です。

・**Number of Sec to run(s)：**始動後補正が有効となる時間を入力します。

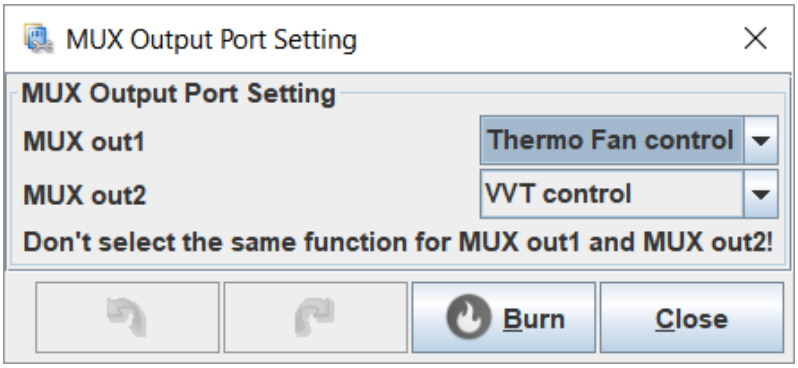
例　Enrichment %＝25（%）、Number of Sec to run(s)＝8（sec）と入力した場合、エンジン始動後 8 秒間、噴射される燃料を 25%増量します。

4.14　MUX output セッティング

4.14.1　概要

PJSC は燃料ポンプ制御やブーストコントロール等、６つの機能に対応した出力ピン－MUX output port を 2 チャンネル持っています。

Accessory メニューの”MUX Output Port Setting”を選択すると以下の様な MUX Output Port Setting ダイアログが表示され、機能を選択する事が出来ます。



・**MUX out1**：MUX out ch1 から出力する信号を以下から選択します。

・**MUX out2**：MUX out ch2 から出力する信号を以下から選択します。

・**Disable/Test Mode**：MUX out の機能を無効にし、信号を出力しません。またハードウェアテストモードで MUX output ピンを使用する場合も、“Disable/Test Mode”を選択します。ハードウェアテストモードについては 5.2 項を参照。

・**IDLE control**：アイドルスピードコントロールバルブを制御する OF/OFF、或いは PWM 信号を出力します。詳細は『4.14 章　アイドリング』を参照して下さい。

・**Thermo Fan control**：冷却ファンを ON/OFF する信号を出力します。詳細は『4.15 章　冷却ファン』を参照して下さい。

・**Fuel pump control**：燃料ポンプを ON/OFF する信号を出力します。詳細は『4.16 章　燃料ポンプ』を参照して下さい。

・**Boost control**：ブーストコントロールバルブを制御する PWM 信号を出力します。詳細は『4.17 章　ブーストコントロール』を参照して下さい。

・**VVT control**：VVT バルブを制御する PWM 信号を出力します。詳細は『4.18 章　VVT コントロール』を参照して下さい。

・**Tach output**：タコメーター用のエンジン回転パルスを出力します。詳細は『4.19 章　タコメーター』を参照して下さい。

4.15　アイドリング

4.15.1　概要

PJSC は ON/OFF または PWM デューティ制御による 2 種類のアイドルスピードコントロールバルブ（Idle Speed Control Valve - ISCV）をサポートしています。ステッピングモータータイプの ISCV には対応していません。

ON/OFF

これは ISCV に流れる電流を ON/OFF で切り替えるシンプルなスイッチ出力です。 このタイプの ISCV は古い車両の多くに採用されていた ON/OFF 高速アイドルバルブ、またはその互換器として設計されたオープン/クローズドソレノイドバルブが該当します。

この様なバルブの例としてバキュームバルブ、ブリザーまたはパージバルブ、燃料バルブ等が挙げられます。

註）ON/OFF バルブは ISCV として空気の流量を増減させる以外にも、暖機や様々な機能のバルブとして使用可能です。例えば、減速停止、空調等の補機の負荷によるアイドル回転数落ち込みの補正、またはターボ・アンチ・ラグ・エア制御などの特定の目的のためのダッシュポット・バルブ等です。 制御情報については、汎用出力の項を参照して下さい。

PWM

PWM バルブの多くはソレノイド ON/OFF バルブと似た様な構造ですが、 PWM アイドルバルブは PWM デューティ比に比例してバルブ開度が大きくなり流量も増えるように設計されています。

オープンループデューティサイクル制御

PJSC はオープンループ制御で PWM 信号を出力し、バルブでの流量調整を可能にしています。 PWM のデューティ比に比例してバルブの開度が変化し、アイドリング回転数が変化します。空気の流量と回転数の関係は気温や気圧等、環境条件によって変化します。

註）バルブによっては通電していない状態でバルブが開いた状態で、PWM 信号のデューティ比が大きくなるに従ってバルブが閉じていく物もあります。バルブを車両に接続する前に動作テストを行って、仕様を確認して下さい。

PWM 設定

TunerStudio の PWM 設定には、PWM アイドル制御、暖機補正の為の温度とデューティ比の関係（PWM Duty Cycle）、クランキング中の PWM デューティ比の設定（PWM Cranking Duty Cycle）が含まれます。

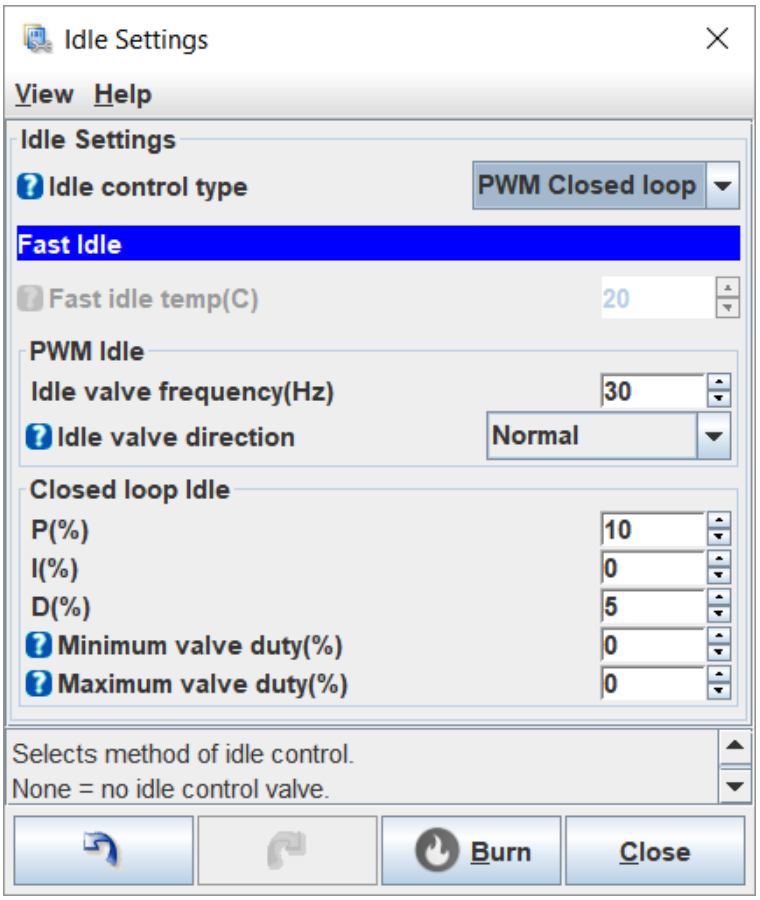
オープンループデューティサイクル制御が有効な場合、PWM Duty Cycle selection カーブ上の水温に合致したデューティサイクルが選択されます。水温対 PWM デューティサイクルカーブは、カーブ上の青い点をクリックして移動するか、手動入力用のテーブルに値を入力する事で変更出来ます。

エンジンによっては、クランキング中は通常より多くの空気を要する物があります。クランキングデューティサイクル（PWM Cranking Duty Cycle）を設定する事で、クランキング中に自動で空気の流量を増やす事が出来ます。エンジンが始動し回転数が設定された最大クランキング回転数を上回ると、クランキングデューティ制御は無効となり通常の設定に切り替わります。

註）PWM 方式の ISCV には 2 線式と 3 線式がありますが、PJSC は 2 線式のみ使用可能です。3 線式には対応していません。

4.15.2　設定

ツールバーメニューの Starting/Ide＞ Idle Control で以下のアイドリング設定ダイアログ（Idle Settings Dialog）が表示されます。



・**Idle control type**：使用する ISCV（Idel Speed Control Valve）の種類に応じた、アイドリング制御方式を選択します。

・**None**：アイドリング制御用を無効にします。

・**ON/Off**：OF/OFF タイプの ISCV を使用する場合、選択して下さい。

・**PWMN Open loop**：PWM 方式の ISCV を使用し、且つオープンループ制御をする場合に選択して下さい。

・**PWMN Closed loop**：PWM 方式の ISCV を使用し、且つクローズドループ制御を行う場合、これを選択して下さい。

・**Fast idel temp(C)**：ON/OFF 方式の ISCV を使用する場合、この値を入力して下さい。アイドリング時に水温がこの温度未満だと Idle control 出力が High になり、水温がこの温度以上になると Idle control 出力が Low になります。

・**Idle valve frequency(Hz)**：PWM 方式の ISCV を使用する場合、この値を入力して下さい。この値が PWM 周波数となります。

・**Idel valve direction**：通常 ISCV は通電するとバルブが開いてインテークマニホールドに空気が流入し、アイドリング回転数が上昇します。この場合は”Normal”を選択して下さい。ISCV によっては通電するとバルブが閉じてアイドリング回転数が低下する物もあります。この場合は、”Reverse”を選択して下さい。

・**Closed loop ldle**：PWM Closed loop cotrol の PID 設定を入力します。

・**P(%)**：Proportional gain

・**I(%)**：Integral gain

・**D(%)**：Differential gain

・**Minimum valve duty(%)**：アイドリング回転数が目標回転数より高い場合、PWM duty 比が下がって ISCV が流量を少なくしていきます。この時の Duty 比の下限を入力します。ISCV の容量とエンジンの組合せににによっては、Duty 比がある値以下になるとアイドリング回転数が変化しない場合があります。この時のアイドリング回転数が目標回転数より高い場合、アイドリング制御が出来なくなってしまいます。アイドリングが変化しなくなる Duty 比を下限値として指定して下さい。

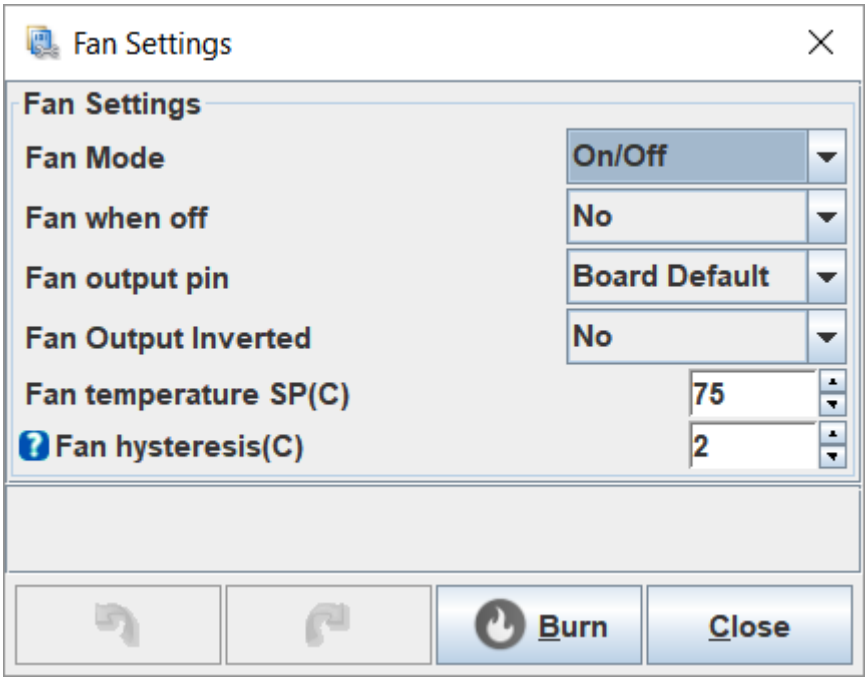
・**Maximum valve duty(%)**：アイドリング回転数が目標回転数より低い場合、PWM duty 比が高くなって ISCV が流量を増やします。この時の Duty 比の上限を入力します。

4.16　冷却ファン

4.16.1　概要

PJSC は水温に応じた冷却ファンの ON/OFF をコントロールする事が出来ます。

MUX output port setting（4.13 参照）で”Thermo Fan control”を選択すると、該当する MUX output ピンから冷却ファン制御信号が出力されます。MUX output は直接冷却ファンを駆動する事は出来ませんので、必ずリレーを介して冷却ファンを駆動して下さい。



・**Fan Mode**：冷却ファン制御信号のモードを選択します。任意の水温で冷却ファンを ON/OFF させたい場合、”On/Off”を選択します。“Off”を選択すると、制御信号は常に OFF になります。

・**Fan when off**：水温が“Fan Temperature SP(C)”で指定した温度以上の時に ON にするか OFF にするかを選択します。“No”を選択した場合、指定温度以上で ON になります。“Yes”を選択すると指定温度以上で OFF になります。

・**Fan output pin**：冷却ファン制御信号を出力する Arduino のピン番号を変更出来ます。通常は“Board Default”を選択して下さい。

・**Fan Output Inverted**：冷却ファン制御信号の信号論理を変える事が出来ます。通常は“No”を選択し、Off で FET が OFF に、On で FET が ONHi になる様にします。“Yes”を選択すると信号論理が反転し、Off で FET が ON に、On で FET が OFF になります。

・**Fan temperature SP(C)**：冷却ファンの ON/OFF を切り替える水温を指定します。

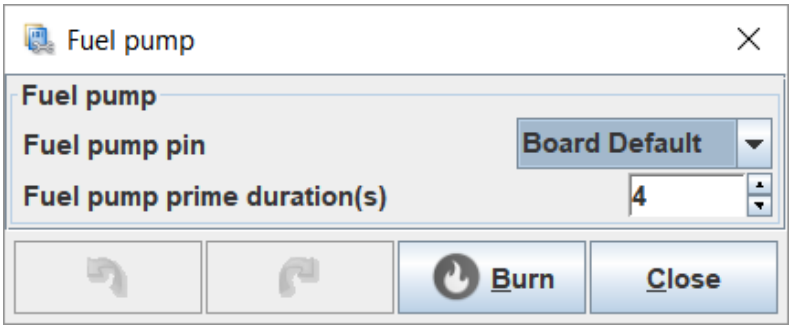
・**Fan hysiteresis(C)**：冷却ファン制御信号のハンチングを防ぐ為、ON/OFF 切り替え温度に対するヒステリシスを設定して下さい。

4.17　燃料ポンプ

4.17.1　概要

PJSC は燃料ポンプの ON/OFF をコントロールする事が出来ます。

MUX output port setting（4.13 参照）で”Fuel Pump control”を選択すると、該当する MUX output ピンから燃料ポンプ制御信号が出力されます。MUX output は直接燃料ポンプを駆動する事は出来ませんので、必ずリレーを介して燃料ポンプを駆動して下さい。



・Fuel pump pin：燃料ポンプ制御信号を出力する Arduino のピン番号を変更出来ます。通常は“Board Default”を選択して下さい。

・Fuel pump prime duration(2)：PJSC は燃料噴射デバイスの初期燃圧を確保する為に、電源 ON 直後に燃料ポンプを駆動します。Fuel pump prime duraion で指定した時間（秒）、燃料ポンプを駆動してから OFF にし、エンジンが始動したら再び燃料ポンプを ON にします。

4.18　ブーストコントロール

4.18.1　概要

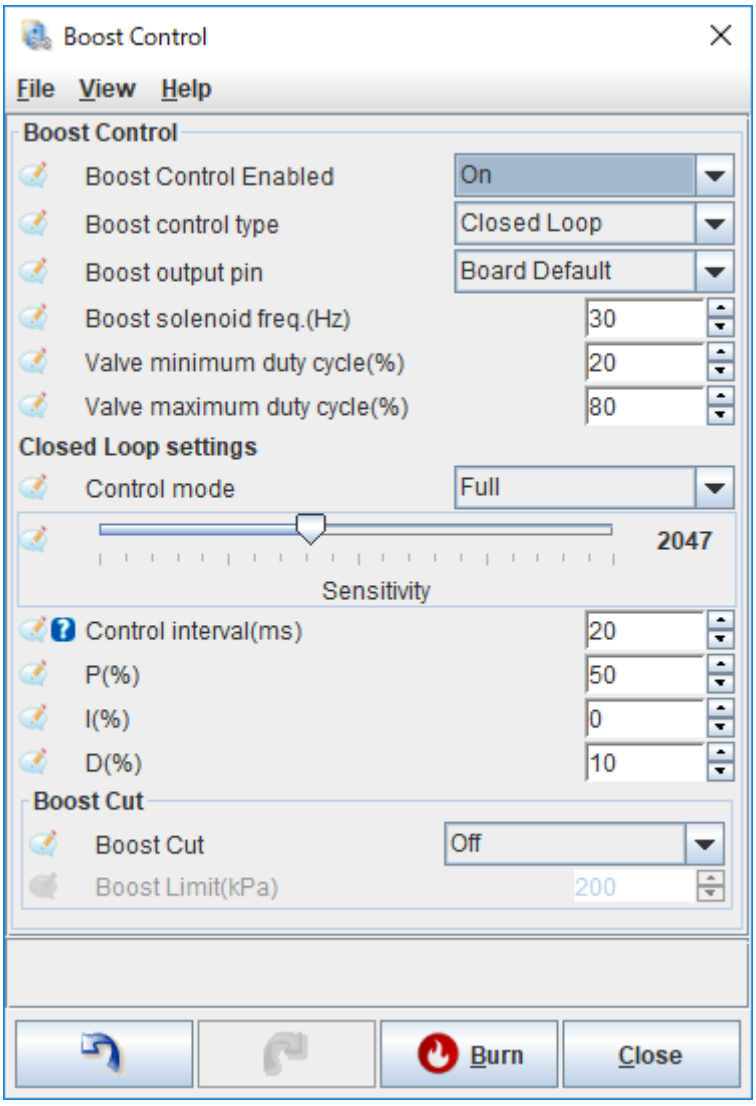
PJSC はクローズドループ制御式のブーストコントロール機能があります。

シングルポートのブーストソレノイドを、15-500Hz の周波数で駆動する事が出来ます。最大電流 3A までのソレノイドをバルブを直接接続出来、ブーストターゲットテーブルに追従するよう PID 制御されます。オーバーブースト制限も可能です。

4.18.2　セッティング

PJSC のブーストコントロールには、シンプルとフルの 2 つの PID 制御モードがあります。 それぞれ以下の様な長所と短所があります。

シンプルモードでは、PID パラメーターが PJSC 計算されます。ユーザーは感度スライダーで制御の感度を指定するだけです。シンプルモードはセットアップが簡単ですが、オーバーブーストを避けるために感度を低く設定する必要があり、遅れが大きくなる可能性があります。



・**Boost Control Enabled**：ブーストコントロール機能の有効。無効を切り替えます。“On”を選択するとブーストコントロールが有効になり、“Off”を選択すると無効になります。

・**Boost control type**：ブーストコントロール方式をオープンループ制御とクローズドループ制御の何れかから選択して下さい。

・**Boost output pin**：ブーストコントロール信号を出力する Arduino のピン番号を変更出来ます。通常は“Board Default”を選択して下さい。

・**Boost solenoid freq.(Hz)**：PJSC はブーストコントロールソレノイドを PWM で制御します。ソレノイドバルブに適した PWM 信号の周波数を入力して下さい。

・**Valve minimum duty cycle(%)**：PWM のデューティ比（%）の最小値を入力します。

・**Valve maximum duty cycle(%)**：PWM のデューティ比（%）の最大値を入力します。

・**Closed Loop settings**：クローズドループ制御を選択した場合、制御のパラメーターを指定します。

・**Control mode**：“Full”と“Simple”の何れかの制御モードを選択します。Simple を選択した場合、Control mode 選択メニューの下に表示されている PID 感度スライダーで感度を指定します。感度スライダーは 0 が最も感度が低く、2047 が最も感度が高くなります。最初は感度 500 程度から始め徐々に上げていき、ブーストのオーバーシュートが起き始めた時の感度より少し低い感度を選択して下さい。

・**Control interval(ms)**：クローズドループ制御のサンプリングインターバル（ms）を指定します。通常は PWM 周期の 50%～100%の範囲で指定します。

・**P(%)**：Proportional gain

・**I(%)**：Integral gain

・**D(%)**：Differential gain

・**Boost Cut**：ブーストカット機能の有効、無効を選択します。“Off”で無効になり、それ以外で有効になります。

・**Boost Limit(kPa)**：ブーストカットが働く圧力を入力して下さい。入力した値以上の圧力になると、ブーストカットが働きます。

4.18.3　ブーストターゲットテーブル

ブーストコントロールを有効にした場合、ブーストターゲットテーブルにターゲットとなるブースト値（kPa）或いは PWM のデューティ比を入力して下さい。

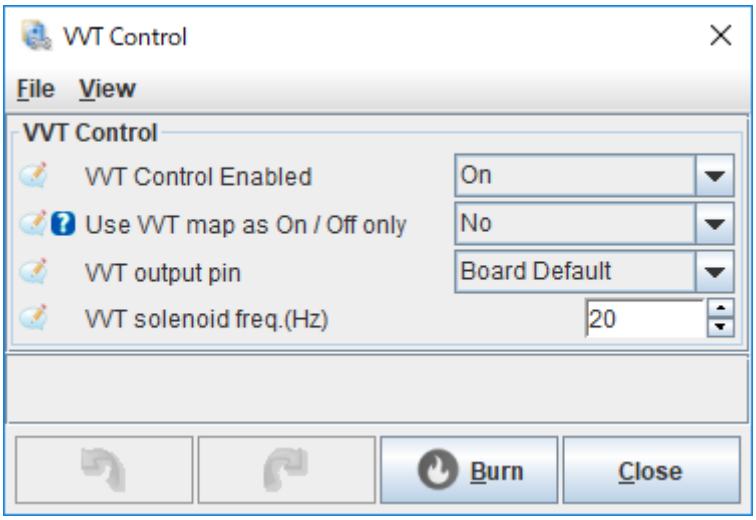
Boost control type で Open Loop を選択した場合、ブーストターゲットテーブルに入力された値は PWM のデューティ比になります。Closed Loop を選択した場合は、ターゲットブースト値（kPa）となります。

4.19　VVT コントロール

4.19.1　概要

PJSC はアクセル開度とエンジン回転数に応じて VVT をコントロールする事が出来ます。

MUX output port setting（4.13 参照）で”VVT control”を選択すると、該当する MUX output ピンから VVT 制御信号が出力されます。



・**VVT Control Enabled**：“On”を選択すると VVT 制御機能が有効に、“Off”を選択すると無効になります。

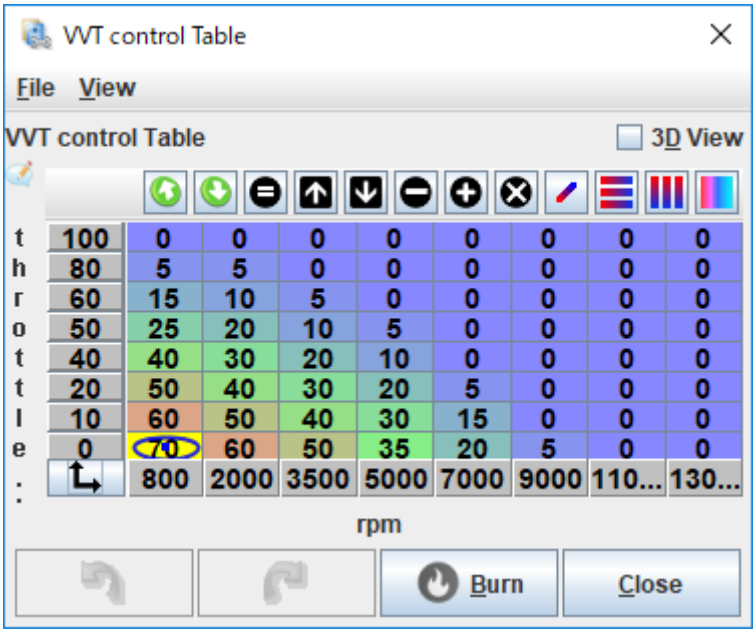
・**Use VVT map as On/Off only**：VVT の制御に On/Off バルブを使用する場合、“Yes”を選択します。PWM 制御を行う場合は“No”を選択して下さい。VVT 制御に On/Off バルブを使用する場合、後述する VVY control Table が 100 の時だけ ON に、100 未満の場合は全て OFF になります。

・**VVT output pin**：VVT 制御信号を出力する Arduino のピン番号を変更出来ます。通常は“Board Default”を選択して下さい。

・**VVT solenoid freq.(Hz)**：VVT 制御に PWM を選択した場合、PWM 周波数を指定して下さい。通常は 10～20Hz 程度ですが、使用するソレノイドバルブによって適切な周波数が異なります。バルブの仕様を確認して下さい。

4.19.2　セッティング

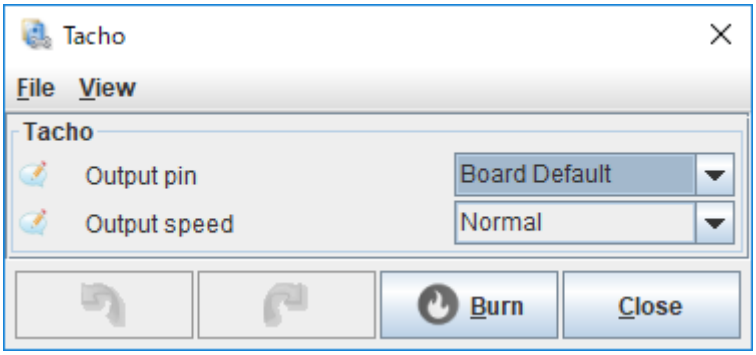
VVT 制御のセッティングは、VVT control Table の値を調整して行います。制御方式に PWM を選択した場合、テーブルには PWM のデューティ比を入力して下さい。“Use VVT map as On/Off only”に“Yes”を選択した場合、VVT 制御出力信号は ON/OFF となります。テーブルの値が 100 の時のみ ON となり、100 未満の場合は OFF になります。



4.20　タコメーター信号

4.20.1　概要

MUX output port setting（4.13 参照）で”Tach output”を選択すると、該当する MUX output ピンからタコメーター用パルス信号が出力されます。



・**Output pin**：タコメーター信号を出力する Arduino のピン番号を変更出来ます。通常は“Board Default”を選択して下さい。

・**Output speed**：タコメーター信号パルスの出力間隔を設定します。“Normal”を選択すると 1 回転に 1 パルス出力し、“Half”を選択すると 2 回転に 1 パルス出力します。

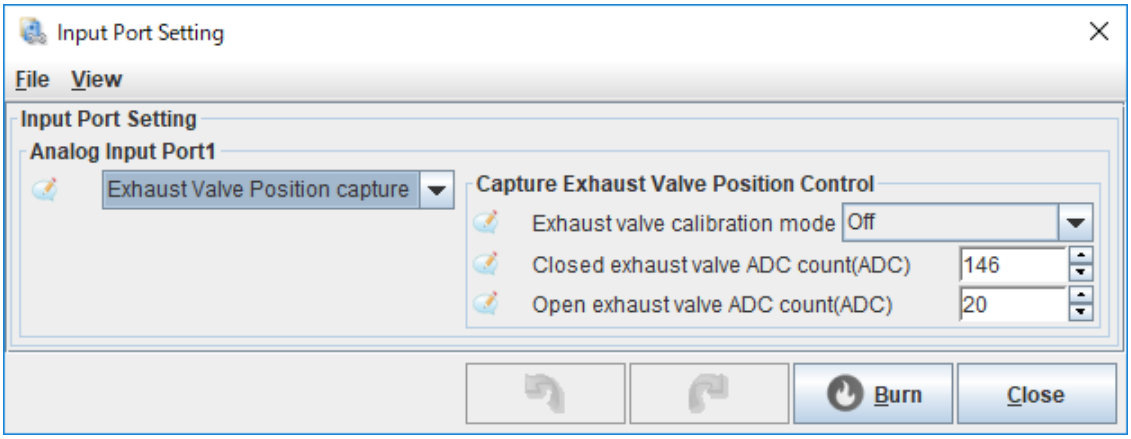
4.21　排気バルブポジション検知

4.21.1　概要

PJSC では、オートバイの２サイクルエンジンで多く採用されている排気バルブの開度信号を取り込み、ログに残す事が可能です。取り込み可能な信号は、TPS と同様なポテンショナーで開度を検知する排気バルブの信号です。

4.21.2　セッティング

ツールバーメニューの Accessories＞Input Port Setting を選択すると、以下の様な Input Port Setting ダイアログが表示されます。



・**Analog Input Port1：プルダウンメニューから**“Exhaust Valve Position capture”を選択すると、排気バルブポジション検知機能が有効になります。

・**Exhaust valve calibration mode：**“On”を選択すると、排気バルブポジション検知機能のキャリブレーションモードが有効になります。この機能が有効な状態で排気バルブを動かした場合、Exhaust valve position ADC の値が Closed exhaust valve ADC count より大きくなると、Closed exhaust valve ADC count の値を取り込んだ ADC の値で更新します。また ADC の値が Open exhaust valve ADC count より小さくなると、Open exhaust valve ADC count の値を取り込んだ ADC の値で更新します。つまりこの機能が有効な状態で排気バルブを全閉、全開の位置に動かすと、自動的に ADC の値の最小値と最大値を Open exhaust valve ADC count、Closed exhaust valve ADC count に設定します。

・**Closed exhaust valve ADC count：**排気バルブが 0%（全閉）になった時の ADC 値を入力します。

・**Open exhaust valve ADC count：**排気バルブ開度が 100%（全開）になった時の ADC の値を入力します。この値と、Closed exhause vlve position ADC count、入力された ADC 値から排気バルブ開度を計算してパーセンテージで表示します。

第５章　ハードウェアテスト

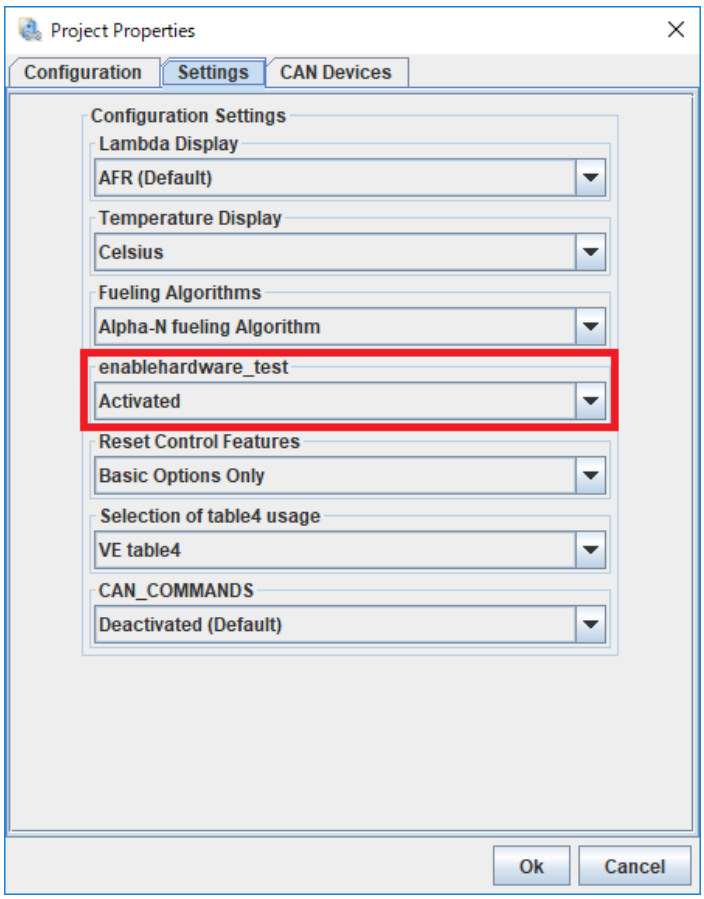
PJSC にはエンジンを始動せずに噴射デバイスを駆動する、ハードウェアテストモードがあります。この機能を使う事で、エンジンを始動する前に PJSC が正しく動作するか、車両との接続が正しいか確認する事が出来ます。

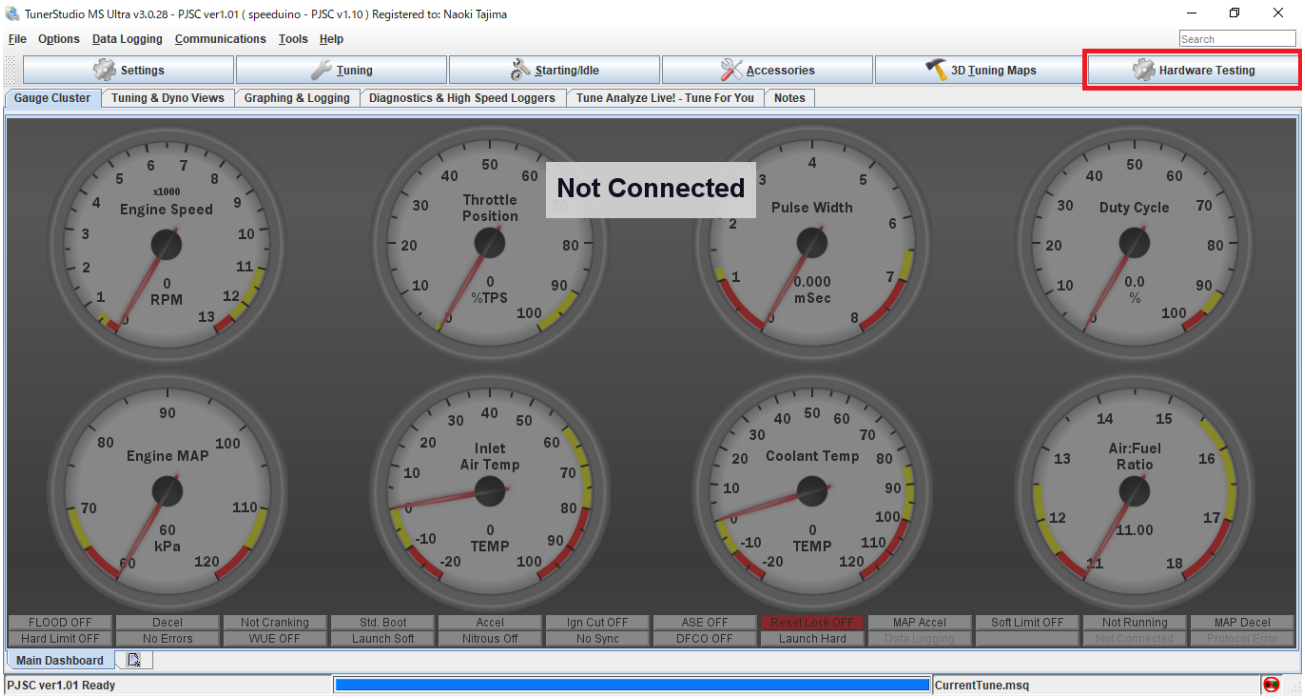
この章では、ハードウェアテストモードの使い方を解説します。

5.1　　プロジェクトプロパティ設定

ハードウェアテスト機能を有効にするには、プロジェクトプロパティで設定を行う必要があります。以下の手順でハードウェアテスト機能を有効にして下さい。

(1) Tuner Studio のメインメニューから File-> Vehicle Projects-> Project Properties を選択して、プロジェクトプロパティ設定ダイアログを開いて下さい。
(2) 次に“Settings”タブを選択して以下の画面を表示させます。“enablehardware_test”のプルダウンメニューから、“Activated”を選択します。
(3) “OK”ボタンをクリックします。
(4) Tuner Studio メイン画面で、“Hardware Testing”メニューが表示されていれば設定完了です。


Project Properties
Configuration
Settings
CAN Devices
Configuration Settings
Lambda Display
AFR (Default)
Temperature Display
Celsius
Fueling Algorithms
Alpha-N fueling Algorithm
enablehardware_test
Activated
Reset Control Features
Basic Options Only
Selection of table4 usage
VE table4
CAN_COMMANDS
Deactivated (Default)
Ok
Cancel

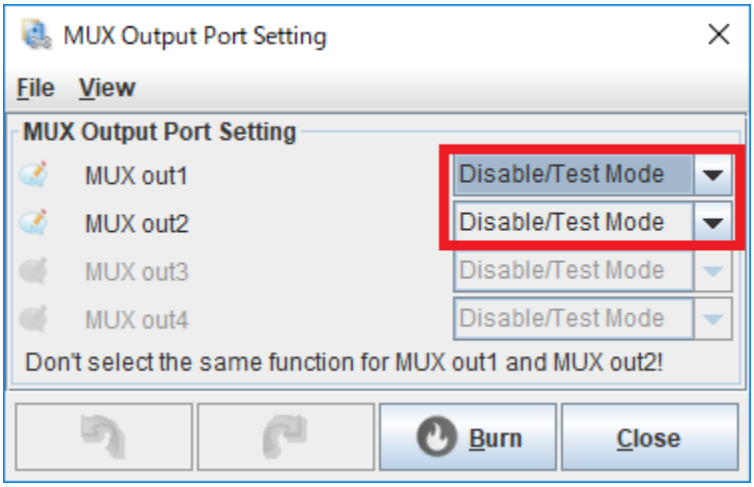

TunerStudio MS Ultra v3.0.28 - PJSC ver1.01 (speeduino - PJSC v1.10) Registered to: Naoki Tajima
File Options Data Logging Communications Tools Help
Search
Settings
Tuning
Starting/Idle
Accessories
3D Tuning Maps
Hardware Testing
Gauge Cluster
Tuning & Dyno Views
Graphing & Logging
Diagnostics & High Speed Loggers
Tune Analyze Live! - Tune For You
Notes
Engine Speed
RPM
Throttle Position
%TPS
Not Connected
Pulse Width
mSec
Duty Cycle
%
Engine MAP
kPa
Inlet Air Temp
TEMP
Coolant Temp
TEMP
Air:Fuel Ratio
11.00
FLOOD OFF
Decel
Not Cranking
Std. Boot
Accel
Ign Cut OFF
ASE OFF
Reset Lock OFF
MAP Accel
Soft Limit OFF
Not Running
MAP Decel
Hard Limit OFF
No Errors
WUE OFF
Launch Soft
Nitrous Off
No Sync
DFCO OFF
Launch Hard
Data Logging
Not Connected
Protocol Error
Main Dashboard
PJSC ver1.01 Ready
CurrentTune.msq

5.2　　ハードウェアテスト

5.2.1　　PWM 出力テスト

インジェクター出力ピンから PWM 信号を出力するテストを行う場合、Hardware Testing のプルダウンメニューから“Output Test Mode - PWM”を選択します。

MUX output ピンからテスト信号を出力する場合、予め MUX output セッティングダイアログで“Disable/Test Mode”を選択します。（詳細は 4.14 項を参照）



以下の様なパルス出力テストダイアログが表示されるので、下記(1)～(5)の手順で PWM 信号を出力します。

PWM 出力テスト手順：

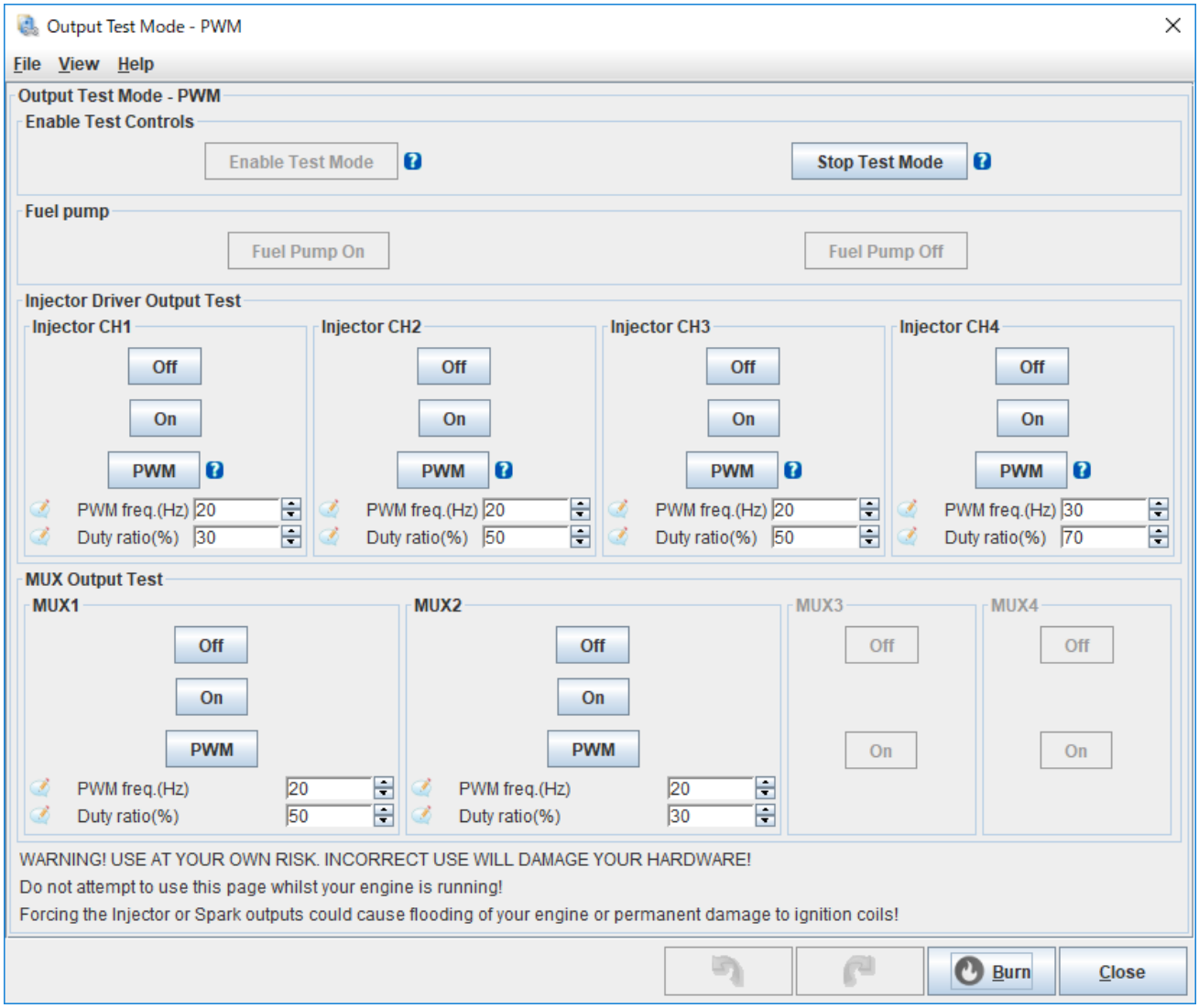
(1) “Enable Test Mode”をクリックする。

(2) 必要なら“Fuel Pump On”をクリックして燃料ポンプを駆動する。

(3) “PWM freq(Hz)”で出力したい周波数を入力する。

(4) “Duty ratio(%)”で出力したいデューティ比を入力する。

(5) “PWM”ボタンをクリックすると、対応するインジェクター出力から PWM 信号が出力されます。



・**Enable Test Mode**：このボタンをクリックすると、テストモードが有効になります。

・**Stop Test Mode**：このボタンをクリックすると、テストモードが無効になります。

・**Fuel Pump On**：このボタンをクリックすると、燃料ポンプが ON になります。

・**Fuel Pump Off**：このボタンをクリックすると、燃料ポンプが OFF になります。

・**Off**：このボタンをクリックすると、対応するインジェクター出力が OFF になります。

・**On**：このボタンをクリックすると、対応するインジェクター出力が ON になります。

・**PWM**：このボタンをクリックすると、対応するインジェクター出力から PWM 信号が出力されます。

・**PWM freq(Hz)**：対応するインジェクター出力から出力される PWM 信号の周波数を入力します。

・**Duty ratio(%)**：対応するインジェクター出力から出力される PWM 信号のデューティ比を入力します。

5.2.2　パルス出力テスト

インジェクター出力ピンから回転数同期のインジェクターと同じパルス出力をするテストを行う場合、Hardware Testing のプルダウンメニューから“Output Test Mode - Pulse”を選択します。

以下の様なパルス出力テストダイアログが表示されるので、下記(1)～(5)の手順で PWM 信号を出力します。

パルス出力テスト手順：

(1) “Enable Test Mode”をクリックする。

(2) 必要なら“Fuel Pump On”をクリックして燃料ポンプを駆動する。

(3) “Output Interval(ms)”で出力したいパルスの間隔を入力する。

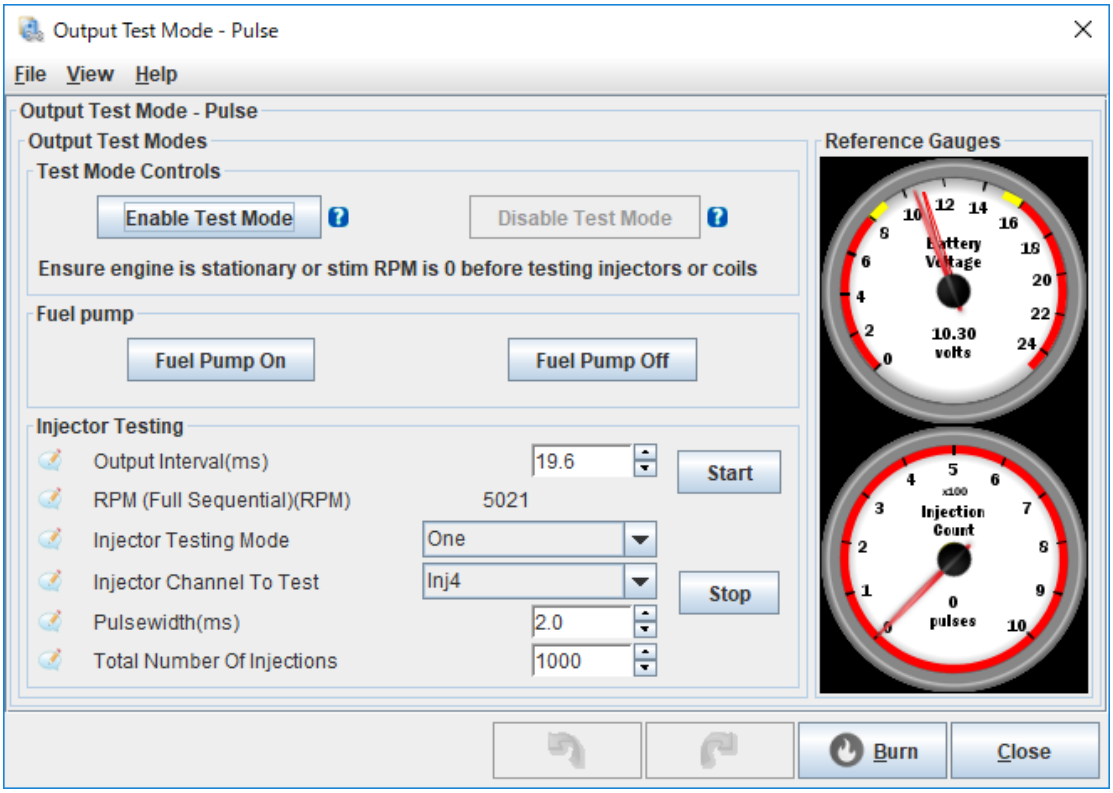
(4) “Injector Testing Mode”でパルス信号を出力したいチャンネル数を選択する。

(5) “Injector Channel To Test”でパルス信号を出力したいチャンネルを選択する。

(6) “Pulsewidth(ms)”で出力したいパルスの幅を入力する。

(7) “Total Number Of Injections”で出力する信号のパルス数を入力します。

(8) “Start”ボタンをクリックすると、パルス信号が出力されます。



・**Enable Test Mode：**このボタンをクリックすると、テストモードが有効になります。

・**Stop Test Mode：**このボタンをクリックすると、テストモードが無効になります。

・**Fuel Pump On：**このボタンをクリックすると、燃料ポンプが ON になります。

・**Fuel Pump Off：**このボタンをクリックすると、燃料ポンプが OFF になります。

・**Output Interval(ms)：**パルスとパルスの間隔を入力します。

・**Injector Testing Mode：**パルス信号を出力するインジェクター出力チャンネルう数を選択します。

・**Off：**これを選択すると、パルス信号は出力されません。

・**One：**“Injector Channel To Test”で選択したインジェクター出力チャンネルから、パルス信号が出力されます

・**All：**全てのインジェクター出力チャンネルから、パルス信号が出力されます。

・**Injector Channel To Test：**パルス信号を出力したいチャンネルを選択します。

・**Pulsewidth(ms)：**出力する信号のパルス幅を入力します。

・**Total Number Of Injections：**出力する信号のパルス数を入力します。

・**Start：**このボタンをクリックすると、パルス信号が出力されます。

・**Stop：**このボタンをクリックすると、パルス信号出力が停止します。