PIC16F1825による正弦波発生器，PWM信号発生器（第2版）

－MCCによる周辺モジュール設定関数の自動生成－

本稿掲載のWebページ

https://mybook-pub-site.sakura.ne.jp/Power_Electronics_Note/

令和6年2月

古橋　武

目次

第１章

はじめに

本稿は,「パワーエレクトロニクスノート II ―製作演習付き講義の実践記録―」で使用している，PIC マイコン PIC16F1825 による正弦波発生器，PWM 信号発生器のプログラムとその動作について解説します.

本稿の初版を 2019 年 12 月に掲載してから 4 年が経ちました．その間，多くの方に閲覧／ダウンロードしていただきました．2024 年 2 月の現時点でも，訪問者が絶えません.

第 2 版では，MCC(MPLAB® Code Cofigurator) を使用します．MCC は，MPLAB® X IDE に組み込まれた，グラフィカルなプログラム開発環境です．MCC により，PIC マイコン内の周辺モジュール設定関数 (C のコード) を，ほぼボタンクリックのみで生成できます．これまで，周辺モジュール設定関数は，データシートをくまなく読み込まないと書けませんでした．MCC は，周辺モジュール設定関数を自動生成してくれる，画期的なユーザインタフェースです．改訂プログラムを本稿と同じ Web ページ

パワーエレクトロニクスノート

に掲載します.

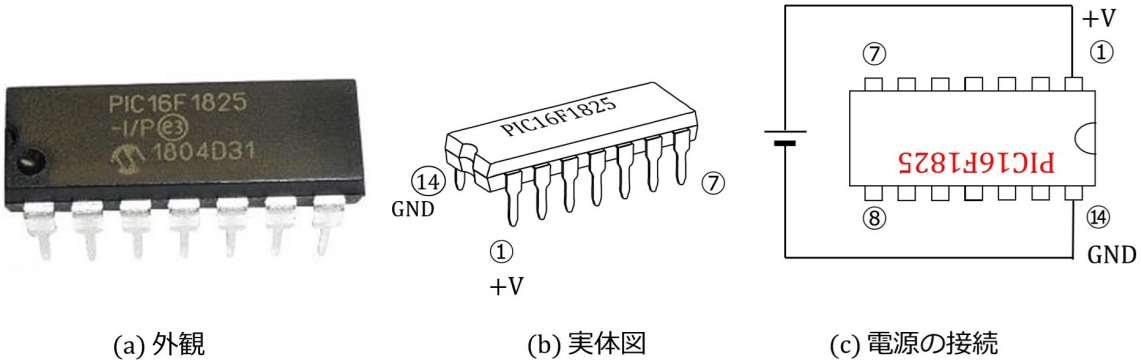
初版の特徴は，周辺モジュール設定関数とそのためのヘッダファイルを提供している点にあります．MCC により，その特徴の効用はほとんど無くなりましたが．初版も掲載を続けることにします．それは，MCC がデータシートの必要性を無くすものではないからです．周辺モジュールの性能をフルに引き出すには，データシートに記載された内容を理解する必要があります．拙稿初版は，データシートと周辺モジュール設定関数の関係を理解する助けになります．ご活用ください.

第 2 章

PIC16F1825 の正弦波生成モードと PWM 信号生成モードおよび動作例

2.1　正弦波生成モード

2.1.1　部品



(a) 外観　　(b) 実体図　　(c) 電源の接続

図　2.1: PIC16F1825

図 2.1 は PIC16F1825 の外観，実体図および電源接続の様子です．このマイコンは Microchip 社の製品です．DIP(Dual In-line Package) と呼ばれるパッケージのものを選びました．電極（ピン）が 2 列に並んでいて，プリント基板，ブレッドボードに差し込めるタイプです．PIC16F1825 は 14 ピンを持ち，これらのピン 7 個ずつが 2 列に配置されています．ピン番号は，同図 (b) の実体図の通り，パッケージに設けられた凹みを左に見て，その左下から反時計回りに①→②→ … →⑭と付けられています． このマイコンの場合，電源は同図 (c) のように，①番ピンに電源の＋側，⑭番ピンに電源の－側（GND : Ground）をつなぎます．

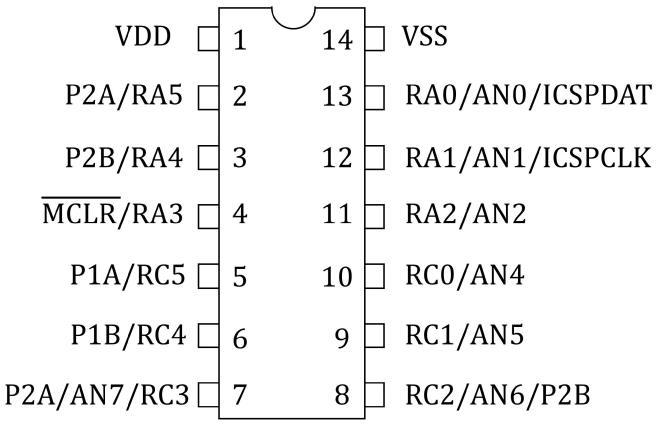


図 2.2: PIC16F1825 のピン配置

図 2.2 は PIC16F1825 のピン配置を示します．②番ピンから⑬番ピンまでは複数の機能が割り当てられていて，プログラムにより選択できます．RAx, RCx は I/O(Input/Output) ポートです．ANx はアナログ入力ポートであり，A-D 変換モジュールをつなげられます．PxA,PxB は PWM 信号の出力ピンです．MCLR は，Master Clear の略で，$\overline{\text{MCLR}}$ とオーバーラインが付いているので，負論理入力です．このピン（4 番ピン）は通常は 5 [V] を入力しておき，強制的にマイコンをリセット（初期化）したい場合に 0 [V] を入力します．その後 5 [V] を入力すると，マイコンは main() 関数の先頭にもどって再起動します．ICSPDAT と ICSPCLK は ICSP$^{\text{TM}}$ (In-Circuit Serial Programming) 用のピンです．MPLAB® Snap などの Debugger/Programmer を接続して，マイコンへのプログラム書き込み，デバッグを行うことができます．

図 2.3 は可変抵抗器の外観と内部構造および記号です．可変抵抗器は抵抗値を変化させられる抵抗器です．写真のタイプは a, b, c の 3 電極を持ちます．抵抗器上面の灰色部分をネジ回しで回転させると，b 電極の先が連動して a-c 間の抵抗体表面を摺動します．これにより，a-b 間，b-c 間の抵抗値を変化させられます．a-c 間の抵抗値は固定です．図 (b) 最下図のように，灰色部分を時計方向いっぱいに回すと，a-b 間の抵抗 R_{a-b} 最小，b-c 間の抵抗 R_{b-c} 最大となります．逆に図 (b) 最上図のように，反時計方向いっぱいに回すと，R_{a-b} 最大，R_{b-c} 最小となります．抵抗器の前面に a-c 間の抵抗値が印字されています．写真の例は 202 です．この数値の意味は

$$\begin{aligned} 202 &= 20 \times 10^2[\Omega] \\ &= 2[\text{k}\Omega] \end{aligned} \tag{2.1}$$

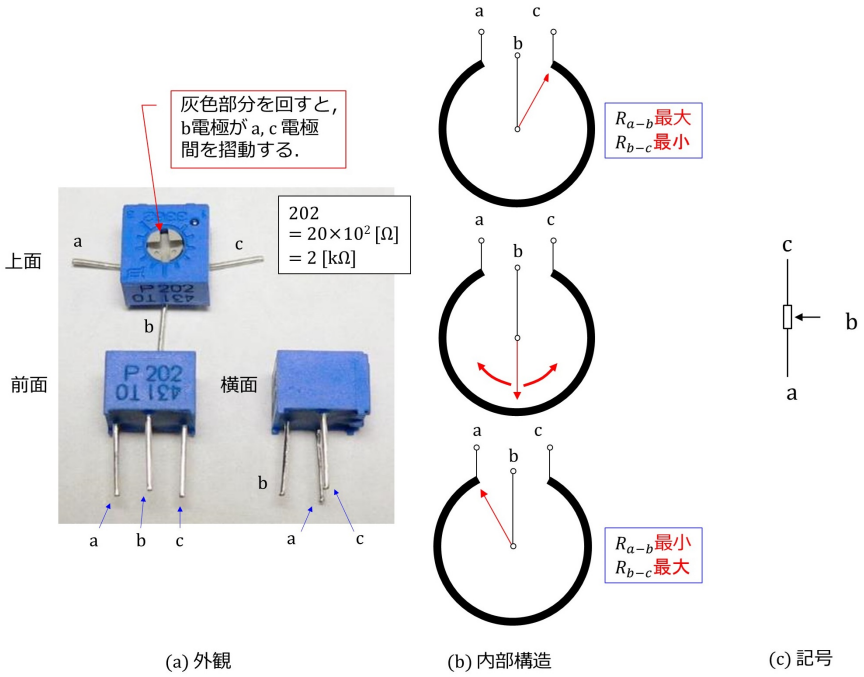


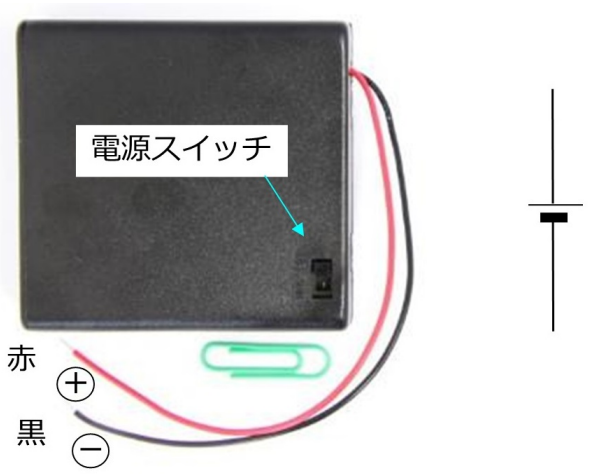
図　2.3: 可変抵抗器

です．したがって，R_{a-b}, R_{b-c} の変化範囲は

$$\begin{aligned} R_{a-b} &= 0 \sim 2[\mathrm{k\Omega}] \\ R_{b-c} &= 2[\mathrm{k\Omega}] - R_{a-b} \\ &= 2 \sim 0[\mathrm{k\Omega}] \end{aligned} \tag{2.2}$$

です．

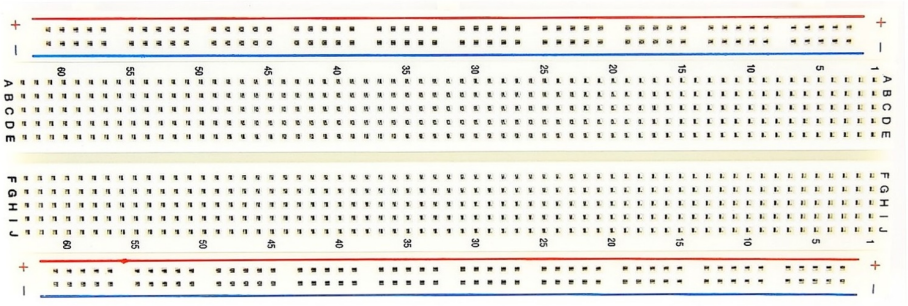
写真の可変抵抗器は半固定形です．ネジ回しでなく，指でつまんで回せるタイプの可変抵抗器と区別するために，半固定形と呼ばれます．また，可変抵抗器はボリュームとも呼ばれます．写真のタイプのボリュームは半固定ボリュームとも呼ばれます．



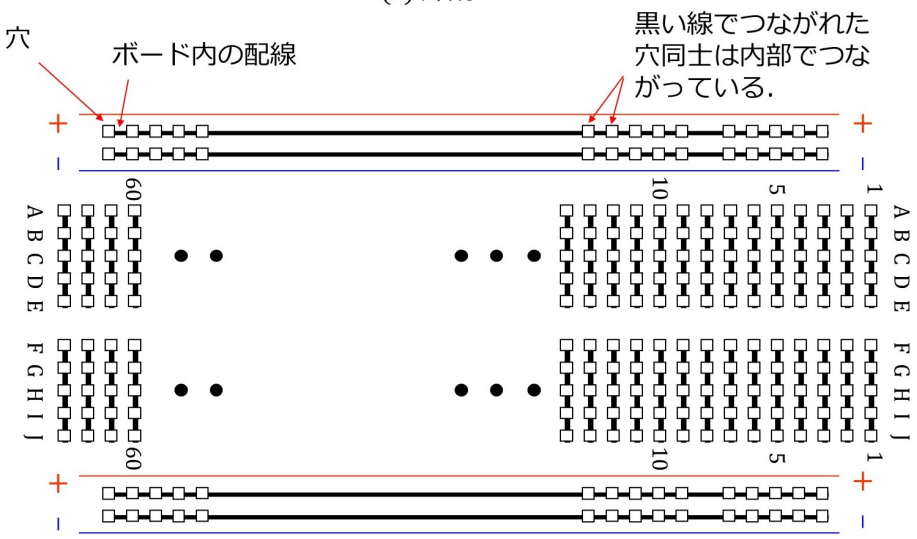
(a) 外観 (b) 記号

図 2.4: 電池ボックス

図 2.4 は電池ボックスの外観と記号です．単 3 乾電池 4 本を直列接続で収納できるタイプです．出力電圧は乾電池の場合 6 [V]，充電池の場合 5 [V] です．赤い電線が電池の＋側，黒い電線が－側につながっています．電源スイッチが付いているので，容易に電源のオン／オフ切替ができます．



(a) 外観



(b) ブレッドボードの穴のつながりの様子

図　2.5: ブレッドボード

図 2.5 はブレッドボードの外観とボード内部での穴のつながりの様子です．□がジャンパ線を差し込める穴です．黒い線によりブレッドボード内部での穴同士のつながりを示します．最上段の 2 行と最下段の 2 行は，各行内の 50 個の穴が内部でつながれています．これら上下各 2 行の穴に挟まれて 63 列の穴があります．各列には 10 個の穴があり，A～E，F～J の 5 個ずつが内部でつながれています．

2.1.2 正弦波生成の実験回路

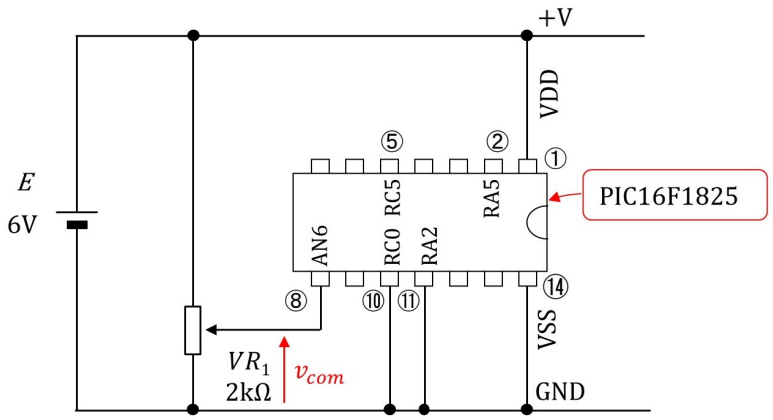


図 2.6: 正弦波生成実験の回路図

図 2.6 は正弦波生成実験の回路図です．マイコンには第 4 章のプログラムが書き込まれている前提です．RC0（10 番ピン），RA2（11 番ピン）ポートを GND につないで電源を投入すると正弦波生成プログラムが走ります．正弦波電圧は RA5（2 番ピン），RC5（5 番ピン）に出力されます．AN6（8 番ピン）はアナログ入力ポートで，マイコン内の A-D 変換モジュールにつながれています．可変抵抗器の b 電極を動かすことで，AN6 への入力電圧 v_{com} を変えられます．この電圧に応じて，マイコン内のプログラムが RA5，RC5 の出力正弦波電圧の周波数を変化させます．

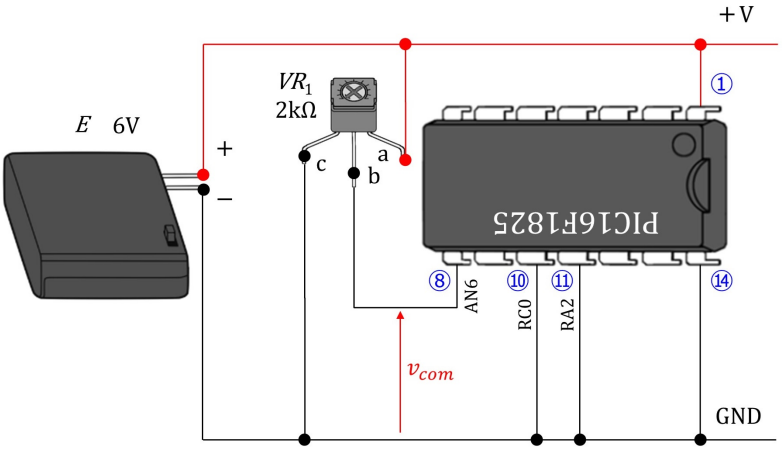


図 2.7: 正弦波生成実験回路の実体配線図

図 2.7 は正弦波生成実験回路の実体配線図です．回路図と実際の部品との関係を把握し

易くするために実体図を用いて配線の様子を描いてあります．

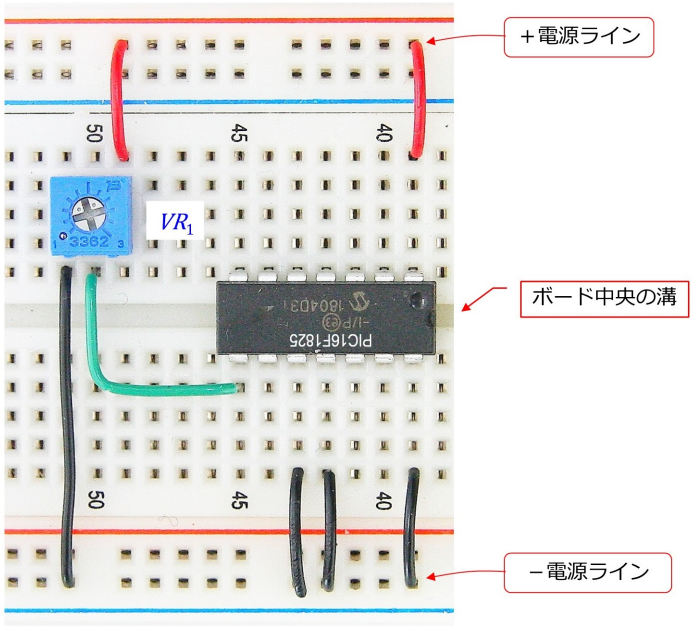
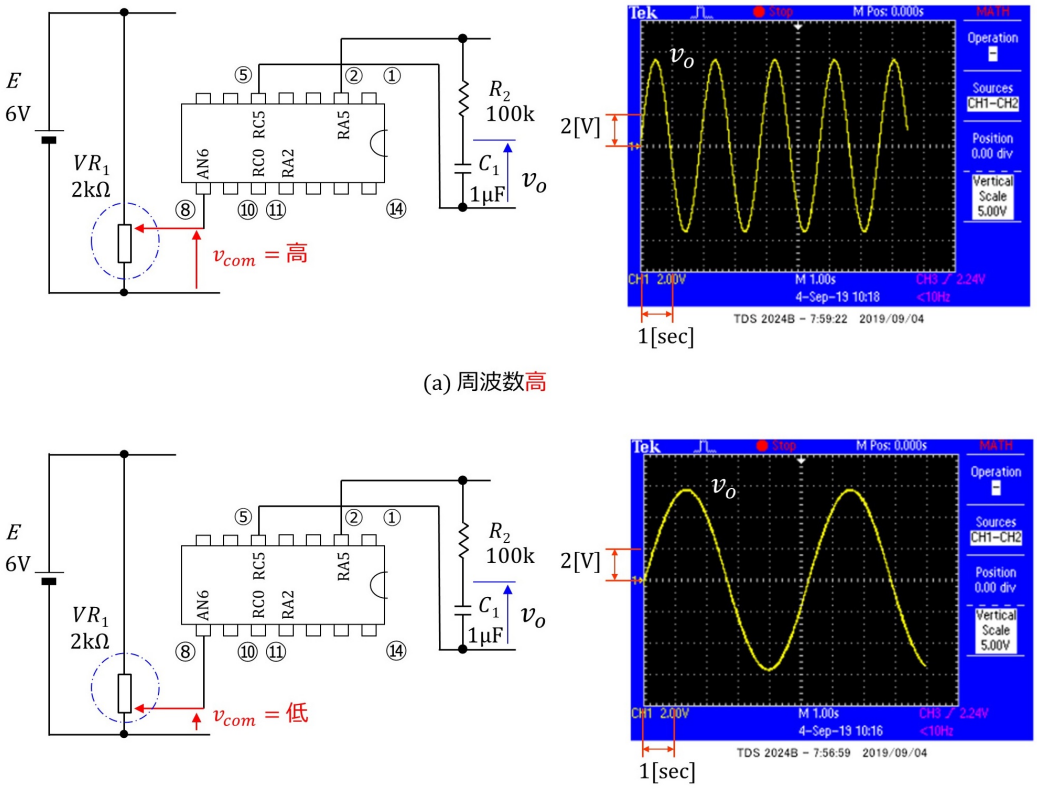


図　2.8: 正弦波生成実験回路の配線例

図 2.8 は正弦波生成実験回路の配線例です．図 2.7 の実体配線をブレッドボード上で実現しています．一番上の穴行を＋電源ライン，一番下の穴行を－電源ラインとしています．マイコンはブレッドボードの中央に設けられている溝を挟んで挿入します．ブレッドボードの各穴列は，図 2.5 に示すように，5 個ずつつながっています．そして，中央の溝で切れています．マイコンはこの溝を挟んで挿入します．



(a) 周波数高

(b) 周波数低

図 2.9: 正弦波生成実験回路の動作波形例

図 2.9 は正弦波生成実験回路の動作波形例です．オシロスコープ画面のスナップショットは横軸が 1 [sec/div]（一目盛り当たりの値）であり，縦軸が 2 [V/div] です．同図 (a) は出力正弦波の周波数が高い（約 0.5 [Hz]）場合，(b) は低い（約 0.2 [Hz]）場合です．v_{com} が高いときに周波数が高くなるようにプログラムしてあります．

2.1.3 PWM 制御法の基本概念

なお，このマイコンはアナログ電圧を出力する機能を持っていません．RA5，RC5 に出力できる電圧は VDD（電源電圧）と 0 [V] の 2 値です．この 2 値を使って，図 2.9 の波形例のような正弦波電圧を生成する手法が PWM(Pulse Width Modulation) 制御法です．

図 2.10 は PWM 制御法の基本概念です．RA5，RC5 ポートの出力電圧をそれぞれ $v_{\mathrm{RA5}}, v_{\mathrm{RC5}}$ とします．同図右は v_{RA5} のイメージ図です．T_{PWM} は PWM 周期と呼ばれます．T_{high} は $v_{\mathrm{RA5}} = VDD$ の期間です．v_{RA5} の平均値を $\overline{v}_{\mathrm{RA5}}$ とすると

$$\overline{v}_{\mathrm{RA5}} = \frac{T_{high}}{T_{PWM}} VDD \tag{2.3}$$

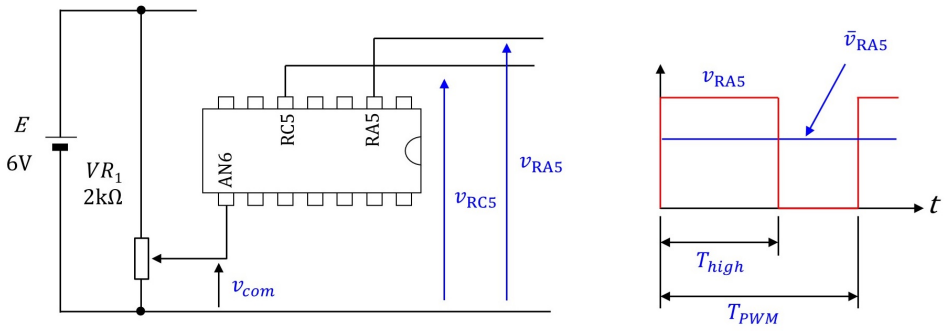


図　2.10: PWM 制御法の基本概念

です．T_{high} 期間の v_{RA5} の波形はパルスと呼ばれます．T_{high} はパルス幅です．PWM 制御法はパルス幅を制御する方法です．正弦波の周期を $T_{\sin}$ とします．PWM 周期 T_{PWM} を正弦波の周期 $T_{\sin}$ に対して十分に短くします．すなわち，

$$T_{PWM} << T_{\sin} \tag{2.4}$$

とします．そして，v_{RA5} の T_{PWM} ごとの平均値 $\bar{v}_{\mathrm{RA5}}$ が近似的に正弦波となるように T_{high} を制御します．

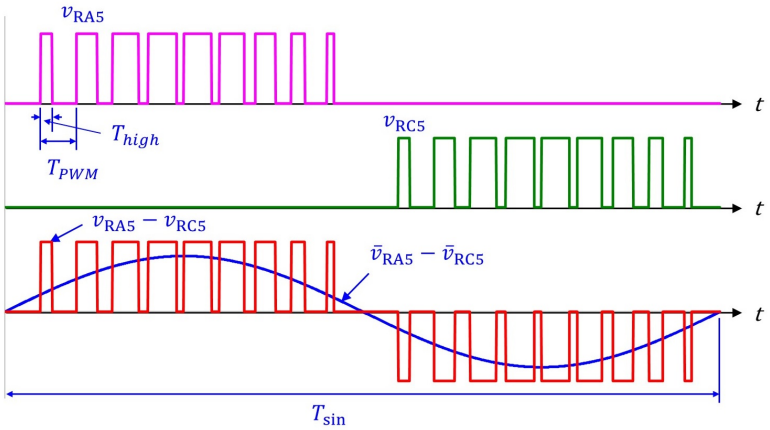


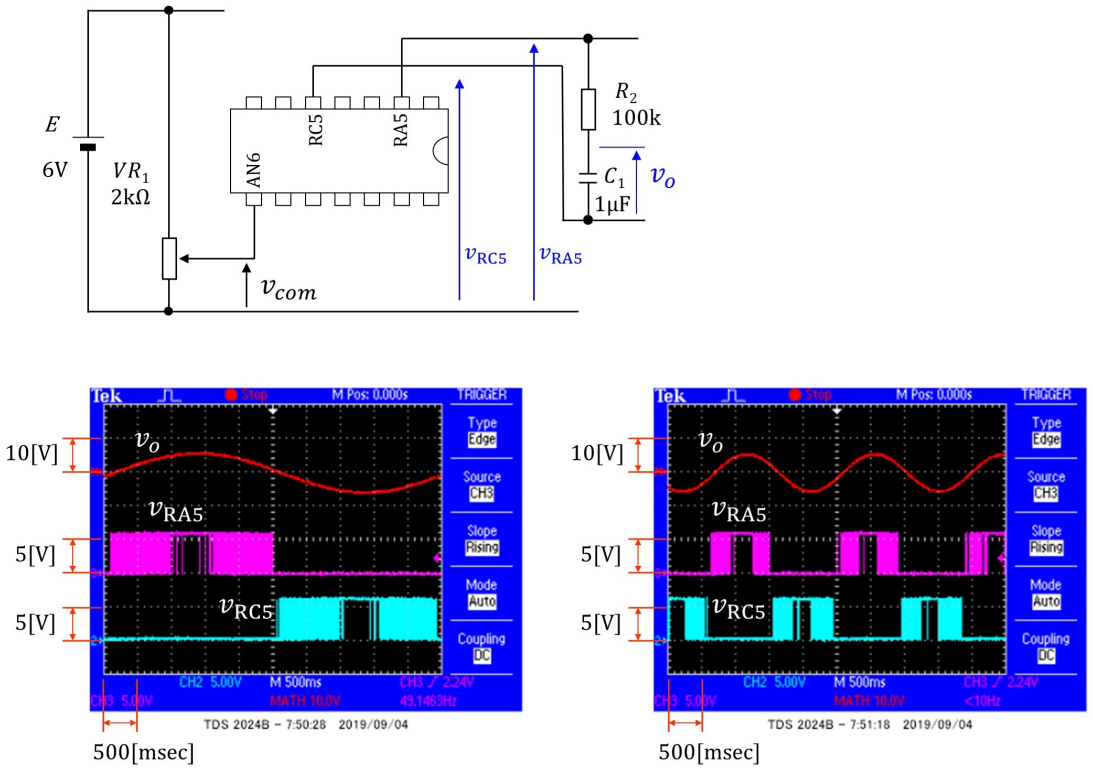
図　2.11: PWM 制御法による正弦波生成のイメージ図

図 2.11 は PWM 制御法による正弦波生成のイメージ図です．図は

$$T_{PWM} = \frac{T_{\sin}}{20} \tag{2.5}$$

の例です．実際には両周期にはもっと大きな差をつけるのですが，ここでは図の見やすさを優先して 20 と小さな値にしています．v_{RA5} が正弦波の正の半波を担当し，v_{RC5} が

負の半波を担当しています．マイコンの 2 番ピンと 5 番ピンの間の電圧は，5 番ピンを基準にすると，$v_{\mathrm{RA5}} - v_{\mathrm{RC5}}$ です．この電圧の T_{PWM} 毎の平均値 $\bar{v}_{\mathrm{RA5}} - \bar{v}_{\mathrm{RC5}}$ は近似的に正弦波となります．



(a) 周波数低　　(b) 周波数高

図 2.12: PWM 波形と正弦波形

図 2.12 は $v_{\mathrm{RA5}}, v_{\mathrm{RC5}}$ と v_o の実験波形例を示します．正弦波の周期 $T_{\sin}$ と PWM 周期 T_{PWM} の関係は

$$T_{PWM} = \frac{T_{\sin}}{240} \tag{2.6}$$

です．同図 (a) が正弦波の周波数が低い（周期が長い）場合で，(b) が周波数が高い（周期が短い）場合です．$v_{\mathrm{RA5}}, v_{\mathrm{RC5}}$ の波形には，パルスが重なってべた塗りの状態で表示されている区間が見られます．抵抗 R_2 とコンデンサ C_1 はフィルタ回路です．フィルタにより細いパルスを除去することで図中の v_o の正弦波形を見ることができます．フィルタ回路の入力電圧 $v_{in} = v_{RA5} - v_{RC5}$ とすると，v_o と v_{in} の複素数表現 V_o と V_{in} の間には

$$\begin{aligned} V_o &= \frac{1}{R_2 + \frac{1}{j\omega C_1}} \times \frac{1}{j\omega C_1} V_{in} \\ &= \frac{1}{1 + j\omega C_1 R_2} V_{in} \end{aligned} \tag{2.7}$$

の関係があります．ω は V_{in} の角周波数です．

$$\omega = 2\pi f \tag{2.8}$$

です．f は v_{in} の周波数です．これはローパスフィルタの特性です．なぜならば，

$$\begin{aligned} V_o &\approx V_{in} \qquad \left(\omega << \frac{1}{C_1 R_2}\right) \\ V_o &\approx 0 \qquad \left(\omega >> \frac{1}{C_1 R_2}\right) \end{aligned} \tag{2.9}$$

となり，V_{in} の周波数 f が低いときは，V_{in} がそのまま V_o に現れ，f が高いとき V_o は 0 に近づくからです．

$$f_c = \frac{1}{2\pi C_1 R_2} \tag{2.10}$$

はカットオフ周波数と呼ばれます．カットオフ周波数において

$$\begin{aligned} |V_o| &= \frac{1}{\sqrt{1 + (2\pi f_c C_1 R_2)^2}} |V_{in}| \\ &= \frac{1}{\sqrt{2}} |V_{in}| \end{aligned} \tag{2.11}$$

です．出力電圧の絶対値 $|V_o|$ は入力電圧の絶対値 $|V_{in}|$ の $1/\sqrt{2}$ となります．この比 $|V_o|/|V_{in}|$ は $f > f_c$ にて小さくなります．f_c をもって，ローパスフィルタが通す周波数成分の目安とします．

図 2.12 の例では $R_2 = 100$ [kΩ]，$C_1 = 1$ [μF] です．よって

$$\begin{aligned} f_c &= \frac{1}{2\pi C_1 R_2} \\ &= \frac{1}{2\pi \times 10^{-6} \times 10^5} \\ &\approx 1.6[\mathrm{Hz}] \end{aligned} \tag{2.12}$$

です．図 (a) の場合，PWM 周期 $T_{PWM} \approx 22$ [msec] です．したがって，PWM 周波数 f_{PWM} は

$$\begin{aligned} f_{PWM} &= \frac{1}{T_{PWM}} \\ &\approx \frac{1}{22[\mathrm{msec}]} \\ &\approx 45[\mathrm{Hz}] \end{aligned} \tag{2.13}$$

です．

$$f_{PWM} >> f_c \tag{2.14}$$

の関係が成立し，PWM 周波数成分はフィルタ出力 V_o にはほとんど現れません．実際に計算してみると

$$\begin{aligned} |V_o| &= \frac{1}{\sqrt{1+\left(\frac{f}{f_c}\right)^2}}|V_{in}| \\ &\approx \frac{1}{\sqrt{1+\left(\frac{45}{1.6}\right)^2}}|V_{in}| \\ &\approx 0.035|V_{in}| \end{aligned} \tag{2.15}$$

となり，6 [V] のパルスは 0.2 [V] 程度に減衰します．このため，図 2.12 の v_o の波形にはパルス状波形は見えません．

一方，例えば 0.5 [Hz] の正弦波は

$$\begin{aligned} |V_o| &\approx \frac{1}{\sqrt{1+\left(\frac{0.5}{1.6}\right)^2}}|V_{in}| \\ &\approx 0.95|V_{in}| \end{aligned} \tag{2.16}$$

となり，5%程度の減衰で済みます．カットオフ周波数より低い成分は，T_{PWM} の区間毎の平均値 $\overline{v}_{\mathrm{RA5}} - \overline{v}_{\mathrm{RC5}}$ に近い値に均されます．

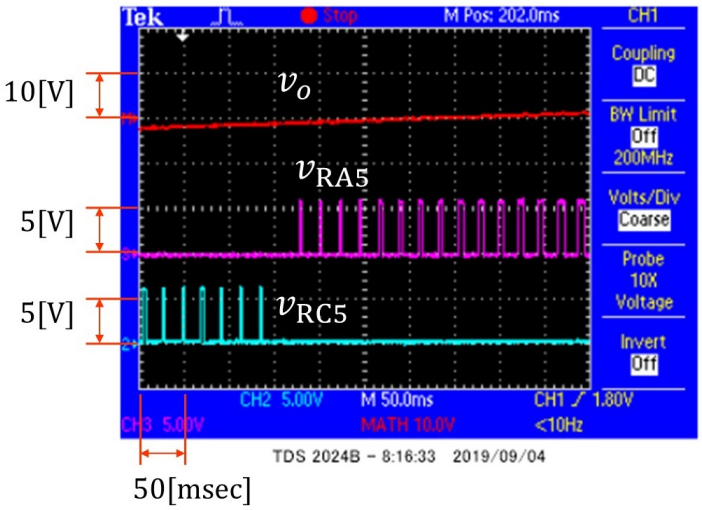


図　2.13: PWM 波形と正弦波形（時間軸拡大図）

図 2.13 は図 2.12 の時間軸を拡大した波形例です．図 2.12 ではべた塗りとなってしまっていた箇所において，時間軸を拡大することで，パルス上の波形が見えるようになりました．v_{RA5} のパルス幅 T_{high} が拡がるにつれて，出力電圧 v_o が大きくなって行く様子が分ります．

2.2 PWM モード

2.2.1 可視化 PWM モード

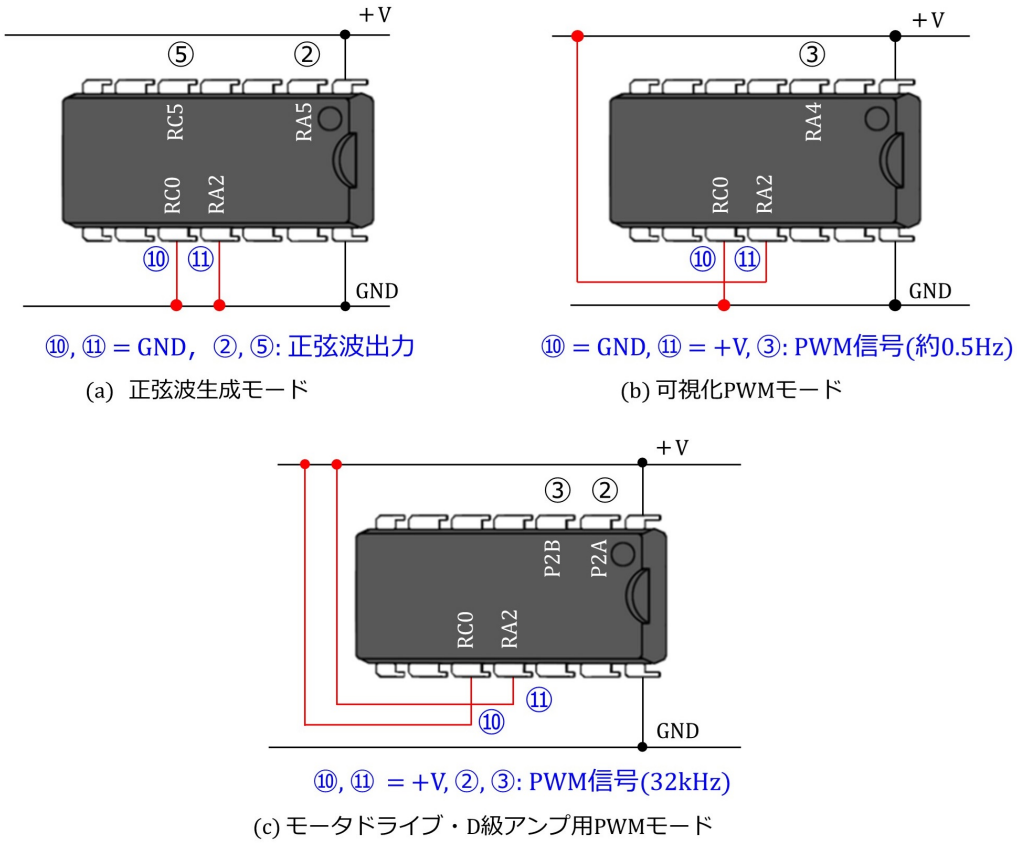
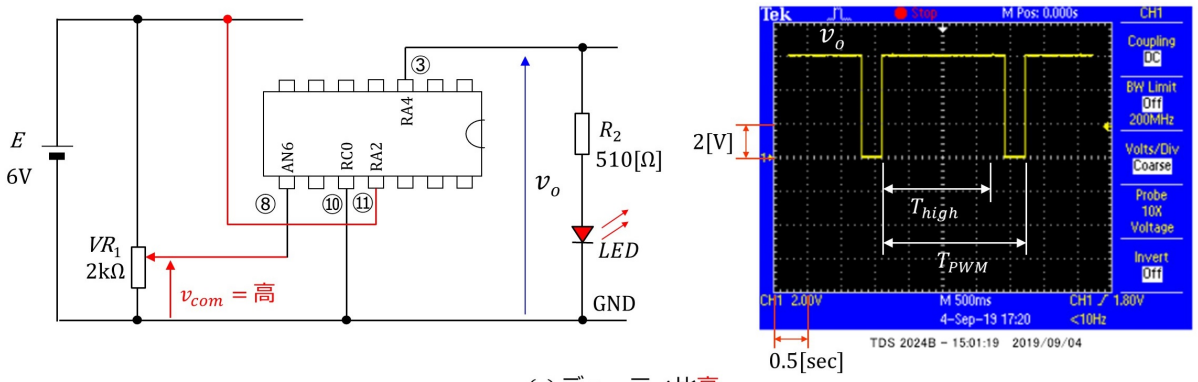


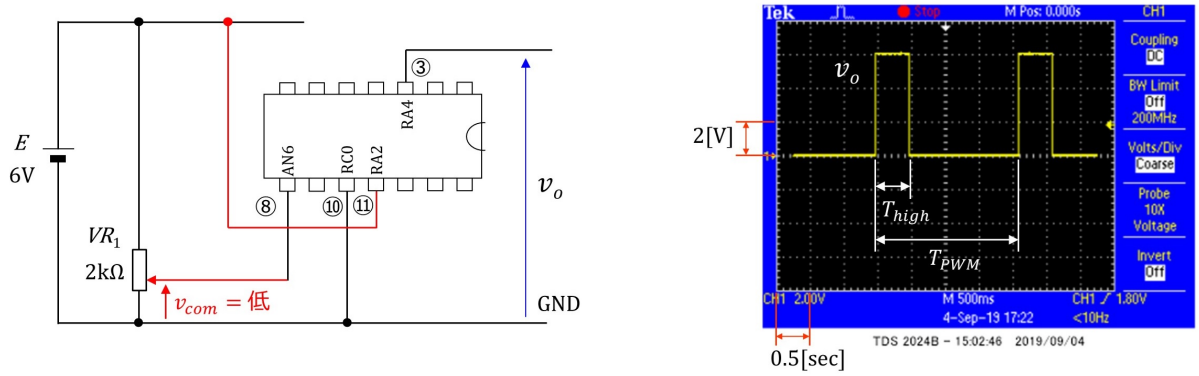
図 2.14: 3 種類のモード設定実体配線図

第 4 章のプログラムには正弦波生成モードの他に可視化 PWM モードとモータドライブ&D 級アンプ用 PWM モードがあります．電源投入前に図 2.14(a) のように 10，11 番ピンを GND につないで電源を投入すると，正弦波生成モード用プログラムが走ります．正弦波電圧は 2，5 番ピン間に出力されます．10 番ピンを GND，11 番ピンを＋電源側につなぐと可視化 PWM モード用プログラムが実行されます．3 番ピンに PWM 波形が出力されます．PWM 制御法の仕組みを LED の点滅で確認できます．そして，10, 11 番ピンともに＋電源側につなぐと，このマイコンはモータドライブ&D 級アンプ用 PWM 信号を生成します．PWM 信号は 2 番ピンと GND 間，3 番ピンと GND 間に出力されます．2，3 番ピンの PWM 信号は相補的（一方が＋ V/GND を出力しているとき，もう一方は GND/＋ V につながっている関係）です．

正弦波生成モードは 2.1 節に解説したので，本節では 2 種類の PWM モードを解説し



(a) デューティ比高



(b) デューティ比低

図　2.15: 可視化 PWM モードの動作波形例

ます．図 2.15 は可視化 PWM モードの動作波形例です．同図 (a) がデューティ比が高い場合，(b) が低い場合です．PWM 制御法の基本概念は図 2.10 に述べたものと同じです．デューティ比を δ，PWM 周期を T_{PWM}，パルス幅を T_{high} とすると

$$\delta = \frac{T_{high}}{T_{PWM}} \tag{2.17}$$

です．(a), (b) ともに $T_{PWM} \approx 2$ [sec] です．マイコン内のプログラムは 8 番ピンの電圧 v_{com} を A-D 変換モジュールにより読み込んで，v_{com} が高いときにデューティ比 δ が高くなるように制御しています．同図 (a) の回路図のように 3 番ピンと GND 間に，抵抗 R_2 と LED（Light Emitting Diode : 発光ダイオード）を直列に接続することで，PWM 制御の様子を LED の点滅で確認できます．

2.2.2 モータドライブ&D 級アンプ用 PWM モード

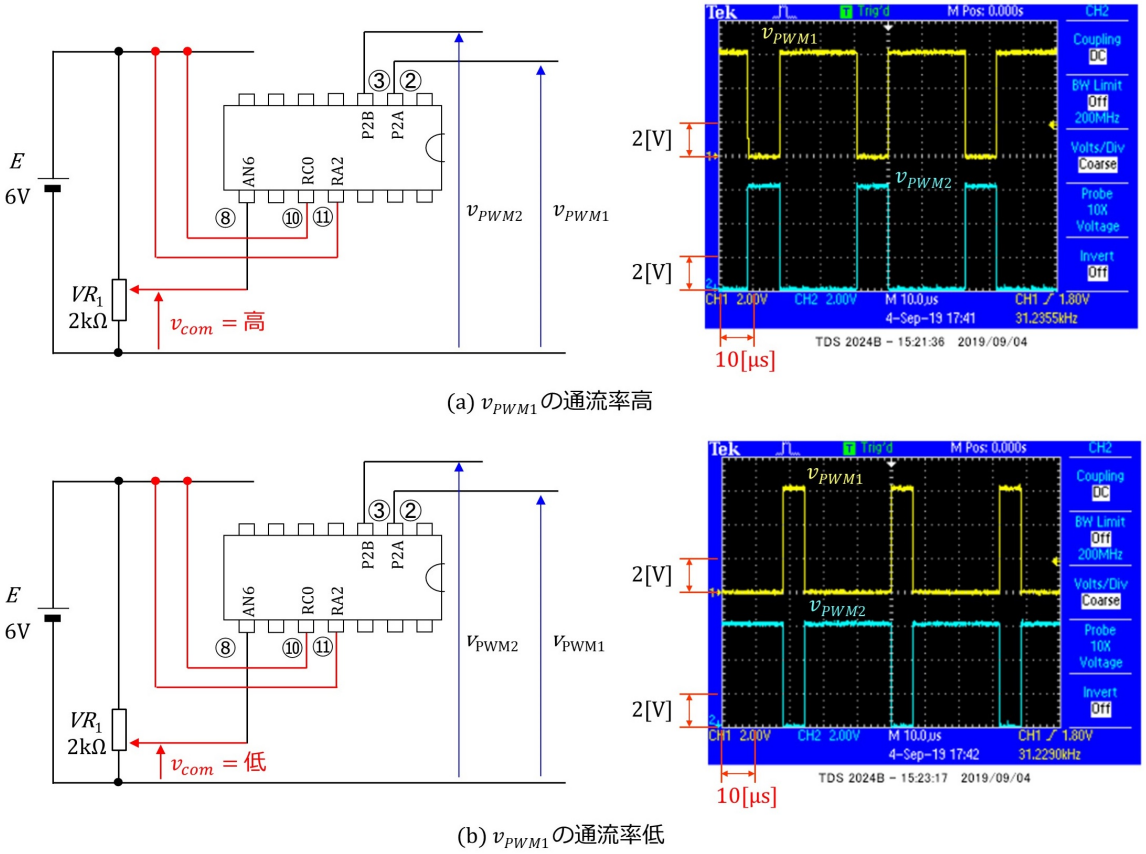


図 2.16: モータドライブ&D 級アンプ用 PWM モードの動作波形例

図 2.16 はモータドライブ&D 級アンプ用 PWM モードの動作波形例です．2, 3 番ピンの出力電圧をそれぞれ v_{PWM1}, v_{PWM2} とします．前項の可視化 PWM モードとの大きな違いは PWM 周期 T_{PWM} です．オシロスコープ画面のスナップショットから $T_{PWM} \approx 32[\mu\text{sec}]$ です．同図 (a) が v_{PWM1} のデューティ比が高い場合，(b) が低い場合です．マイコン内のプログラムは 8 番ピンの電圧 v_{com} を A-D 変換モジュールにより読み込んで，v_{com} が高いときに v_{PWM1} のデューティ比 δ が高くなるように制御しています．v_{PWM1}, v_{PWM2} は相補的（一方が＋ V/0 を出力しているとき，もう一方は 0/＋ V を出力している関係）であることが分ります．

第 3 章

MPLAB® X IDE, XC8 コンパイラ，New Project，デバッガ

3.1 MPLAB® X IDE, XC8 コンパイラのインストール方法

本稿では Microchip 社が無償提供している統合開発環境 MPLAB® X IDE (Integrated Development Environment：統合開発環境，および，MPLAB® XC8 コンパイラを使用します．

MPLAB® X IDE は，Microchip 社のホームページ → Tools and Resources → Develop → MPLAB® X IDE とたどることで，ダウンロード画面を開くことができます (2024 年 1 月時点)． Windows の場合は，MPLAB X IDE Windows を左クリック（マウスの左ボタンを 1 回クリック）すると，インストーラ (MPLABX-v6.15-windows-installer.exe) をダウンロードできます．このインストーラを立ち上げ，インストーラの推奨通りに Next ボタンを押していくことで，MPLAB® X IDE をインストールできます．無事インストールに成功すれば，C:¥Program Files (x86) および C:¥Program Files のフォルダ内に Microchip という名前のフォルダが，また，ユーザアカウントフォルダ（デスクトップフォルダやドキュメントフォルダ等が入っているフォルダ）内に MPLABXProjects という名前のフォルダが作られます．

同様に，MPLAB® XC8 コンパイラは Microchip 社のホームページ → Tools and Resources → Develop → MPLAB® XC8 Compiler (2024 年 1 月時点) とたどることでインストーラ (Windows の場合は xc8-v2.45-full-install-windows-x64-installer.exe) をダウンロードできます．このインストーラを立ち上げ，推奨通りに Next ボタンを押していくことで，XC8 コンパイラをインストールできます．無事インストールに成功すると，C:¥Program Files¥Microchip フォルダ内に xc8¥v2.45 という名前のフォルダが作られます．

3.2 プロジェクトの読み込み

本稿で解説するプロジェクトは，本稿と同じ
パワーエレクトロニクスノート
圧縮ファイル sin_wave_2types_PWM_MCC.zip にあります．ダウンロード，解凍して，sin_wave_2types_PWM_MCC.X フォルダを MPLABXProjects フォルダ内に置いてください．

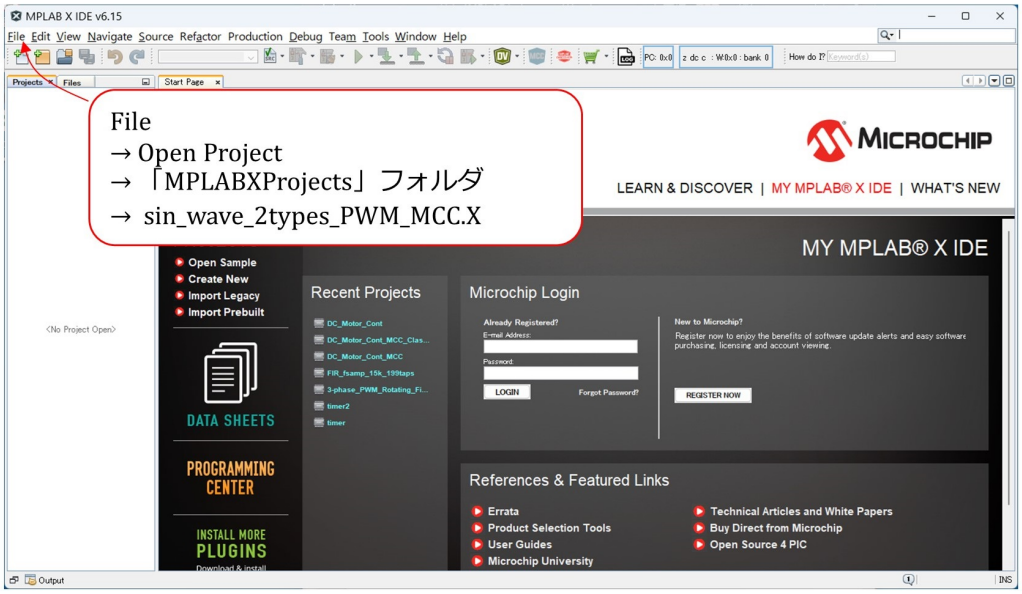


図 3.1: ダウンロードした Prject の読み込み

MPLAB® X IDE のアイコンを左ダブルクリックすることで，この統合開発環境を立ち上げることができます．図 3.1 の画面が立ち上がったら，File → Open Project を選択します． → 「MPLABXProjects」フォルダ → sin_wave_2types_PWM_MCC.X を選択することで，同プロジェクトを MPLAB® X IDE に読み込めます．

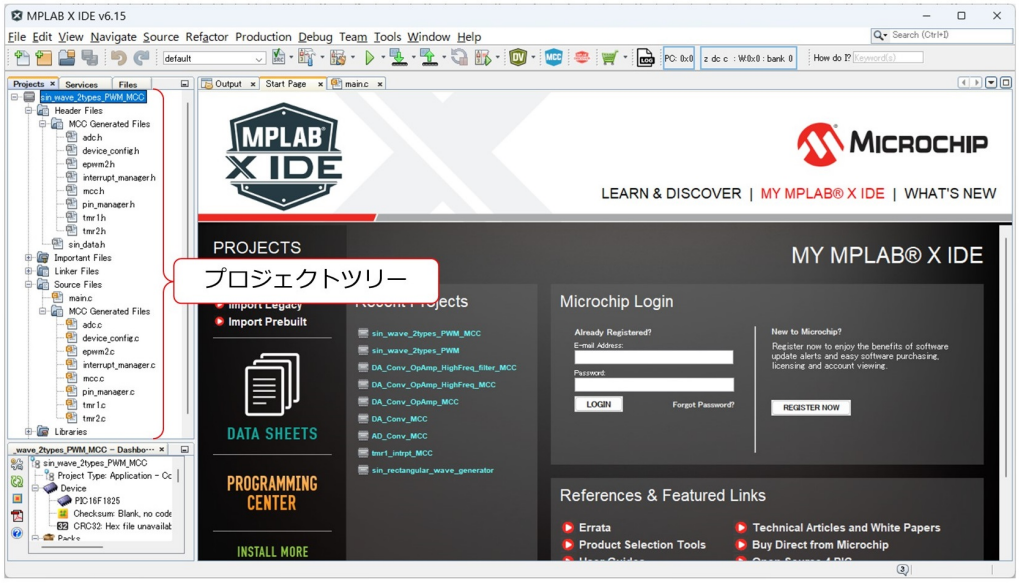


図 3.2: sin_wave_2types_PWM_MCC のプロジェクトツリー

図 3.2 は，MPLAB® X IDE に読み込んだ，sin_wave_2types_PWM_MCC のプロジェクトツリーです．これは MCC により自動生成された MCC Genrated Files と，筆者の作成した sin_data.h, main.c ファイルからなります．main.c 内に正弦波生成モード，可視化 PWM モード，モータドライブ&D 級アンプ用 PWM モードのコードを書き込みました．

3.3 プログラム書き込み用回路

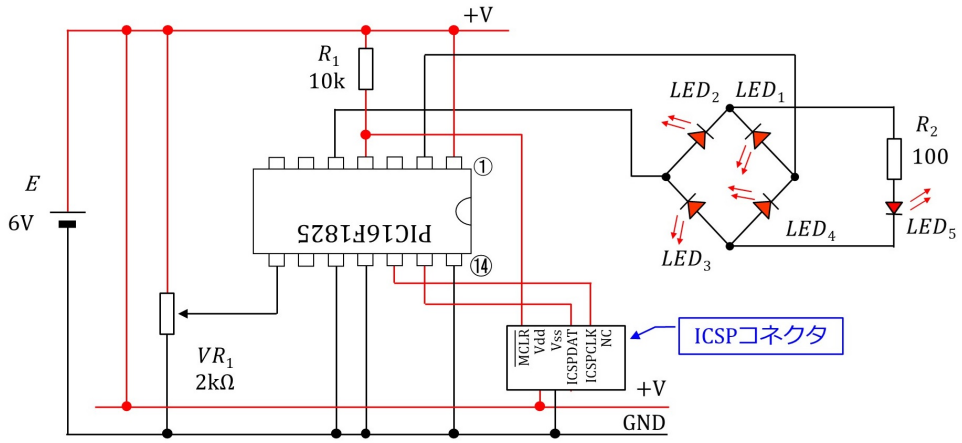


図 3.3: In-Circuit Debugger/Prgrammer によるプログラム書き込み用回路

MPLAB® X IDE の準備が整ったら，プログラム書き込み用回路の製作です．図 3.3 は In-Circuit Debugger/Programmer によるプログラム書き込み用回路です．Microchip 社の In-Circuit Debegger/Programmer にはいくつかありますが，この回路は MPLAB® Snap, PICkit™4 用です．図 2.6 の正弦波生成実験回路に ICSP コネクタと全波整流回路をつなげてあります．ICSP は In-Circuit Serial Programming の略です．ISCP コネクタに MPLAB® Snap, PICkit™4 をつなぐことで，プログラムの書き込みとデバッギングができます．$\overline{\text{MCLR}}$, ICSPDAT, ICSPCLK) などはピンの名称です．全波整流回路は LED $LED_1 \sim LED_5$ と抵抗 R_2 からなり，正弦波を全波整流する様子を LED で可視化します．

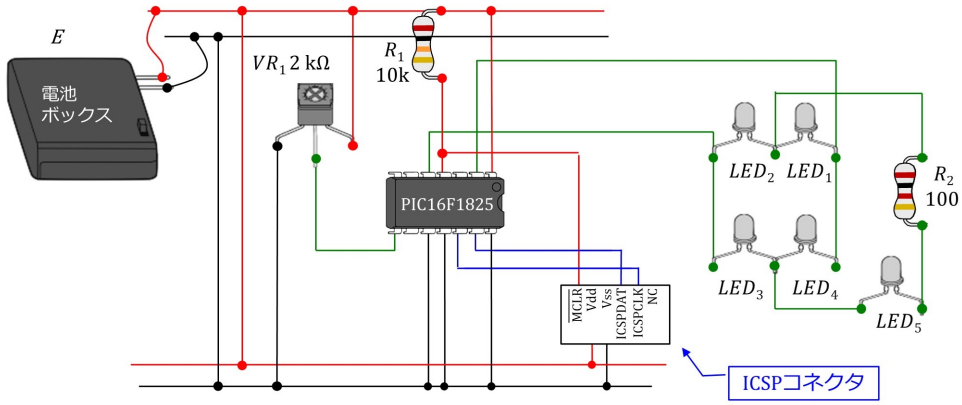
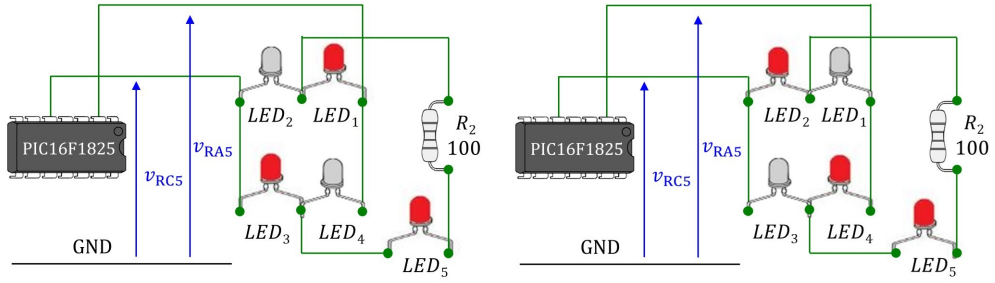


図 3.4: In-circuit Prgrammer によるプログラム書き込み用回路の実体配線図

図 3.4 はプログラム書き込み用回路の実体配線図です．ICSP コネクタは 2 個の部品（ピンヘッダ，ピンソケット）を組み合わせて作ります．



(a) $v_{RA5} > v_{RC5}$のとき　　(b) $v_{RA5} < v_{RC5}$のとき

図　3.5: プログラム書き込み成功したときの回路動作

図 3.5 はプログラム書き込みが成功したときの回路動作を示します．第 4 章のプログラムの書き込みに成功すると，マイコンは 2.1 節の正弦波生成モードで動作します．正の半波（$v_{RA5} > v_{RC5}$）のとき，同図 (a) のように LED_1, LED_3, LED_5 が点灯し，負の半波（$v_{RA5} < v_{RC5}$）のとき，同図 (b) のように LED_2, LED_4, LED_5 が点灯します．各 LED の明るさは正弦波電圧に応じて変化します．

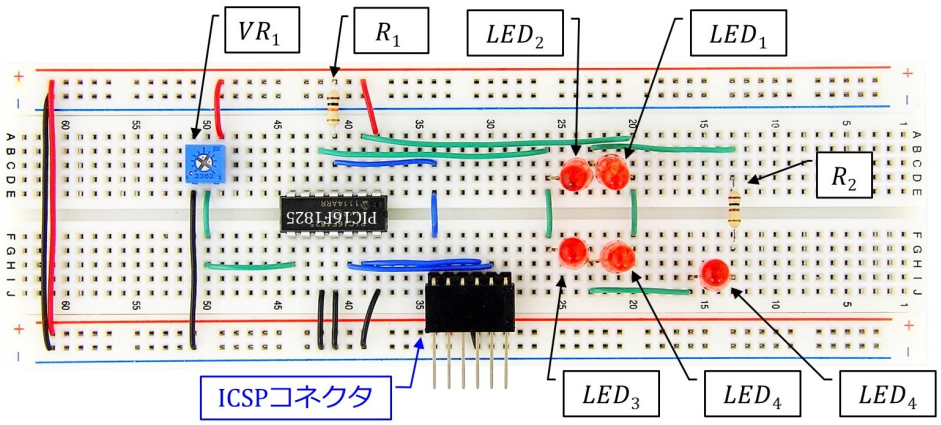


図　3.6: In-circuit Prgrammer によるプログラム書き込み用回路の配線例

図 3.6 はプログラム書き込み用回路の配線例です．

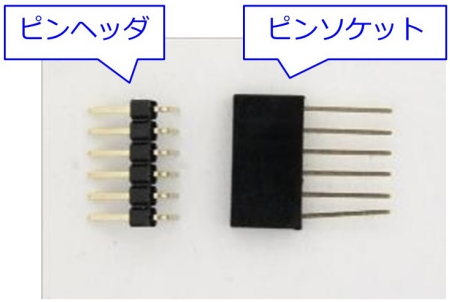
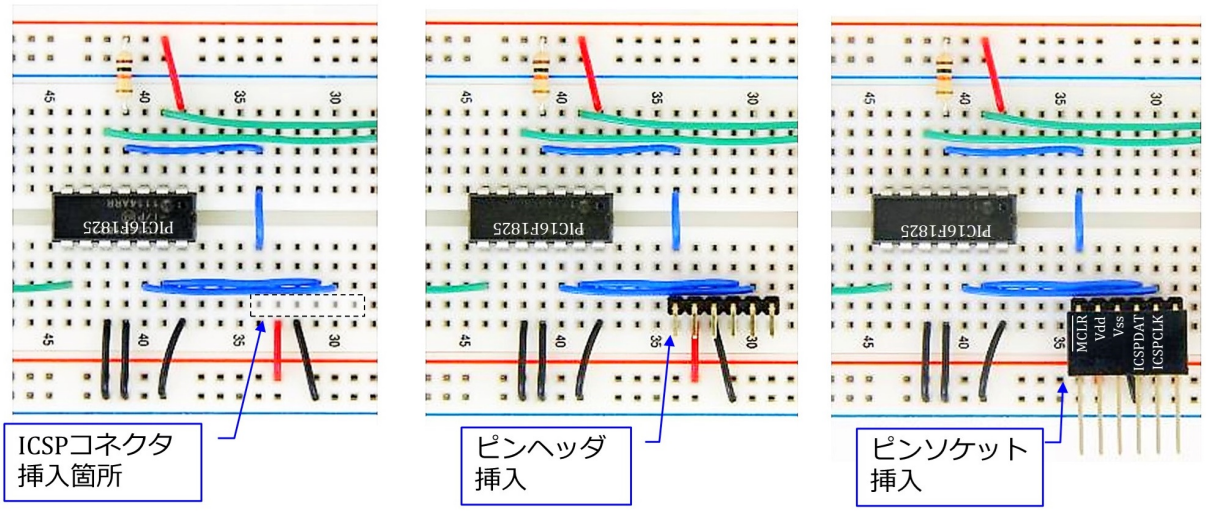


図 3.7: ピンヘッダとピンソケット

ICSP コネクタは，図3.7の6ピンのピンヘッダとピンソケットを組み合わせて作ります．



(a) ICSPコネクタ挿入箇所　(b) ピンヘッダの挿入　(c) ピンソケットの挿入

図 3.8: ICSP コネクタの製作

Fig.3.8 は ICSP コネクタの製作過程を示します．同図 (a) の破線で囲った 6 個の穴にピンヘッダを挿入します．同図 (b) がピンヘッダを挿入した写真です．同図 (c) のように，このピンヘッダにピンソケットを挿入して ICSP コネクタができあがります．

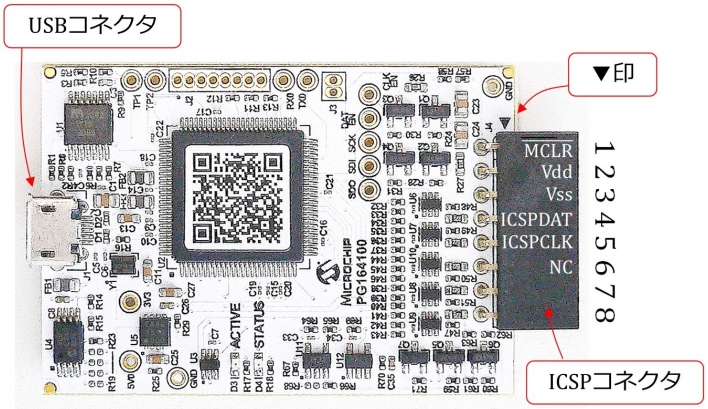


図 3.9: MPLAB® Snap

図 3.9 は MPLAB® Snap の外観とピン配置を示します．ICSP コネクタのピン穴は 8 個あります．▼記号側が 1 番ピンです．Fig.3.8 のオスピンを挿入できます．左側は USB コネクタで，パソコンと接続できます．

図 3.10 は Fig.3.8 の ICSP コネクタに MPLAB® Snap を挿入した様子です．写真のように，左端を揃えて挿入します．MPLAB® Snap は USB ケーブルによりパソコンとつなぐことができ，パソコンよりプログラム書き込みおよびデバッギングができます．

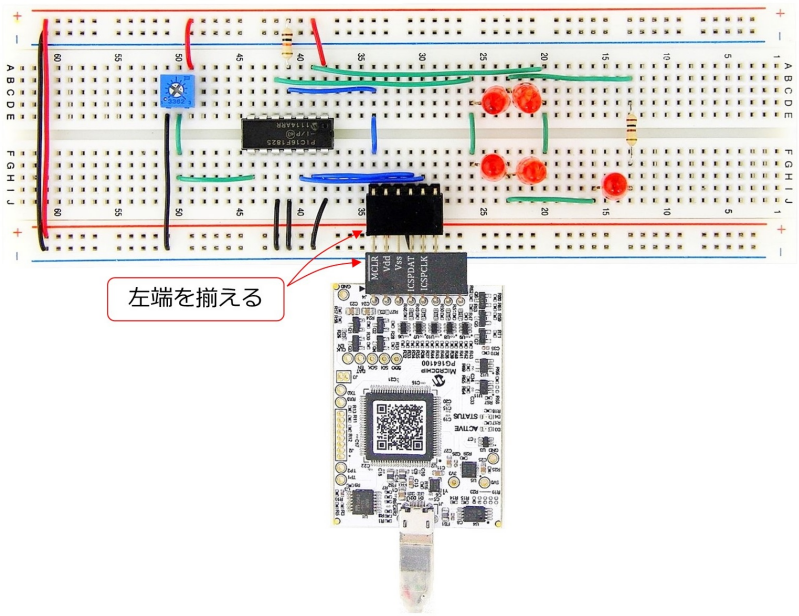


図 3.10: ICSP コネクタに MPLAB® Snap を接続

図　3.11: ゼロプレッシャーソケット

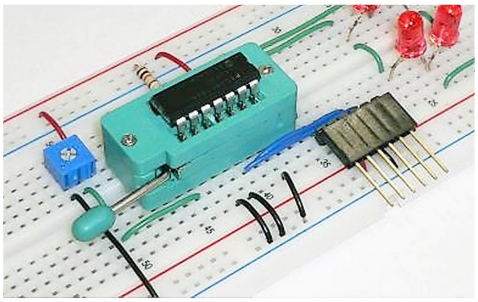


図　3.12: ゼロプレッシャーソケット使用によるマイコン挿抜の容易化

以上でマイコンにプログラム書き込みができるようになったのですが，たくさんのマイコン（例えば 100 個）にプログラム書き込みを行うには，マイコンのブレッドボードへの挿抜が手間です．そのようなときにはゼロプレッシャーソケットが便利です．図 3.11 はゼロプレッシャーソケットの外観です．写真のようにレバーを立てた状態では，ソケット口は広く開いていて，圧力ゼロ（ゼロプレッシャー）です．マイコンをソケット口に挿入できます．レバーを倒すとソケット口が固く閉じるため，マイコンをゼロプレッシャーソケットに固定できます．そこで，あらかじめゼロプレッシャーソケットだけをブレッドボードに差し込んでおきます．そして，マイコンをソケット口に挿入してレバーを倒すことで，マイコンを回路に接続できます．図 3.12 はゼロプレッシャーソケットによりマイコンをブレッドボード上の回路に接続した様子です．これでマイコンにプログラムを書き込むことができます．書き込み後は，レバーを立てることでマイコンを抵抗なくソケットから抜き取ることができます．

3.4 プログラムの書き込み

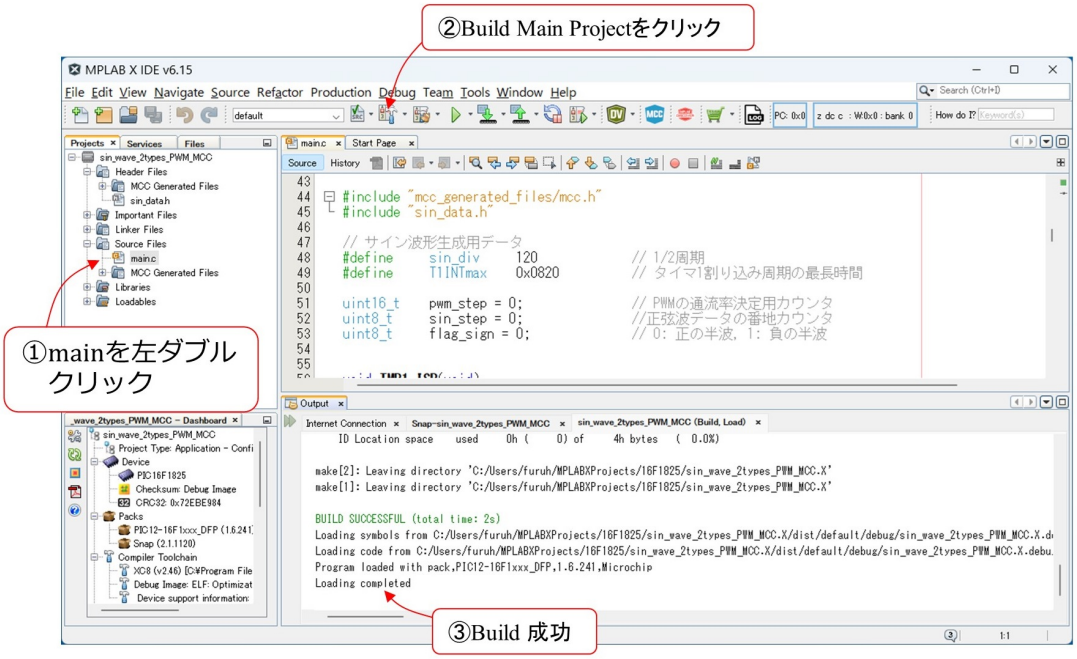


図 3.13: MPLAB® X IDE: Build

以上で MPLAB® X IDE による PIC マイコンへの書き込み準備が完了しました．電池ボックスに 4 本の乾電池 (6[V]) もしくは充電池 (5[V]) を入れて，ブレッドボードの＋電源および GND ラインに接続して電圧を印加します．ICSP コネクタに MPLAB® Snap を接続し，MPLAB® Snap とパソコンを USB ケーブルで接続します．

図 3.13 のように，画面内の main.c を左ダブルクリックすることで，このファイル内のプログラムを開くことができます．Build Main Project ボタンをクリックして，同図下のように

BUILD SUCCESSFUL (total time: xxs)

Loading completed

のメッセージが出れば，文法上のエラーは無いので，いよいよマイコンへの書き込みです．

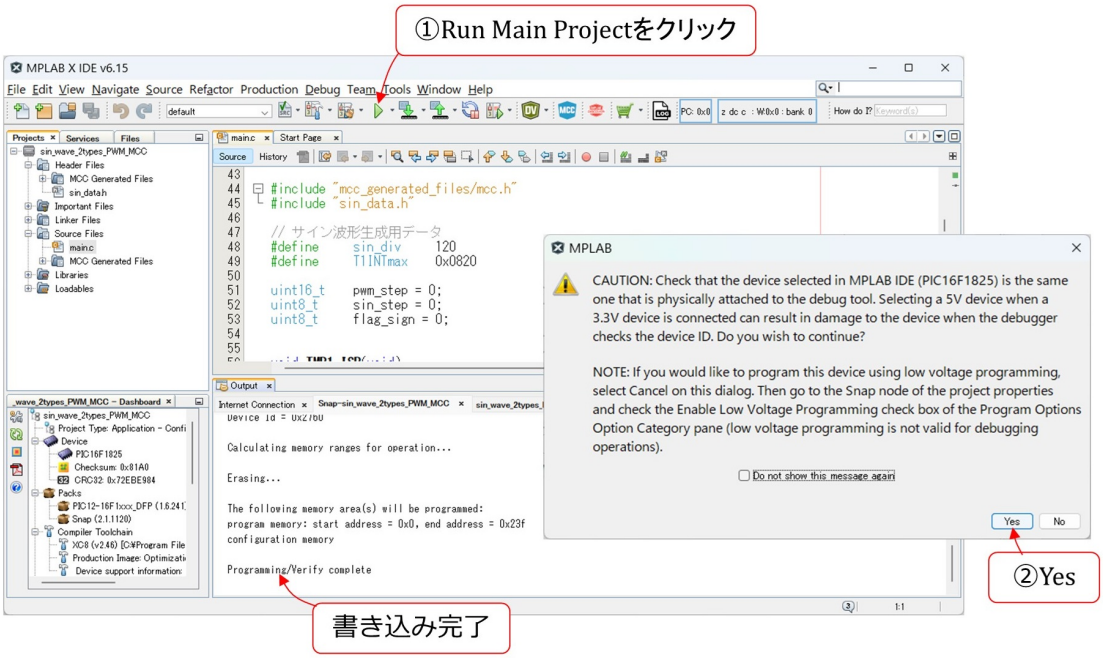


図 3.14: MPLAB® X IDE: Build とマイコンへの書き込み

図 3.14 のように ▷(Run Main Project) ボタンを押すとプログラムの Build とマイコンへの書き込みを実行されます．途中で，実際につながれているマイコンが，PIC16F1825 ではなく，低電圧仕様のものだと壊してしまうことの注意が表示されます．マイコンが PIC16F1825 であることを確認して Yes ボタンをクリックします．また，使用ツールを問い合わせるウィンドウが開かれた場合は，Show All にチェックを入れて，プルダウンメニューから Snap-xx を選択します．

同図下の画面のように

Programming/Verify complete

が表示されれば，書き込み完了し，図 3.5 の様に LED の明るさが変わり始めます．可変抵抗器上面のつまみをネジ回しで回すと，明かりの変化周期が変わります．

3.5 デバッガの使用方法

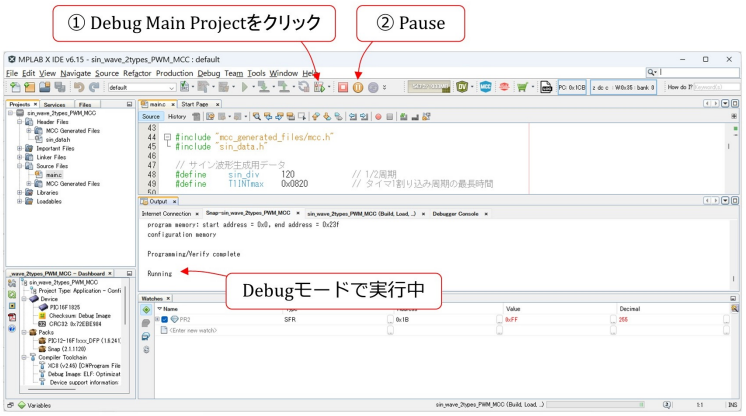


図 3.15: デバッガの起動

プログラムを改変したいときにはデバッガが便利です．図 3.15 の Debug Main Project ボタンを左クリックすることで，プログラムのビルドとマイコンへの書き込み，デバッガの起動を実行します．

Running の文字が，マイコンがデバッグモードで実行中であることを表します．

Pause ボタンを左クリックすることで，プログラムを一時停止できます．

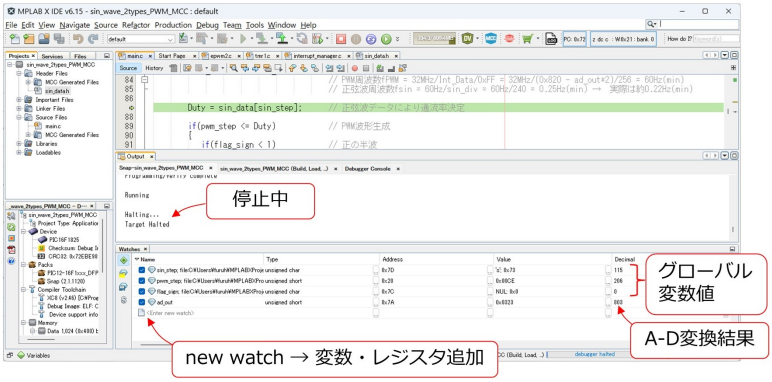


図 3.16: デバッガの一時停止と変数・レジスタ値の表示

停止時のグローバル変数の値，レジスタの値などを Watch エリアに表示させることができます．例えば，第 4 章のプログラムは，8 番ピンの入力電圧を A-D 変換して 10 ビットの値で読み込み，グローバル変数 ad_out に格納します．図 3.16 はマイコンの一時停止中に，ad_out の値を表示している画面です．Enter new watch を右クリックすると，プル

ダウンメニューが表示されます．New Watch を選択すると，表示可能なレジスタおよび変数の一覧が表示されます．その中から ad_out を選択して OK ボタンを押すと，図のように ad_out の値が表示されます．値の表示形式は 10 進，16 進，2 進を選べます．図 3.16 では，さらにグローバル変数の sin_step, pwm_step を選択・表示させています．

第4章

プログラム

4.1 New Project の作成方法

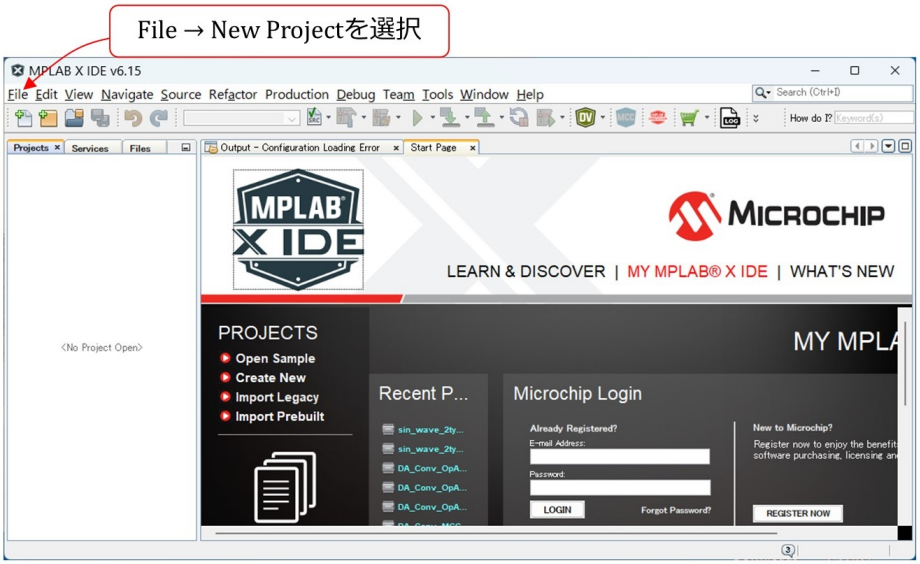


図 4.1: MPLAB® X IDE の立ち上げと New Project の選択

既成のプロジェクトを使わずに，全く新しいプロジェクトを作成する場合を紹介します．MPLABprjects フォルダ内には，まだ sin_wave_2types_PWM_MCC.X プロジェクトが作られていないとします．

MPLAB® X IDE のアイコンを左ダブルクリックすることで，この統合開発環境を立ち上げることができます．図 4.1 の画面が立ち上がったら，File → New Project を選択します．

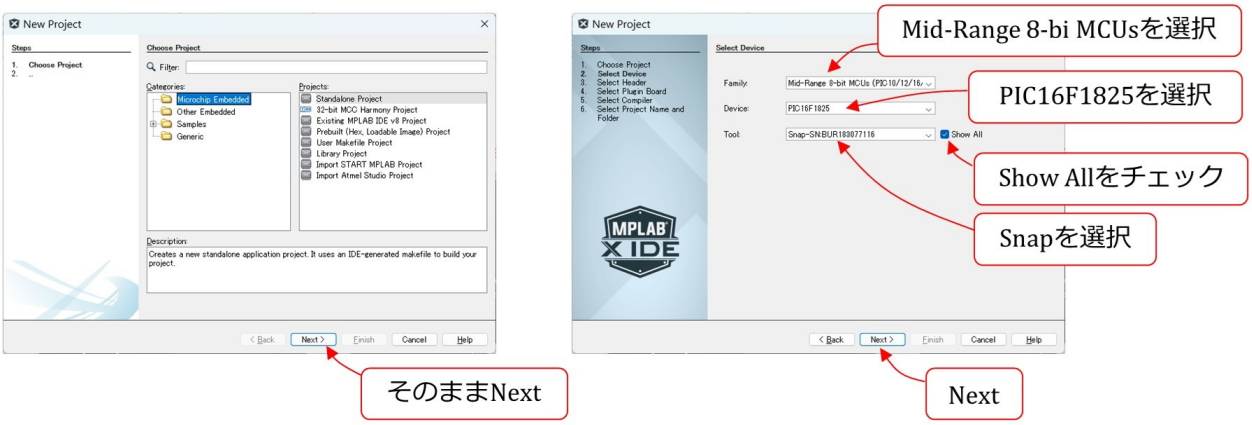


図 4.2: MPLAB® X IDE: Device 選択

図 4.2 のように進み，デバイスに PIC16F1825 を選択し，ツールに Snap を選択します．プルダウンメニューに Snap が見当たらない場合は，Show All にチェックを入れます．パソコンに Snap がつながれていると，「Snap-SNxxxx」と，SN 以降の文字が表示されます．つながれていない場合は，Snap の文字だけが表示されます．後者の場合は，プログラムをマイコンに書き込む際に，再度使用ツールの問い合わせがあります．

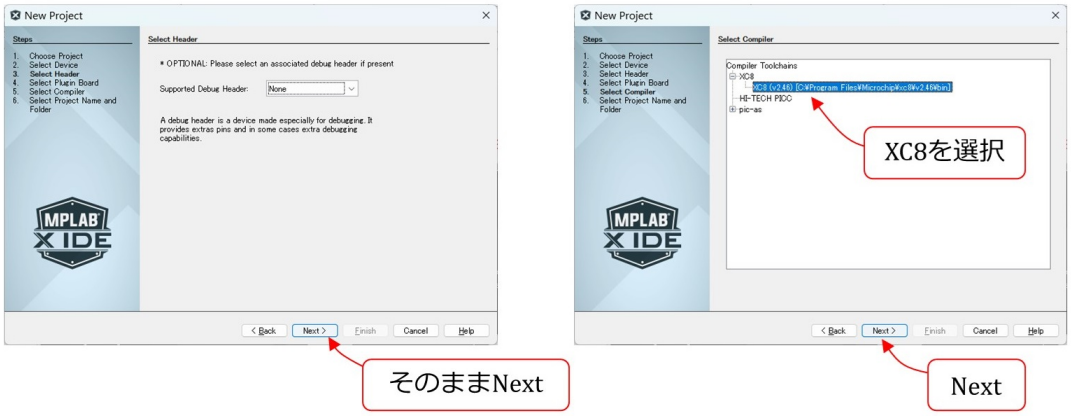


図 4.3: MPLAB® コンパイラの選択

その後は図 4.3 のようにコンパイラに XC8(vx.xx).... を選択します．

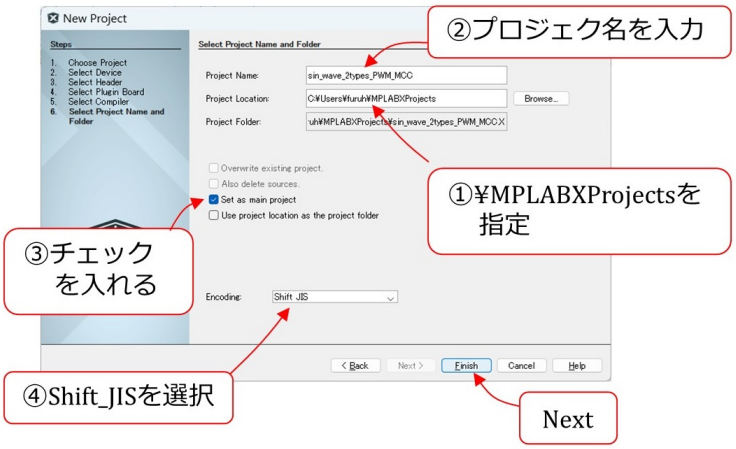


図 4.4: プロジェクト名，フォルダ位置, 言語選択

図 4.4 は次に表示される画面です．MPLABXProjects フォルダをブラウズし，プロジェクト名を自分で決めて（例えば，sin_wave_2types_PWM_MCC とします.）入力し，”Set as main project”にチェックを入れて，言語に Shift_JIS を選択します．

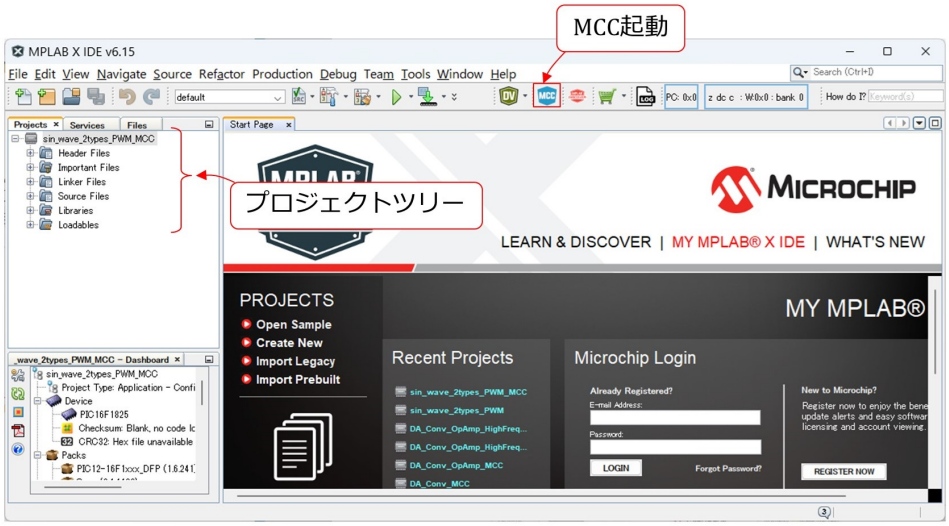


図 4.5: New Project ツリー

以上が完了した段階で，MPLABXProjects フォルダ内に「sin_wave_2types_PWM_MCC」という名前の）フォルダが作られます．そして，図 4.5 の New Project ツリーが作られます．図 3.2 のプロジェクトツリーとの違いは，New Project ツリーの各フォルダ内にはファイルが 1 つもありません．

4.2 MCC による周辺モジュール設定関数の自動生成

本章では MCC(MPLAB® Code Cofigurator) を用いて，周辺モジュール設定関数を生成します．MCC は，MPLAB® X IDE に組み込まれた，グラフィカルなプログラム開発環境です．PIC マイコン内の周辺モジュール設定関数 (C のコード) を，ほぼボタンクリックのみで生成できます．データシートをくまなく読み込まないと書けなかった周辺モジュール設定関数を，自動生成してくれる画期的なユーザインタフェースです．慣れると快適な環境です．

筆者は，これまで，周辺モジュール設定関数を自作してきました．そして，その引数を英語で表現して，引数とレジスタ数値との対応表をヘッダファイルにまとめました．MCC が無かった頃からの手法です．この従来手法によるプログラムとその解説記事は，本稿と同じ

パワーエレクトロニクスノート

に旧稿として残してあります．MCC は周辺モジュール設定関数を自動生成してくれるので，従来のデータシートと首っ引きで行わなければならなかった周辺モジュール設定関数記述を「ほぼ」不要にしました．ここで，ほぼ，としたのは，MCC の Easy Setup ではできない設定が必要になったときには，データシートを読まざるを得ないからです．そのような例外処理の際には，旧稿がお役に立つかと思います．

4.2.1 MCC の起動

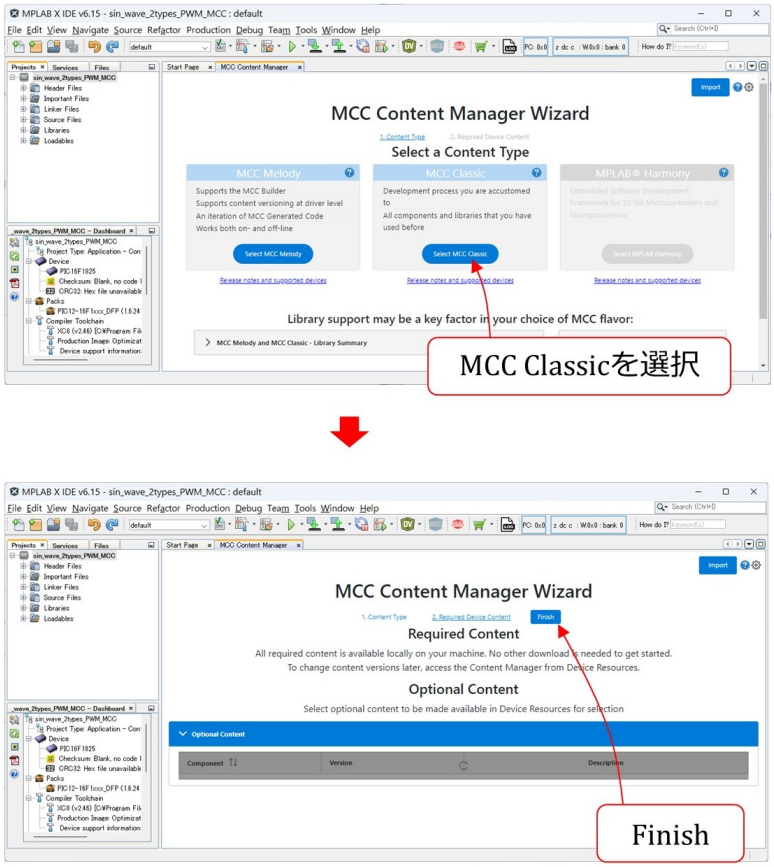


図 4.6: MCC の起動

図 4.5 の MCC ボタンを左クリックすることで MCC を立ち上げられます．図 4.6 の MCC Classic を選択し，finish ボタンが現れたら，これを左クリックします．

4.2.2 システムモジュール設定

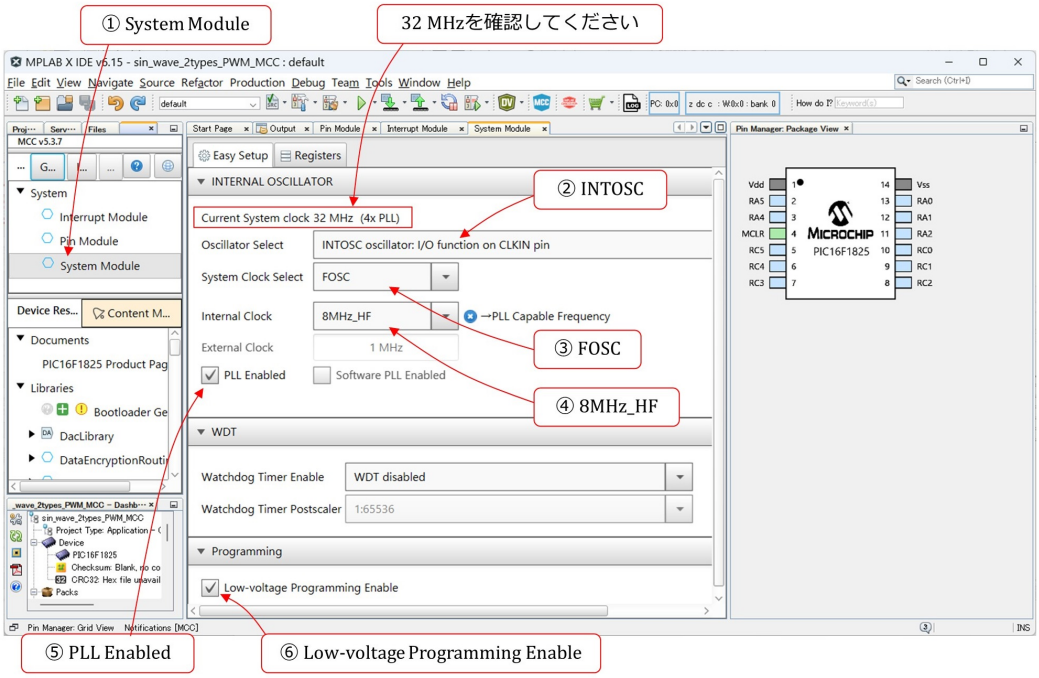


図 4.7: システムモジュールの設定

まず，System Module の設定から始めます．図 4.7 のように，画面左上の Project Resources エリア内の System Module の文字を左クリックします．すると，画面中央の Composer エリアに INTERNAL OSCILLATOR 等の設定画面が開かれ，画面右の Pin Manager エリアにピン配置のパッケージ表示が示されます．以下の手順で設定を進めます．

(2) INTOSC oscillator を選択

マイコン内蔵のオシレータ (INTOSC oscillator) を使用します．本稿の製作ではクロックに高い精度を必要としないので，マイコン内蔵の精度の低い (±1%) クロックを使います．精度の高いクロックを必要とする場合には，マイコンに水晶発振子を外付けする方法があります．

(3) FOSC を選択

システムクロック源を FOSC とします．

(4) 8MHz_HF を選択

内蔵オシレータのクロック周波数を 8MHz とします．この周波数を選ぶと 4 逓培 PLL

が使用可能になります．選択肢に 16MHz がありますが，16MHz では 4 逓倍 PLL は使えません．

(5) PLL Enabled をクリック

4 逓倍 PLL を使って，内蔵オシレータのクロック周波数を 4 倍にできます．これにより，FOSC を 4 × 8 MHz = 32 MHz とします．この周波数がこのマイコンの最高動作周波数です．

(6) Low-voltage Programming Enable にチェックマークが入っていることを確認

Low-voltage Programming(LVP) には，デフォルトでチェックマークが入っています．MPLAB® SNAP は LVP でしか使えません．このチェックマークを外して，MPLAB® SNAP でマイコンへの書き込みを行うと，エラーが出ます．チェックは外さないでください．

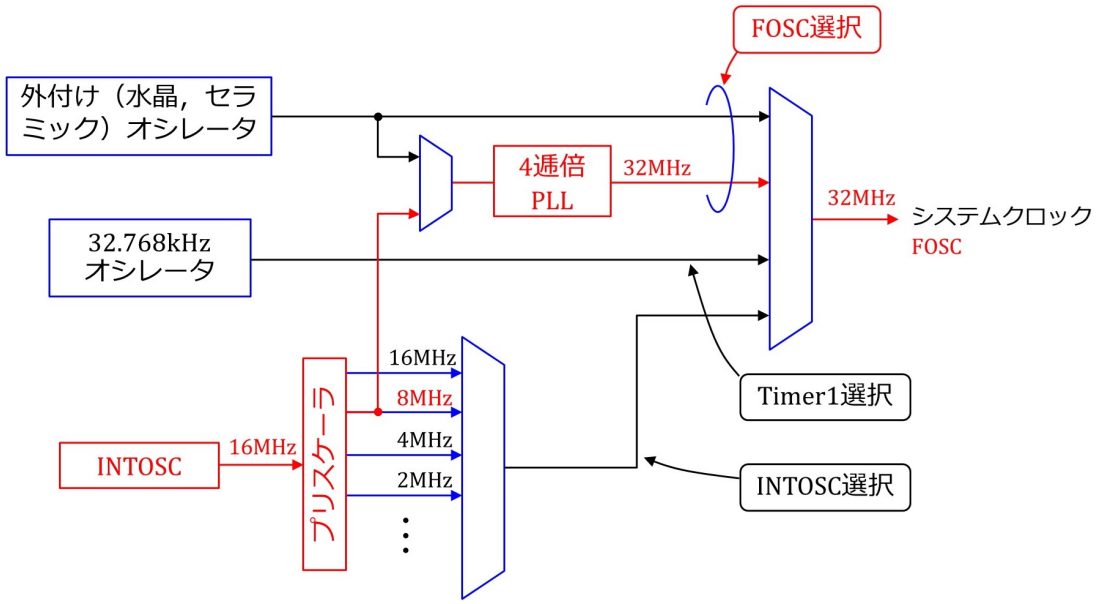


図 4.8: クロックソースのブロック図

以上の設定により，システムクロック FOSC が定まります．システムクロックは周辺モジュールに供給されます．図 4.8 は，クロックソースのブロック図です．上記設定によるシステムクロック生成経路を赤線で示します．内蔵オシレータ INTOSC は 16MHz のクロックを生成します．Internal Clock の 8MHz 設定により，プリスケーラ出力が 8MHz のクロックになります．そして，4 逓培 PLL により，この 8MHz クロックは，4 逓培され

て 32MHz のクロックになります．最後に，FOSC 選択により，システムクロック FOSC は PLL 出力の 32MHz クロックになります．ここでは，クロック源の 1 つが FOSC と呼ばれ，システムクロックそのものも FOSC と読ばれています．紛らわしいですが，以降，本稿では FOSC はシステムクロックを指します．

4.2.3 タイマ 1 モジュールの設定

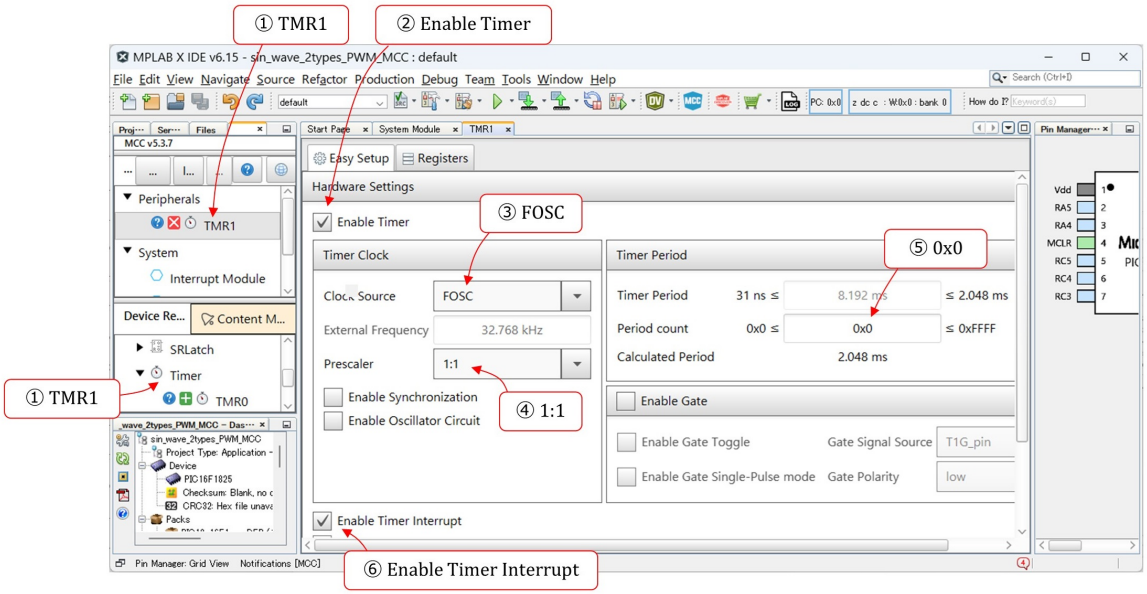


図 4.9: タイマ 1 モジュールの設定

図 4.9 はタイマ 1 モジュールの設定画面です．画面左中段の Device Resources エリアの TRM1 を選択すると，画面左上の Project Resources エリアに TMR1 の文字が現れ，画面中央の Composer エリアにタイマ 1 モジュールの設定項目が表示されます．一定時間間隔でタイマ 1 による割り込みを発生させるように，タイマ 1 を設定します．設定手順は以下のとおりです．

(2) Enable Timer を左クリック

タイマ 1 モジュールを起動します．

(3) FOSC を選択

タイマ 1 モジュールのクロック周波数を FOSC(= 32 MHz) とします．

(4) 1:1 を選択

クロックを 1/1 に分周します．クロック周波数は FOSC のまま不変です．

(5) 0x0 に設定

タイマ 1 による割り込み周期を暫定的に 0x0 (=2.048 ms) に設定します．TMR1 レジスタには，初期値として 0x0 が書き込まれます．TMR1 は 16 ビットレジスタです．

クロックによりカウントアップして，オーバーフローしたところで割り込み処理関数を起動します．オーバーフローまでのカウントアップ回数は

$$2^{16} - 0x0 \quad = \quad 65536 \tag{4.1}$$

です．FOSC = 32 MHz なので，オーバーフローまでの所要時間は 2.048 ms です．

(6) Enable Timer Interrupt を左クリック

タイマ 1 による割り込み機能を起動します．

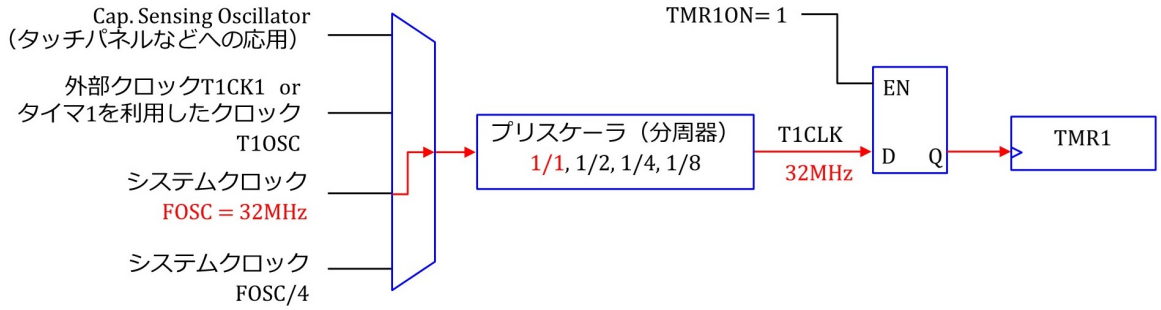


図 4.10: タイマ 1 のクロック源のブロック図

図 4.10 は，タイマ 1 のクロック源（クロックソース）のブロック図（抜粋）です．上記の設定により，タイマ 1 モジュールにおけるクロックの経路が図示のようになります．システムクロック FOSC(= 32 MHz) が取り込まれ，プリスケーラにより 1 倍されます．ENABLE Timer により，TMR1ON = 1 となり，TMR1 レジスタに 32 MHz のクロックが供給されます．

4.2.4 A-D 変換モジュールの設定

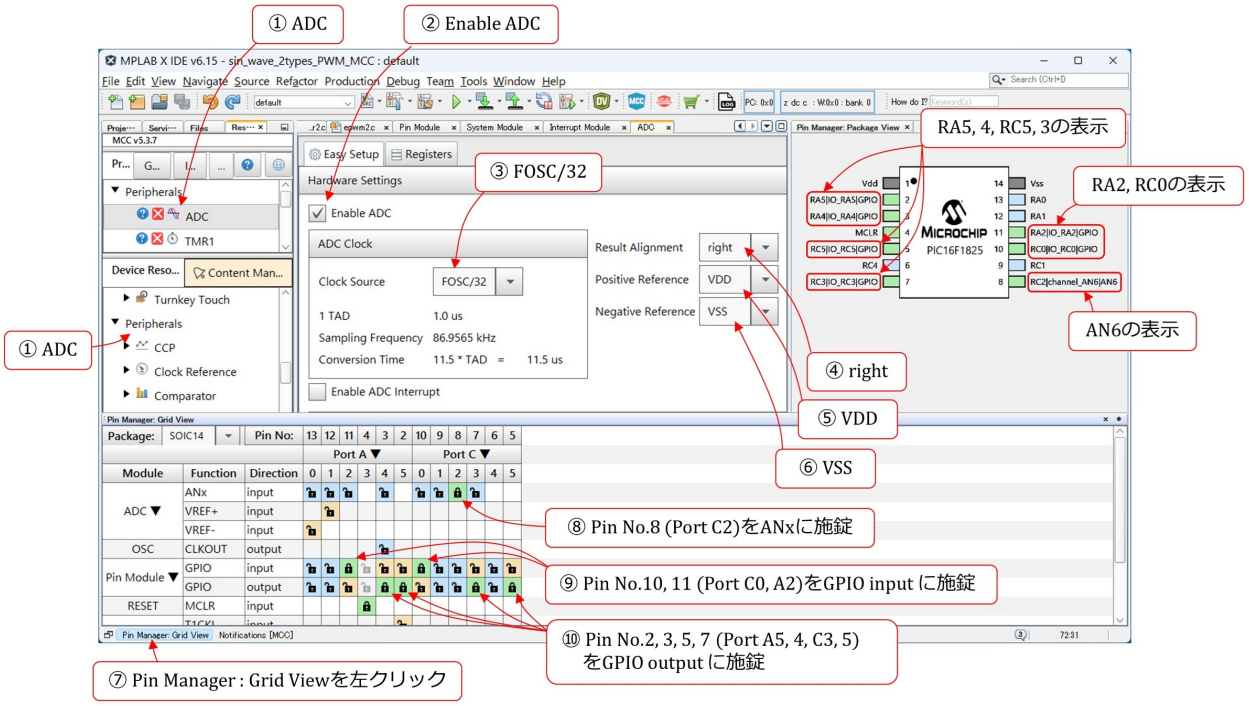


図 4.11: A-D 変換モジュールの設定

図 4.11 のように，画面左中段の Device Resources エリアから ADC を選択すると，画面左上の Project Resources エリアに ADC の文字が表示され，画面中央の Composer エリアに A-D 変換モジュールの設定項目が表示されます．以下の手順で，設定を行います．

(2) Enable ADC を左クリック

A-D 変換モジュールを起動します．

(3) FOSC/32 を選択

A-D 変換モジュールのクロックをシステムクロック FOSC の 1/32 に設定します．データシートによると，PIC16F1825 の A-D 変換モジュールのクロック周期 TAD は，1TAD ≥ 1 [μs] を満たさなければなりません．FOSC $= 32$ MHz なので，FOSC/32 により 1TAD $= 1$ [μs] とします．

(4) right を選択

PIC16F1825 の A-D コンバータは 10 ビットです．right を選択することで，A-D 変換結果は，16 ビットのレジスタに右寄せで得られます．

(5) VDD を選択

A-D 変換の正側の基準値を VDD（1 番ピンの入力電圧値）とします．

(6) VSS を選択

負側の基準値を VSS（14 番ピンの入力電圧値）とします．以上の基準値設定により，A-D コンバータは，VDD − VSS 間の電圧を 10 ビットのディジタル値に変換します．

(7) Pin Manager: Grid View を左クリック

画面左下の Pin Manager: Grid View の文字を左クリックすると，画面下段にピン設定テーブルが表示されます．

(8) Pin No.8(Port C2) の ANx 欄を施錠

8 番ピン (Port C2) をアナログ入力ピンに設定します．ANx 欄の当該箇所を左クリックすると，錠の画が施錠状態に変わります．この変化と同時に，右上のパッケージの 8 番ピンに AN6 の文字が表示されます．

(9) Pin No.10, 11 (Port C0, A2) を GPIO input に施錠

モード設定信号の入力用として，10, 11 番ピンをディジタル入力に設定します．

(10) Pin No.2, 3, 5, 7 (Port A5, 4, C5, 3) を GPIO output に施錠

正弦波電圧出力用として 2, 5 番ピン，可視化 PWM 電圧出力用として 3 番ピン，そして，タイマ 1 による割り込みモニタ信号出力用として 7 番ピンを，ディジタル出力に設定します．右上のパッケージ表示において，それぞれのピンに GPIO の文字が表示されます．

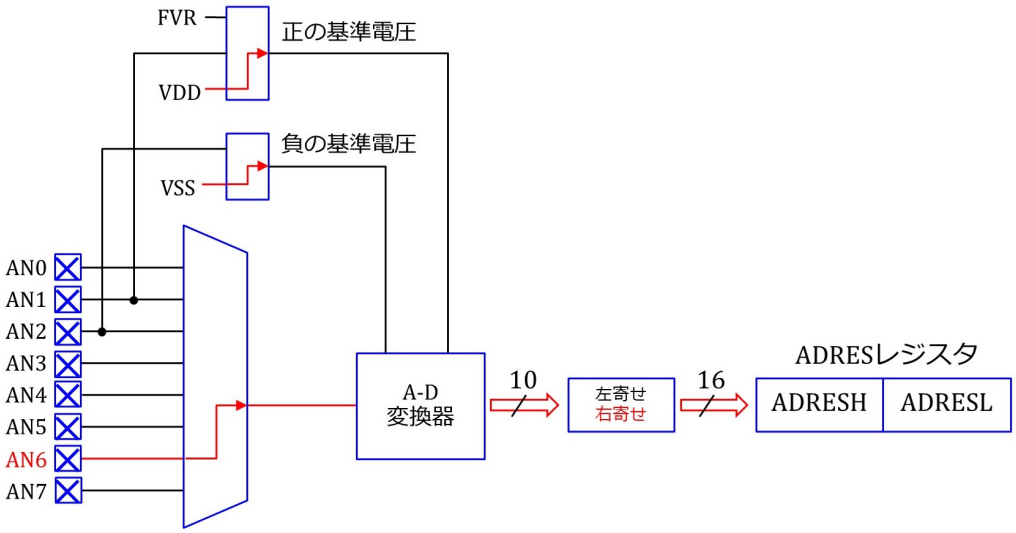


図 4.12: A-D 変換モジュールのブロック図

図 4.12 は，A-D 変換モジュール設定結果のブロック図です．信号／電圧の経路を赤色で示します．A-D 変換器は，AN6 からアナログ電圧を得て，VDD を正の基準電圧，VSS を負の基準電圧として，10 ビットのディジタル値へと変換します．この変換結果が 16 ビットの ADRES レジスタに格納されます．ADRES レジスタは，8 ビットの ADRESH と ADRESL の 2 個のレジスタからなります．right（右寄せ）設定の場合，変換結果の上位 2 ビットが ADREH の下位 2 ビットに格納され，変換結果の下位 8 ビットが ADRESL の全 8 ビットに格納されます．left（左寄せ）の場合，変換結果の上位 8 ビットが ADRESH の全 8 ビットに格納され，変換結果の下位 2 ビットが ADREL の上位 2 ビットに格納されます．

4.2.5 PWM モジュールの設定

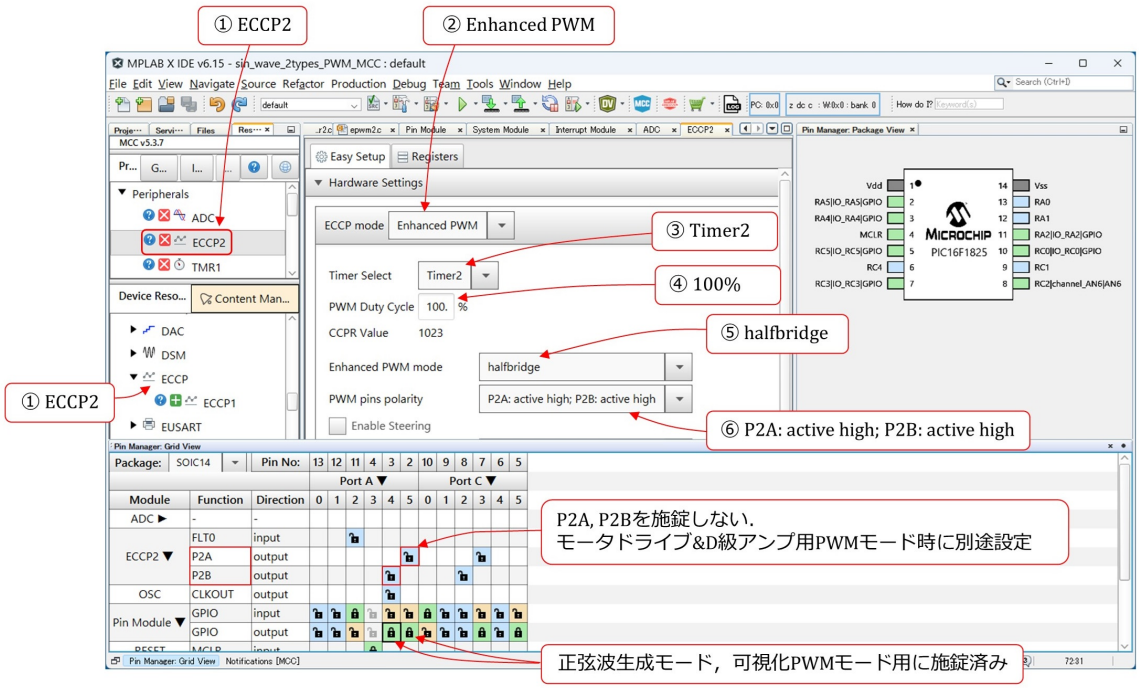


図 4.13: PWM モジュールの設定

図 4.13 は PWM モジュールの設定画面です．Device Resources エリアの ECCP2 を選択し，Composer エリアに ECCP2 モジュールの設定項目を表示します．ECCP2 による PWM 波形の出力ピンは 2 番ピン (P2A) と 3 番ピン (P2B) です．以下の手順で設定します．

(2) Enhanced PWM を選択

ECCP モジュールを PWM モジュールとして使用します．

(3) Timer2 を選択

PWM モジュールのタイムベースをタイマ 2 とします．タイマ 2 は次項 (4.2.6 項) で設定します．

(4) PWM Duty Cycle を 100%に設定

デューティ比 (Duty Cycle) 設定値をとりあえず最大の 100%にします．

図 4.14 は PWM モジュールの動作を示します．横軸は時間 t です．デューティ比設定値は CCPR レジスタ（データシートでは CCPR2L レジスタ）に格納されます．タ

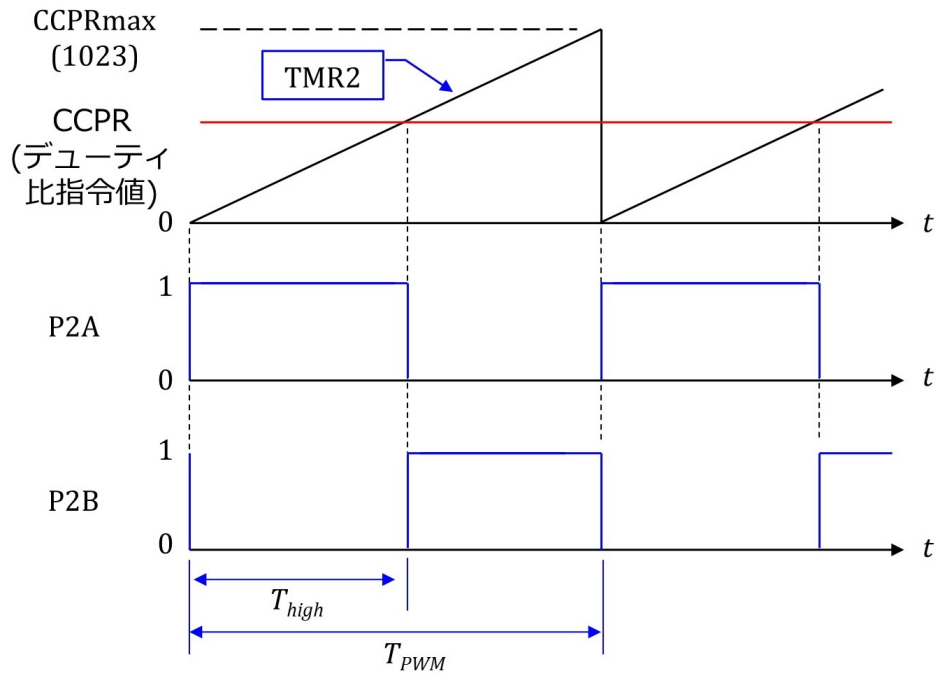


図 4.14: PWM モジュールの動作

イマ 2 モジュール設定 (図 4.15) のステップ 5 にて，タイマ 2 の Timer Period を 32μs に設定すると，CCPR の最大値 CCPRmax（図 4.13 中の CCPR Value）は，

$$\begin{aligned} \text{CCPRmax} &= \text{FOSC} \times \text{TimerPeriod} - 1 \\ &= 1023 \end{aligned} \tag{4.2}$$

となります．FOSC はシステムクロック (32 MHz) です．

TMR2 レジスタの値が FOSC によりカウントアップされて，t とともに増加します．TMR2 と CCPR が比較されて，halfbridge(図 4.13 のステップ 5 にて選択) の場合，

TMR2 $\leq$ CCPR のとき P2A $= 1$, P2B $= 0$

TMR2 $>$ CCPR のとき P2A $= 0$, P2B $= 1$

となります．そして，

TMR2 = CCPRmax + 1 となった瞬間に TMR2 = 0

とリセットされます．CCPR により P2A, P2B のパルス幅を制御する，PWM(Pulse Width Modulation) 制御法です．TMR2 のリセット間隔が PWM 周期 T_{PWM} です．

$$\begin{aligned} T_{PWM} &= \text{TimerPeriod} \\ &= 32\mu\text{s} \end{aligned} \tag{4.3}$$

です．

デューティ比 δ は，P1A = 1 の期間を T_{high} とすると，

$$\delta = \frac{T_{high}}{T_{PWM}} \tag{4.4}$$

と定義されます．CCPR により，δ は

$$\delta = \frac{\text{CCPR}}{\text{CCPRmax}} \tag{4.5}$$

と決定されます．

CCPRmax = 1023 は 2 進数 10 ビットに相当するので，10 ビットの A-D 変換結果をそのまま CCPR に格納することで，デューティ比を 0 ～ 100% の範囲で変えられます．

(5) halfbridge を選択

P2A, P2B を使用します．

(6) P2A: active high; P2B: active high を選択

この設定と halfbridge 設定により，P2A, P2B は図 4.14 のように相補的に動作します．

(7) P2A, P2B に施錠しない

P2A, P2B は施錠しないでおきます．これは，図 2.14 のように，2 番ピンを正弦波生成モード，3 番ピンを可視化 PWM モードのディジタル出力とすることと重複するためです．

ディジタル出力ピンを 2, 3 番ピン以外に選べば，重複は避けられます．「パワーエレクトロニクスノート II」にて，この重複使用を選んでいます．本稿はこの拙稿の副読本です．そこで，本稿では，重複使用はそのままにして，図 4.20 のシステム初期設定関数における，モード切り替えの際に，2, 3 番ピンを P2A, P2B に割り当てます．

4.2.6 タイマ 2 モジュールの設定

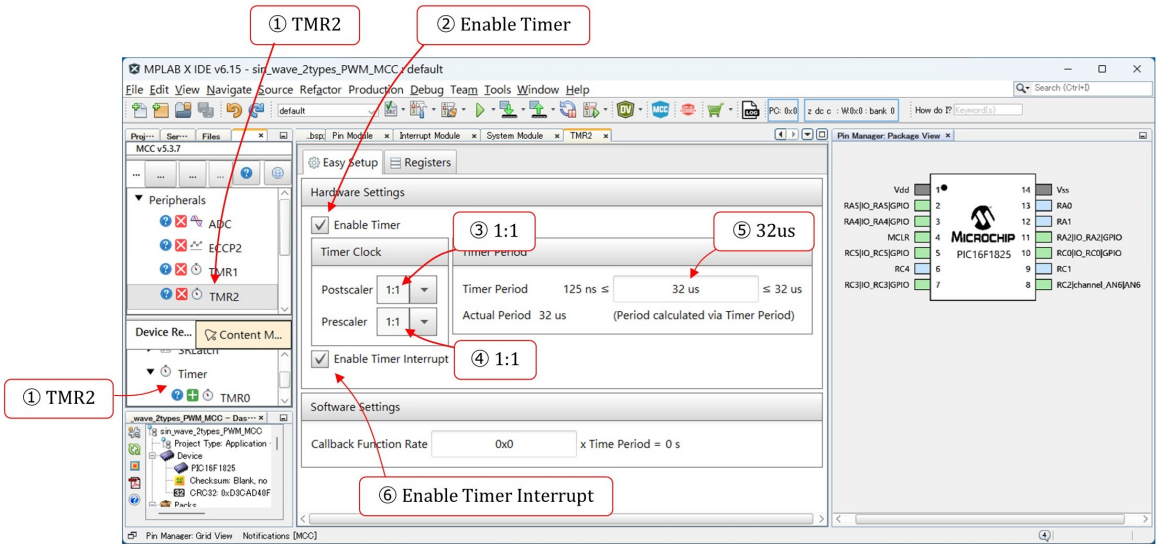


図 4.15: タイマ 2 モジュールの設定

図 4.15 はタイマ 2 モジュールの設定画面です．Device Resources エリアの TRM2 を選択し，Composer エリアにタイマ 2 モジュールの設定項目を表示します．タイマ 2 は，前項の PWM モジュールのタイムベースとして使います．以下の手順で設定します．

(2) Enable Timer を左クリック

タイマ 2 モジュールを起動します．

(3) 1:1 を選択

クロックを 1/1 に分周します．なお，データシートによると，Postscaler はタイマ 2 による割り込み信号用です．本稿の設定では使いません．

(4) 1:1 を選択

クロックを 1/1 に分周します．PWM モジュールのタイムベースとして指定すると，タイマ 2 には 2 ビットが追加されて 10 ビットタイマとなり，そのクロックが FOSC となります．

(5) 32 us を設定

タイマ 2 は，32[μs] の時点でリセットされ，再び 0 からカウントアップすることを繰り返します．PWM モジュールのタイムベースとして使われると，この時間が図 4.14

の PWM 周期 T_{PWM} となります．以上の設定により，図 4.13 の CCPR value が 1023 になります．

4.2.7 周辺モジュール設定関数の生成

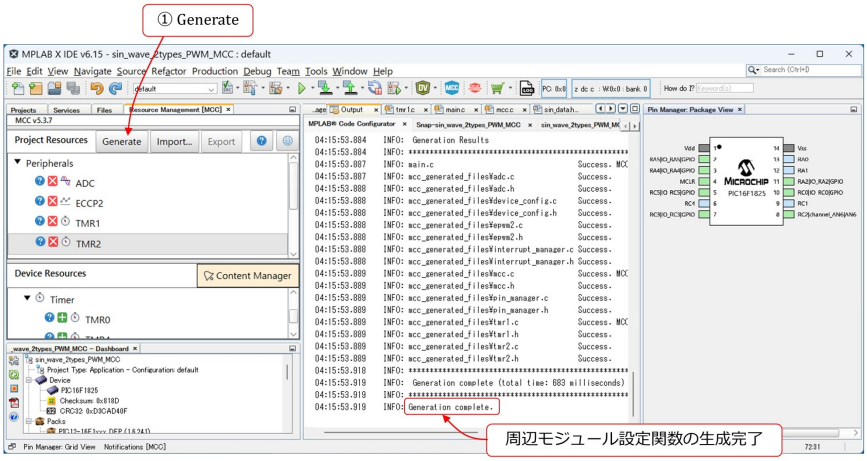


図 4.16: 周辺モジュール設定関数の生成

以上で周辺モジュール設定が終了です．図 4.16 のように，画面左上の Generate ボタンを左クリックし，画面中央に Generation complete の文字が表示されれば，周辺モジュール設定関数の生成完了です．

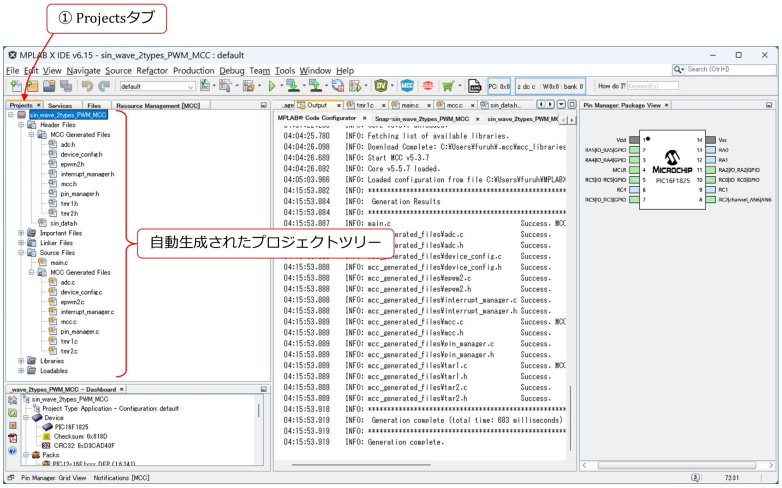


図 4.17: MCC により自動生成されたプロジェクトツリー

図 4.17 は MCC により自動生成されたプロジェクトツリーです．画面左上の Project タブを左クリックすることで，このツリーを表示できます．

4.3 追加の設定

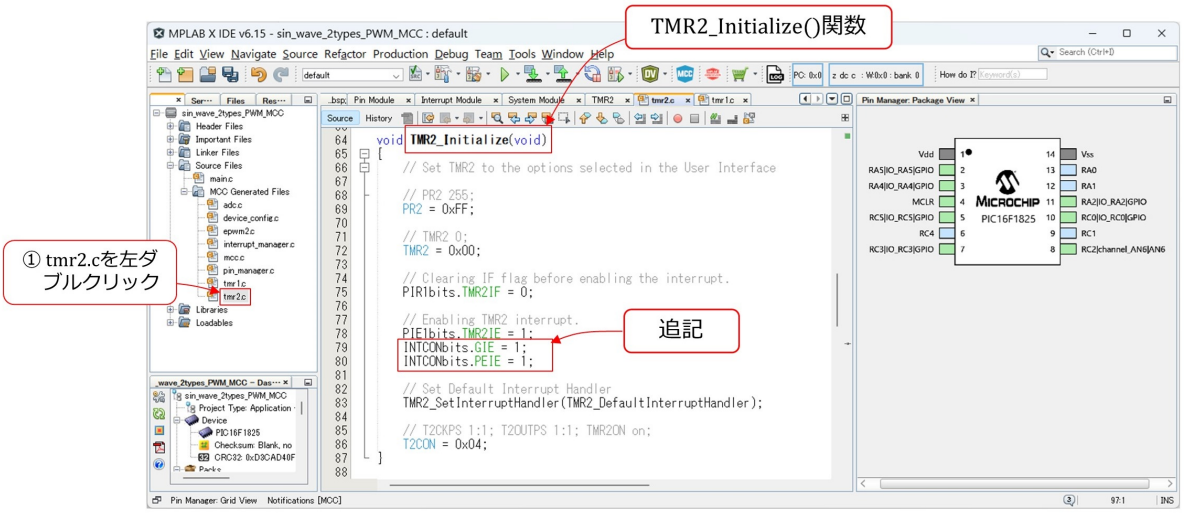


図 4.18: TMR1_Initialize() 関数追加の設定 (GIE, PEIE)

タイマ 1 モジュールでは，追加で GIE(Global Interrupt Enable) ビットと PEIE(PEripheral Interrupt Enable) ビットを “1” に設定しなければなりません．図 4.18 は，GIE と PEIE を 1 に設定するコードとその追加箇所を示します．プロジェクトツリー内の tmr1.c ファイルを左ダブルクリックします．すると，Composer エリアに同ファイルが展開されます．スクロールすると TMR1_Initialize() 関数が現れます．PIE1bits.TMR1IE =1 の下に

$$\begin{aligned} \text{INTCONbits.GIE} &= 1; \\ \text{INTCONbits.PEIE} &= 1; \end{aligned} \tag{4.6}$$

を追記します．GIE ビットと PEIE ビットは INTCON レジスタ内にあります．GIE は PEIE を包含し，PEIE は TMR1IE を包含します．TMR1 による割り込みを有効にするには，それを包含する PEIE と GIE を “1” にする必要があります．

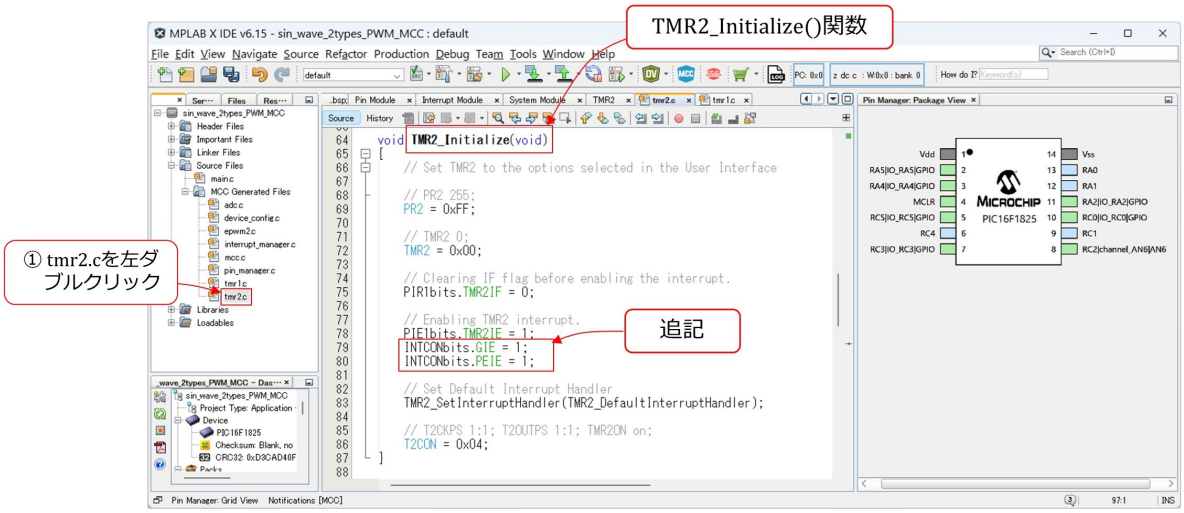


図 4.19: TMR2_Initialize() 関数追加の設定 (GIE, PEIE)

タイマ 2 モジュール関数においても，図 4.19 のように GIE ビットと PEIE ビットを “1” に設定する必要があります．タイマ 1 モジュールは正弦波生成モード，可視化 PWM モードのときに起動し，これらモードのプログラムはタイマ 1 割り込み処理関数内に記述します．また，タイマ 2 モジュールはモータドライブ&D 級アンプ用 PWM モードのときに起動し，このモードのプログラムをタイマ 2 割り込み処理関数内に記述します．

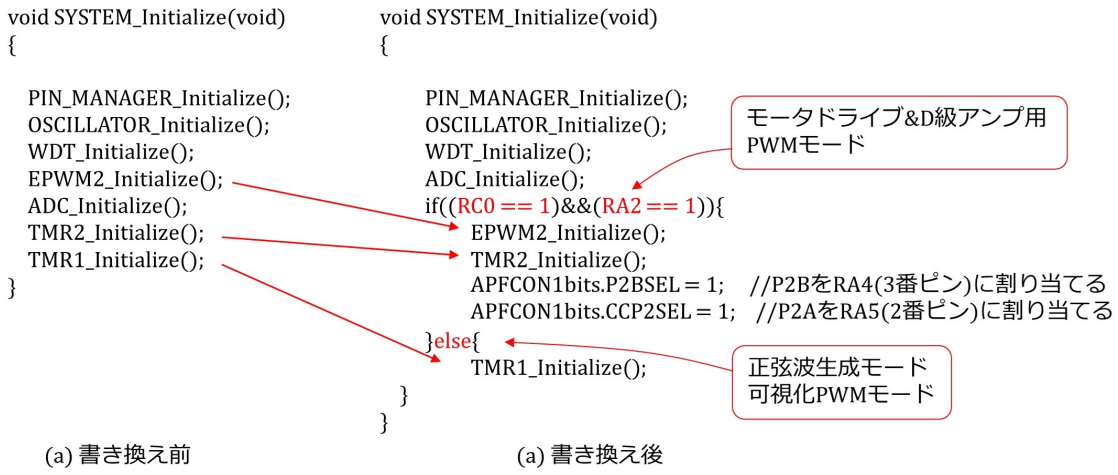


図 4.20: システム初期設定関数の書き換え

図 4.20 は，システム初期設定関数の書き換えを行っている画面です．(a) は書き換え前，(b) は書き換え後です．プロジェクトツリー内の mcc.c ファイルを左ダブルクリックして，

同ファイルを画面中央に開きます．スクロールすると，図 4.20(a) の SYSTEM_Initialize() 関数が現れます．これは MCC により自動生成された関数です．この関数を，正弦波生成モード・可視化 PWM モードとモータドライブ&D 級アンプ用 PWM モードで場合分けをします．

図 2.14 より，RC0 = 1, RA2 = 1 のときが，モータドライブ&D 級アンプ用 PWM モードです．このモードでは，図 4.20(b) のように，PWM モジュールとタイマ 2 モジュールを起動し，P2A を RA5(2 番ピン) に割り当て，P2B を RA4(3 番ピン) に割り当てます．APFCON1 レジスタについては，データシートを参照してください．

RC0 = 0, RA2 = 0 のときが正弦波生成モード，そして，RC0 = 0, RA2 = 1 のときが可視化 PWM モードです．これらのモードでは，タイマ 1 モジュールを起動します．RA4(3 番ピン)，RA5(2 番ピン) は図 4.11 にて，ディジタル出力に設定済みです．

4.4 モータドライブ&D 級アンプ用 PWM モード

4.4.1 TMR2_DefaultInterruptHandler() 関数

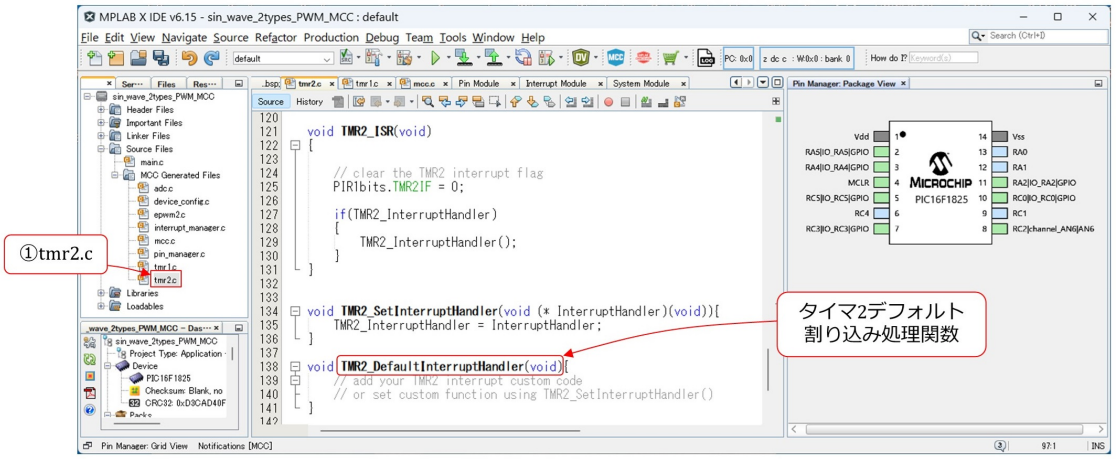


図 4.21: tmr2.c ファイル内に生成されたタイマ 2 デフォルト割り込み処理関数

図 4.21 は MCC により tmr2.c ファイル内に生成されたタイマ 2 デフォルト割り込み処理関数 TMR2_DefaultInterruptHandler() です．

タイマ 2 は，図 4.13 のステップ 3 にて，PWM モジュールのタイムベースに選定しています．TMR2 レジスタの値は，図 4.14 のように，CCPRmax の値に達すると 0 にリセットされます．このリセットのタイミングで TMR2_DefaultInterruptHandler() 関数が起動されます．タイマ 2 による割り込み周期 T_{2INT} は PWM 周期 T_{PWM} と同期が取れています．

4.4.2 割り込み処理関数の main.c への移動とプログラムの記述

```
 adc_result_t   ad_out;                    // A-D変換結果

// モータドライブ＆D級アンプ用PWMモード
void TMR2_DefaultInterruptHandler(void)
{
   RC3 = 1;                               // ポートCのRC3(7番ピン)に1を出力する．割り込み発生のモニタリング用

   ad_out = ADC_GetConversion(0x06);      // A-D変換結果の取得
   EPWM2_LoadDutyValue(ad_out);           // PWMデューティ比の設定

   RC3 = 0;                               // ポートCのRC3(7番ピン)に0を出力する．割り込み発生のモニタリング用
}
```

図 4.22: main.c ファイル内に移動させたタイマ 2 デフォルト割り込み処理関数とモード用プログラム

図 4.22 は，main.c ファイル内に移動させたタイマ 2 デフォルト割り込み処理関数です．この関数内にモータドライブ&D 級アンプ用 PWM モードのコードを記述してあります．ただし，変数 ad_out の型宣言は，グローバル変数とするために

```
adc_result_t    ad_out;
void TMR2_DefaultInterruptHandler(void)
```

と，割り込み処理関数の外で行っています．adc_result_t は，符号無し 16 ビット整数型です．MCC により自動生成された adc.h ファイル内で定義されています．uint16_t と同じです．

```
RC3 = 1;
```

```
RC3 = 0;
```

は，割り込み発生のモニタリングのために，RC3(7 番ピン) に “High”，“Low” の電圧を出力します．

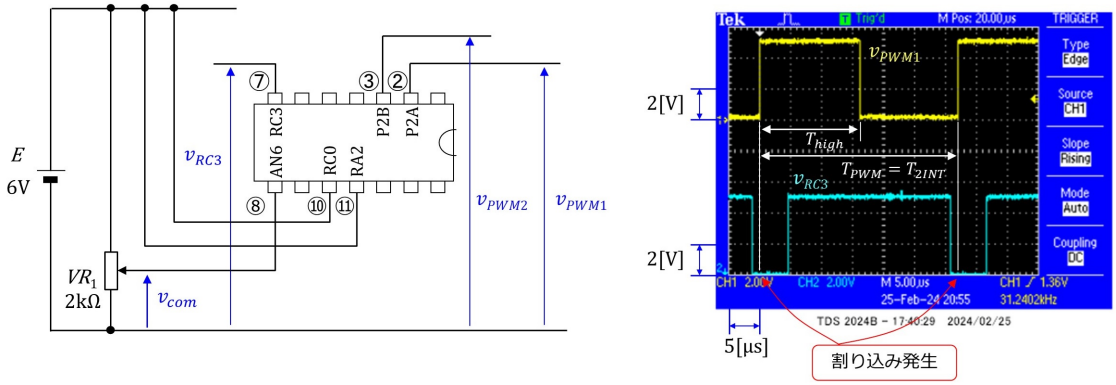
```
ad_out = ADC_GetConversion(0x06);
```

により，A-D 変換の入力チャネルを 0x06(AN6, 8 番ピン) に指定して，A-D 変換実行→

待機→結果取得→ ad_out 格納を行います．

```
EPWM2_LoadDutyValue(ad_out);
```

により，デューティ比を設定します．この関数は，MCC により epwm2.c ファイル内に生成されています．



(a) マイコンの配線　　(b) 動作波形例

図 4.23: モータドライブ&D 級アンプ用 PWM モードの配線と動作波形例

図 4.23 は，モータドライブ&D 級アンプ用 PWM モードのときのマイコン配線と動作波形例です．マイコンは，8 番ピンから指令電圧 v_{com} を読み込んで ad_out に変換し，この ad_out によりデューティ比を設定し，2 番ピン (P2A), 3 番ピン (P2B) に PWM 電圧 v_{PWM1}, v_{PWM2} を出力します．7 番ピンからは，モニタ電圧 V_{RC3} を出力します．

同図 (b) の黄色線は v_{PWM1}，水色線は v_{RC3} です．図 4.14 の PWM モジュールの動作波形より，$v_{PWM1} \approx 5$V の期間が T_{high}, v_{PWM1} の 1 周期が T_{PWM} です．また，v_{PWM1} の立ち上がりのタイミングでタイマ 2 割り込みが発生し，TMR2_DefaultInterruptHandler() 関数が起動されます．タイマ 2 による割り込み周期 $T_{2INT} = T_{PWM}$ です．

指令電圧 v_{com} を変えたときのデューティ比の変化例は図 2.16 に示してあります．

4.5 正弦波生成モード

4.5.1 TMR1_ISR() 関数と TMR1_DefaultInterruptHandler() 関数

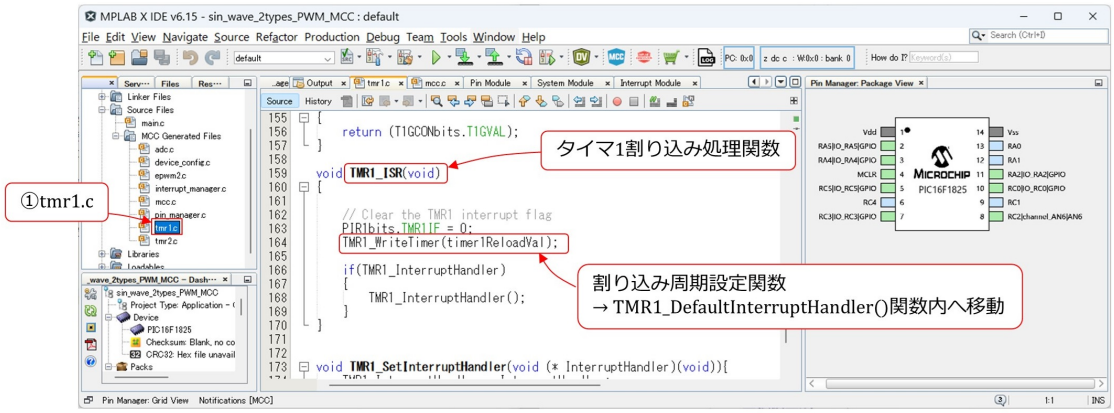


図 4.24: tmr1.c ファイル内に生成されたタイマ 1 割り込み処理関数

図 4.24 は MCC により tmr1.c ファイル内に生成されたタイマ 1 割り込み処理関数 TMR1_ISR() です．図 4.9 のステップ 5 で設定した周期 (T_{1INT} とします）ごとに，この関数が起動されます．そして，この関数が同じ tmr1.c ファイル内にある

```
TMR1_DefaultInterruptHandler()
```

関数を起動します．正弦波生成モードのプログラムは，TMR1_DefaultInterruptHandler() 関数を main.c ファイルに移動させて，その関数内に記述します．

正弦波生成モードでは，図 2.12 のように，8 番ピンから指令電圧 v_{com} を読み込んで，出力正弦波電圧の周波数を変えます．この周波数変化は，図 4.24 中の割り込み周期設定関数

```
TMR1_WriteTimer(timer1ReloadVal);
```

により，T_{1INT} を変えることで実現できます．そこで，この TMR1_WriteTimer() 関数を，main.c ファイル内の TMR1_DefaultInterruptHandler() 関数内へ移動させます．

4.5.2 割り込み処理関数の main.c への移動とプログラムの記述

```
void TMR1_DefaultInterruptHandler(void)
{
    RC3 = 1;                                  // ポートCのRC3(7番ピン)に1を出力する.
                                              //割り込み発生のモニタリング用

    ad_out = ADC_GetConversion(0x06);  // A-D変換結果の取得

    // 正弦波生成モード
    if((RC0 == 0) && (RA2 == 0)){
        ・・・
    }
    // 可視化PWMモード
    else if((RC0 == 0) && (RA2 == 1)){
        ・・・
    }

    RC3 = 0; ;                                // ポートCのRC3(7番ピン)に0を出力する.
}
```

指令電圧v_{com}の読み込み

正弦波生成モード用プログラム

可視化PWMモード用プログラム

図 4.25: main.c ファイル内に移動させたタイマ 1 デフォルト割り込み処理関数の基本構成

図 4.25 は main.c ファイル内に移動させたタイマ 1 デフォルト割り込み処理関数の基本構成を示します.

$$\mathrm{RC3} = 1;$$

$$\mathrm{RC3} = 0;$$

により，割り込み発生のモニタリング用信号を出力します.

$$\mathrm{ad\_out} = \mathrm{ADC\_GetConversion}(0\mathrm{x}06);$$

により，8 番ピンから指令電圧 v_{com} を読み込んで，A-D 変換結果を ad_out に格納します.

図 2.14 より，正弦波生成モードは

$$\mathrm{RC0} = 0,\ \mathrm{RA2} = 0$$

の場合に起動されるようにします．また，可視化 PWM モードは

$$RC0 = 0,\ RA2 = 1$$

の場合に起動されるようにします．

```
#include "sin_data.h“                          ← 正弦波データのヘッダファイル

// 正弦波生成モード用定数                                                    ┐
#define   sin_halfT          120      // 1/2周期のデータ数                  │
#define   sin_fullT          240      // 1周期のデータ数                    ├ 定数宣言
#define   pwm_step_max       256      // PWM1周期のステップ数               │
#define   T1INTmax           2080     // タイマ1割り込み周期の最長時間      ┘

uint16_t        pwm_step = 0;          // PWMのデューティ比決定用カウンタ    ┐
uint8_t         sin_step = 0;          // 正弦波データの番地カウンタ         ├ グローバル変数宣言
adc_result_t    ad_out;                // A-D変換結果                        ┘

void  TMR1_DefaultInterruptHandler(void)
{     uint16_t              Int_Data, Duty;                                 ┐ ローカル変数宣言
      ad_out = ADC_GetConversion(0x06);  // A-D変換結果の取得
                                                ← 正弦波生成モードへの分岐
      if((RC0 == 0) && (RA2 == 0)){
            Int_Data = 0xFFFF - (T1INTmax - ad_out*2);// T1INTの決定          ┐ タイマ1割り込み周期決定
            TMR1_WriteTimer(Int_Data);     // タイマ1のカウントアップ値の設定  ┘

            Duty = sin_data[sin_step];     // 正弦波データによりデューティ比決定
            if(pwm_step <= Duty){          // PWM波形生成                     ┐
                  if(sin_step < sin_halfT){ // 正の半波                      │
                        RA5 = 1;           // 2番ピン                        │
                  }else{                   // 負の半波                      │
                        RC5 = 1;           // 5番ピン                        ├ PWM波形生成
                  }                                                          │
            }else{                                                           │
                  RA5 = 0;                                                   │
                  RC5 = 0;                                                   │
            }                                                                ┘

            pwm_step = (pwm_step + 1) % pwm_step_max;                        ┐
            if(pwm_step == 0) {                                              ├ T1INTごとにpwm_step更新
                  sin_step = (sin_step + 1) % sin_fullT;                     ├ PWM1周期で sin_stepを1回更新
            }                                                                ┘
      }
```

図 4.26: 正弦波生成モードのプログラム

図 4.26 は，正弦波生成モードのプログラムです．このモードに関係するコードを抜粋しました．タイマ 1 割り込み周期は

```
#define     T1INTmax     2080
ad_out = ADC_GetConversion(0x06);
```

```
Int_Data = 0xFFFF - (T1INTmax - ad_out*2);
TMR1_WriteTimer(Int_Data);
```

により設定します．

ad_out は 10 ビット値なので，電圧指令値 v_{com} により，0 ～ 1023 の範囲を変化します．したがって，Int_Data は 0xFFFF − 2080 ～ 0xFFFF − 34 の範囲を変化します．TMR1_WriteTimer(Int_Data) 関数は，TMR1 レジスタに Int_Data の値を格納します．

TMR1 レジスタはシステムクロック FOSC によりカウントアップされて，オーバフローにより割り込みを発生します．よって，タイマ 1 割り込み周期 T_{1INT} は

$$T_{1INT} = \frac{\text{T1INTmax} - 2 \times \text{ad\_out} + 1 + \alpha}{\text{FOSC}} \tag{4.7}$$

です．α は主に ADC_GetConversion() 関数の実行ステップ数です．$\alpha = 0$ と仮定すると，ad_out = 0 のとき，T_{1INT} は最大となり，

$$\begin{aligned} T_{1INT}max &= \frac{2080 + 1}{32\ [\text{MHz}]} \\ &\approx 65\ [\mu\text{s}] \end{aligned} \tag{4.8}$$

です．正弦波の 1 周期を T_{sin} とすると，後述の (4.12) 式より

$$\begin{aligned} T_{sin}max &= 61440 \times T_{1INT}max \\ &\approx 4\ [\text{s}] \end{aligned} \tag{4.9}$$

です．実際は約 5.8 s でした．

```
#include “sin_data.h”
```

は，正弦波データのヘッダファイルです．

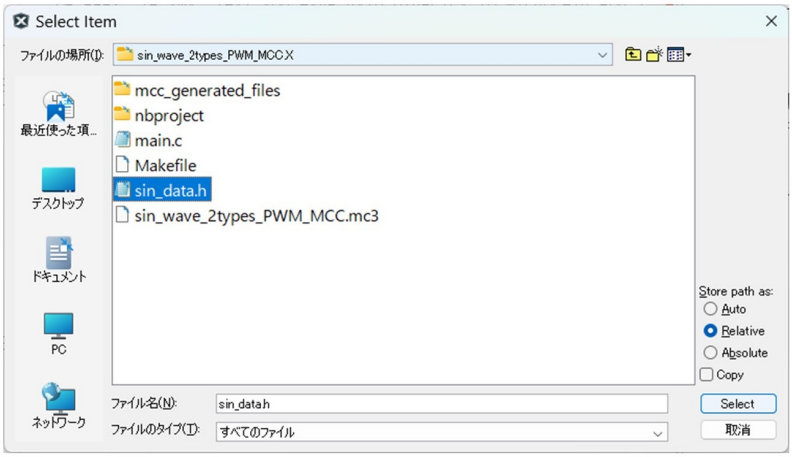


図 4.27: sin_data.h ファイルの追加

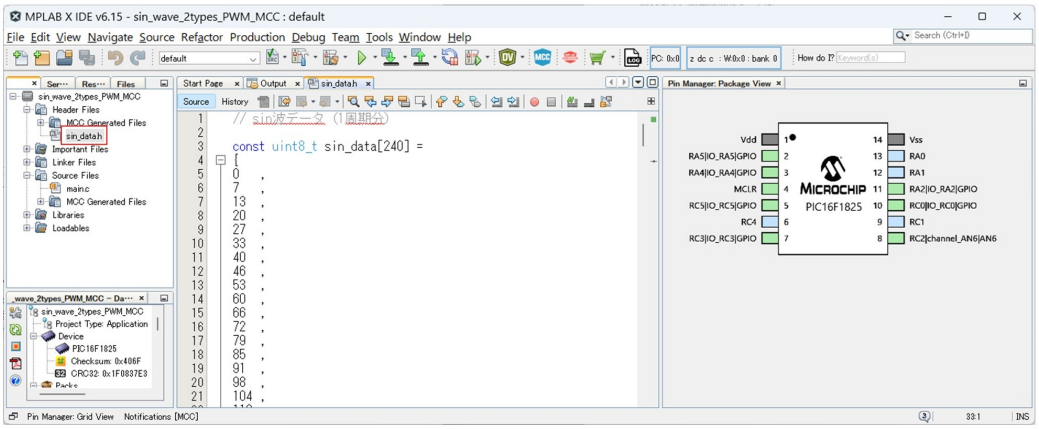


図 4.28: Header Files に追加された sin_data.h ファイル

別途作成した sin_data.h ファイルを main.c と同じフォルダに置いておきます．プロジェクトツリー内の Header Files を右クリックして，プルダウンメニューから Add Existing Item を選択すると図 4.27 のウィンドウが現れます．sin_data.h ファイルを選択すると，図 4.28 のように，Header Files に sin_data.h ファイルが追加されます．sin_data.h の文字を左ダブルクリックすると，画面中央に同ファイルの内容が表示されます．sin_data[] の値は

$$\text{sin\_data} = 255 \times |\sin\theta| \tag{4.10}$$

を用いて，1 周期 2π を 240 分割して，$\theta = 0, 2\pi/240, 2 \times 2\pi/240, \cdots$ の 240 点における値を四捨五入して得ました．

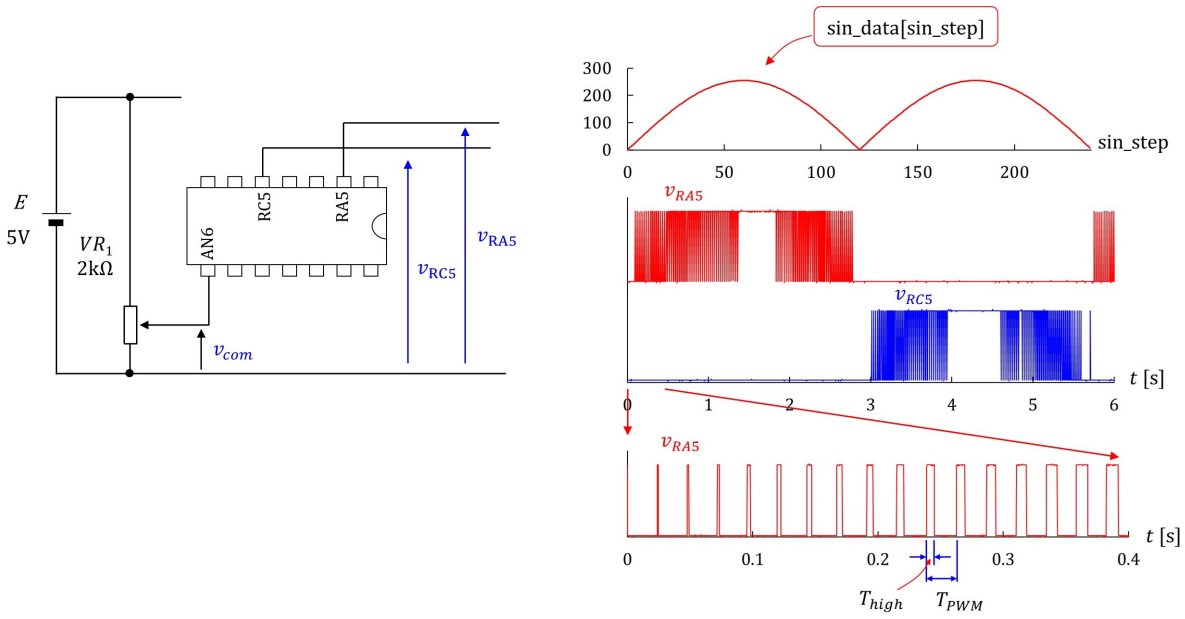


図 4.29: 正弦波生成モードの動作

図 4.29 は正弦波生成モードの動作を示します．左側の回路図において，RA5(2 番ピン) の出力電圧を v_{RA5}，RC5(5 番ピン) の出力電圧を v_{RC5} とします．右側の v_{RA5}, v_{RC5} は実験波形例です．横軸は時間 t です．これらは PWM 波形です．上側に，対応する正弦波データ sin_data を，横軸を合わせて描いてあります．この波形の横軸は，要素番号 sin_step で，0 ～ 239 の範囲で変化します．正弦波生成モードのプログラムは，PWM 周期 T_{PWM} (この例では約 24 ms) ごとに sin_step を 1 ずつ増やして，

$$\text{Duty} = \text{sin\_data}[\text{sin\_step}];$$

により，デューティ比 Duty を設定します．sin_data[] は (4.10) 式より 0 ～ 255 の範囲で変化します．v_{RA5}, v_{RC5} はそれぞれ正の半波，負の半波の正弦波を PWM 波形で実現しています．

v_{RA5}, v_{RC5} はオシロスコープで計測しました．細すぎるパルスはオシロスコープで捉えられていません．最下段には，オシロスコープの時間軸を拡大して計測した v_{RA5} の実験波形を示します．時間軸を拡大したことで，左端の最も細いパルスも捉えられています．v_{RA5} = “high” の期間を T_{high} とし，v_{RA5} の立ち上がり時点間を T_{PWM} とします．

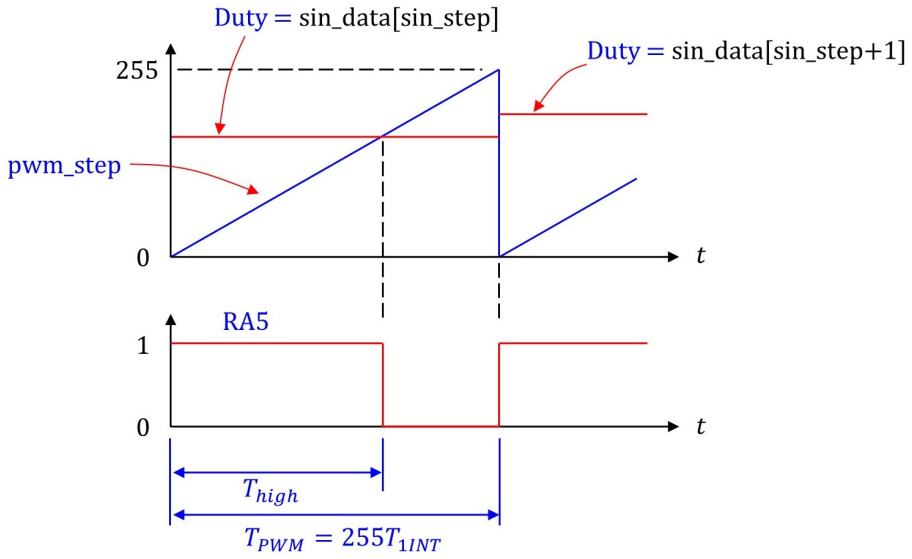


図 4.30: PWM 波形生成プログラムの動作

図 4.30 は PWM 波形生成プログラムの動作を示します．PWM の 1 周期分の波形例です．pwm_step は，タイマ 1 割り込み周期 T_{1INT} ごとに 1 ずつ増やされて，256 に達した瞬間に 0 にリセットされます．変化範囲は 0 ～ 255 です．PWM 周期 T_{PWM} は

$$T_{PWM} = 256 \times T_{1INT} \tag{4.11}$$

です．

sin_data[sin_step] の要素番号 sin_step は，pwm_step が 0 にリセットされたときに，1 増やされます．正弦波の 1 周期を T_{sin} とすると

$$\begin{aligned} T_{sin} &= 240 \times T_{PWM} \\ &= 240 \times 256 \times T_{1INT} \end{aligned} \tag{4.12}$$

の関係があります．

PWM 波形生成プログラムは，pwm_step と Duty を比較して，

pwm_step $\leq$ Duty のとき RA5 = 1 (正の半波) or RC5 = 1 (負の半波)

pwm_step $>$ Duty のとき RA5 = RC5 = 0

と設定します．

これより，デューティ比 δ と sin_data[] の関係は

$$\begin{aligned} \delta &= \frac{T_{high}}{T_{PWM}} \\ &= \frac{\text{Duty}}{255} \\ &= \frac{\text{sin\_data[sin\_step]}}{255} \end{aligned} \tag{4.13}$$

となります．(4.10) 式より，sin_data[] が 0 ～ 255 の範囲で変化するので，δ は 0 ～ 1 の範囲で変化します．

4.6 可視化 PWM モード

```
// 可視化PWMモード
else if((RC0 == 0) && (RA2 == 1)){

    TMR1_WriteTimer(0x0);          // タイマ1のカウントアップ値の設定.
                                   // オーバフローで割り込みを発生
                                   // f1INT = 32MHz/(0xFFFF+α) = 約500Hz
                                   // fPWM500Hz/1024 = 約0.5Hz
    if(pwm_step <= ad_out){
        RA4 = 1;
    }else{
        RA4 = 0;
    }

    pwm_step += 1;                 // PWMステップ +1

    if(pwm_step >= 1024){          // PWMステップ数最大値　1024
        pwm_step = 0;
    }
}
```

図 4.31: 可視化 PWM モードのプログラム

図 4.31 は，可視化 PWM モードのプログラムです．

TMR1_WriteTimer(0x0);

によりタイマ 1 割り込み周期 T_{1INT} を設定します．

$$\begin{aligned} T_{1INT} &= \frac{0x10000 + \alpha}{32[\text{MHz}]} \\ &\approx 2\ [\text{ms}] \end{aligned} \tag{4.14}$$

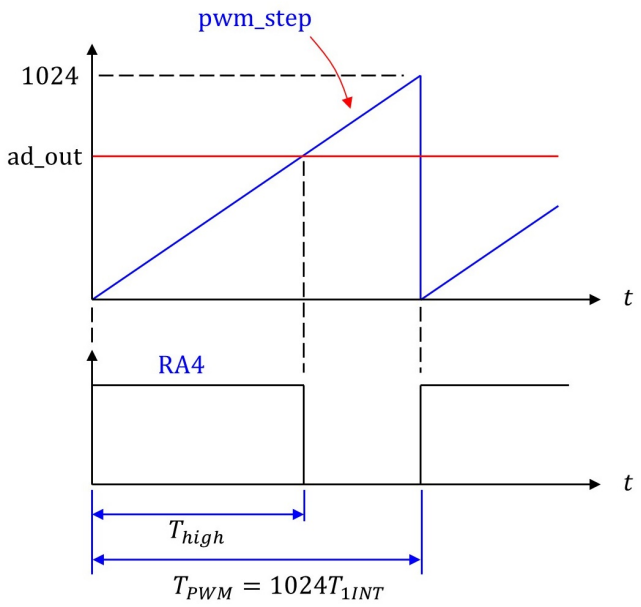


図 4.32: 可視化 PWM モードの動作

です.

図 4.32 は可視化 PWM モードの動作を示します. pwm_step を ad_out と比較して,

pwm_step $\leq$ ad_out のとき　RA4 = 1
pwm_step $>$ ad_out のとき　RA4 = 0

としています. pwm_step, ad_out のいずれも変化範囲は 0 ～ 1023 です. デューティ比 δ は 0 ～ 1 の範囲を変化します. PWM 周期 T_{PWM} は

$$\begin{aligned} T_{PWM} &= 1024 \times T_{1INT} \\ &\approx 2 \text{ [s]} \end{aligned} \tag{4.15}$$

です. 実際は 2.1 S でした.

索引

参考文献

[1] 古橋武「パワーエレクトロニクスノート」コロナ社，2008.

[2] 古橋武「パワーエレクトロニクスノート II: 製作演習付き講義の実践記録」Kindle 本，Amazon

[3] パワーエレクトロニクスノート

[4] モータドライブノート

著者
古橋　武
名古屋大学名誉教授（令和 2 年 4 月より）
furuhashi.takeshi*

*に @gmail.com を付けてください.