パワーエレクトロニクスノート
ーPIC16F1825による正弦波発生器，PWM信号発生器ー

本稿掲載のWebページ

http://mybook-pub-site.sakura.ne.jp/Motor_Drive_note/

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 ー製作演習付き講義の実践記録ー」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.

本稿では各種モジュールの設定関数と専用の用語を使用しています．これら関数と用語は

モータドライブノート　I. PIC16F1825 による DC モータの回転数制御

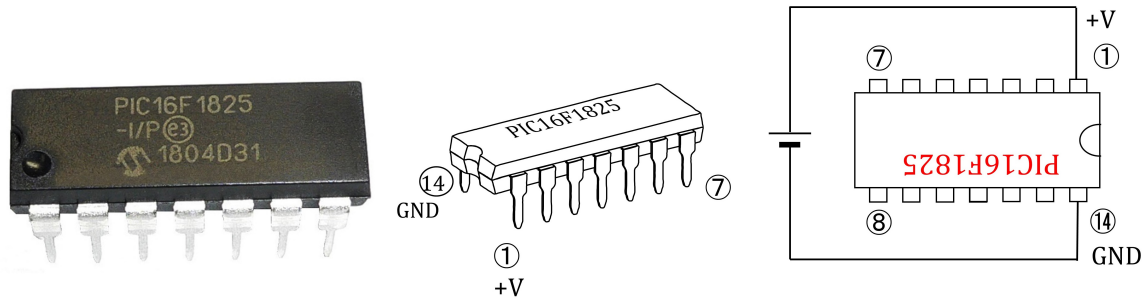
に詳述してあります．この pdf ファイルをダウンロードして，本稿と一緒にお読み下さい.

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) 立体図　　(c) 電源の接続

図 2.1: PIC16F1825

Fig. 2.1 は PIC16F1825 の外観，立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) の立体図の通り，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．

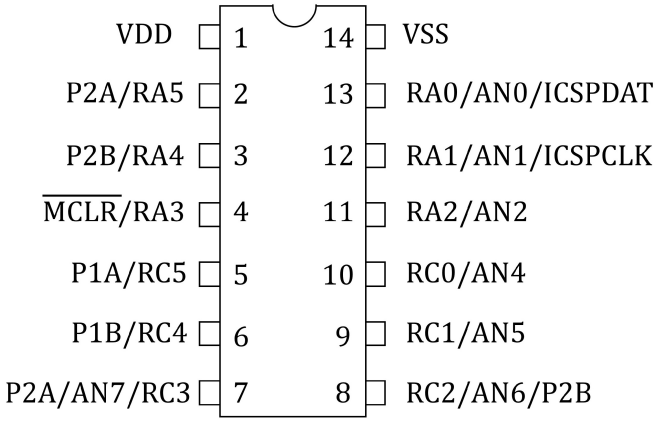


図　2.2: PIC16F1825 のピン配置

Fig. 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．

Fig. 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．抵抗器の全面に a-c 間の抵抗値が印字されてい

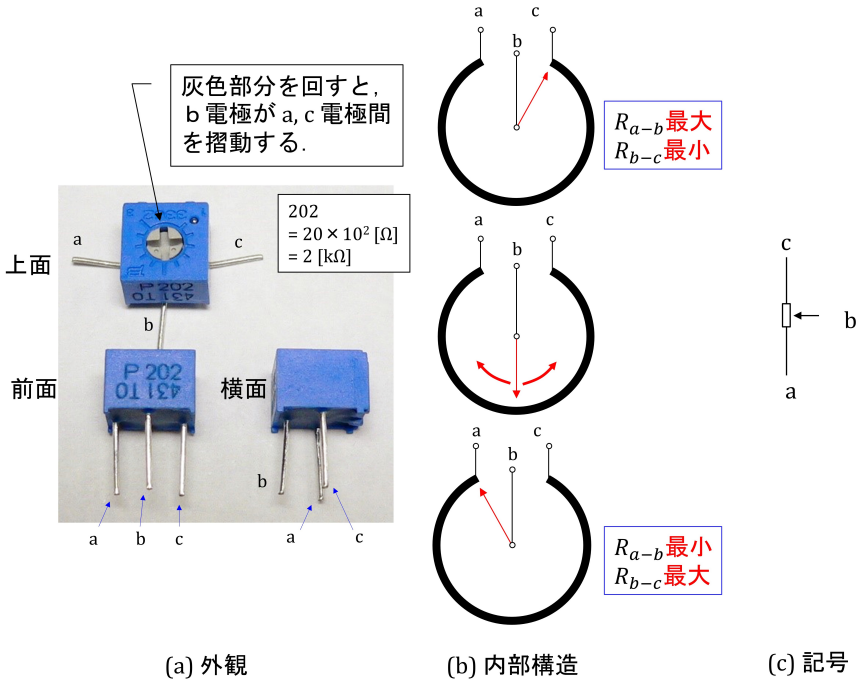


図 2.3: 可変抵抗器

ます．写真の例は 202 です．この数値の意味は

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

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．

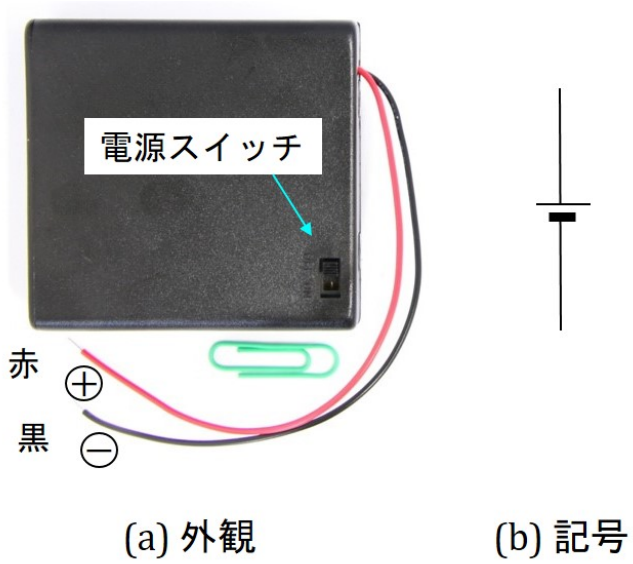


図　2.4: 電池ボックス

Fig. 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．

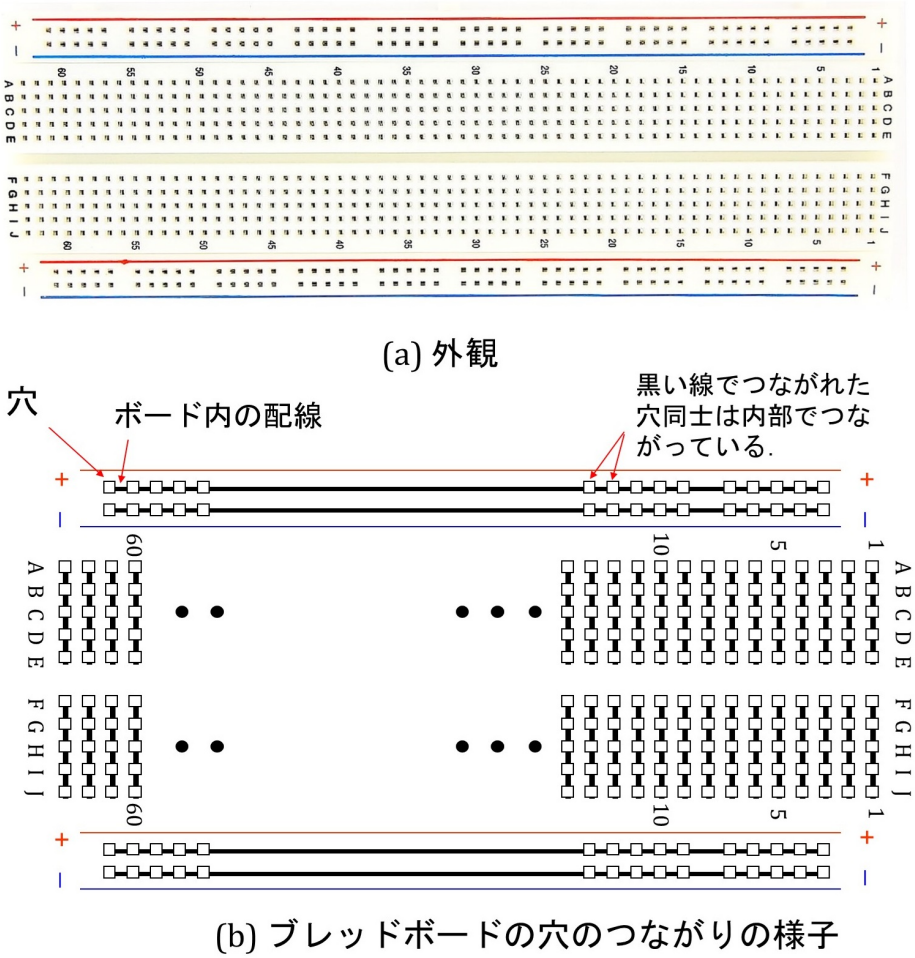


図 2.5: ブレッドボード

Fig. 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

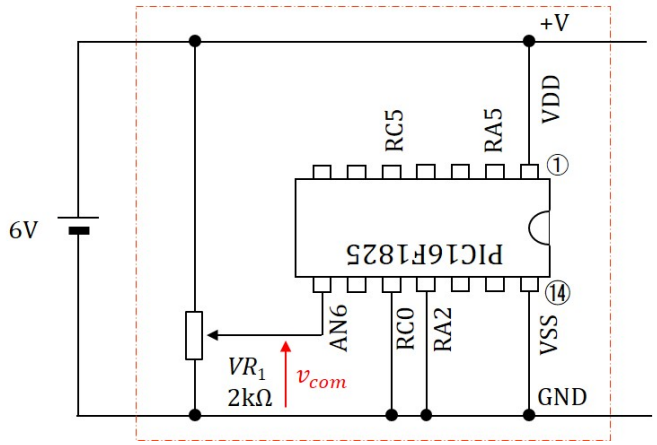


図　2.6: 正弦波生成実験の回路図

Fig. 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 に出力される正弦波電圧の周波数を変化させます．

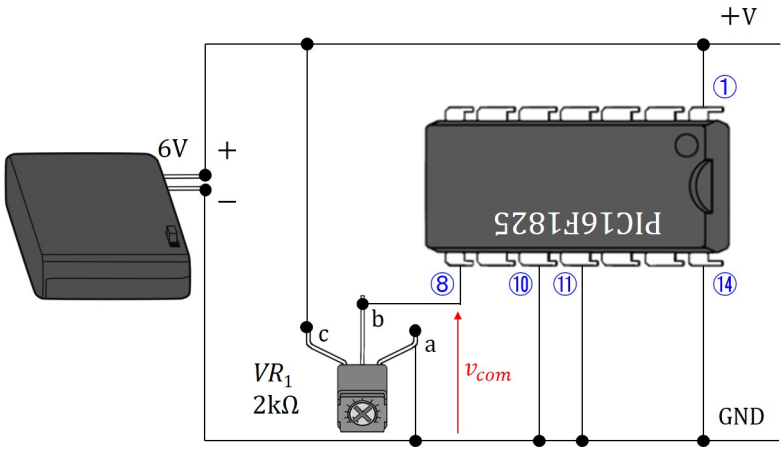


図　2.7: 正弦波生成実験回路の立体配線図

Fig. 2.7 は正弦波生成実験回路の立体配線図です．回路図と実際の部品との関係を把

握し易くするために立体図を用いて配線の様子を描いてあります．

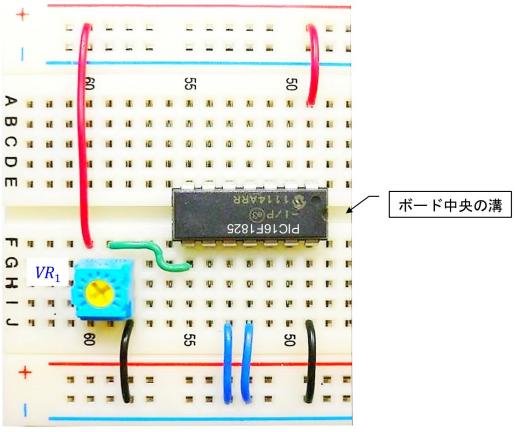


図 2.8: 正弦波生成実験回路の配線写真

Fig. 2.8 は正弦波生成実験回路の配線写真です．図 2.7 の立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の内部配線は図 2.5 に示すように，中央の溝で切れています．この溝を挟んで挿入することで，マイコンの各ピンに 4 個ずつ配線用の穴がボード内部でつながります．

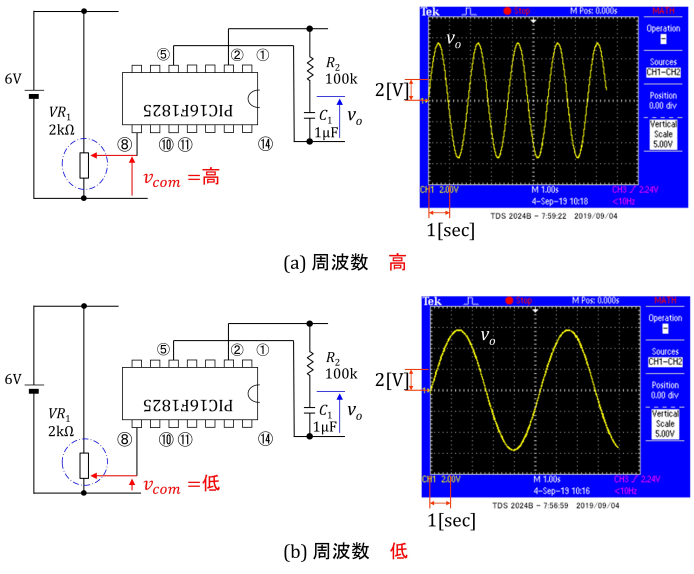


図 2.9: 正弦波生成実験回路の動作波形例

Fig. 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．

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，Fig. 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．

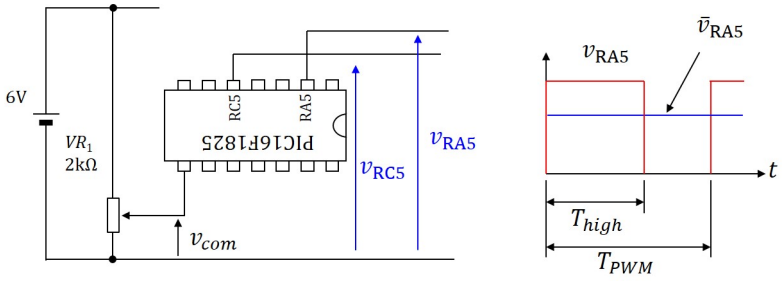


図　2.10: PWM 制御法の基本概念

Fig. 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} はスイッチング周期と呼ばれます．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

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

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 $\overline{v}_{\mathrm{RA5}}$ が近似的に正弦波となるように T_{high} を制御します．

Fig. 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

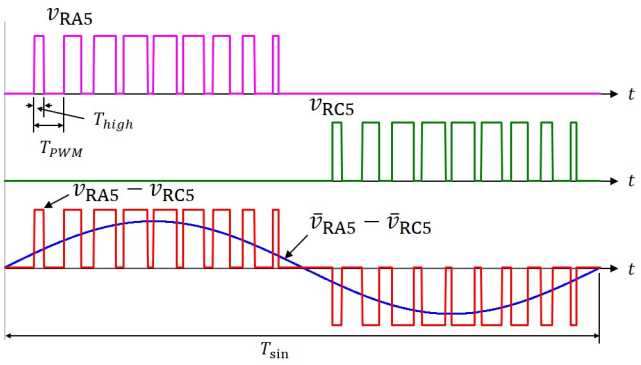


図 2.11: PWM 制御法による正弦波生成のイメージ図

負の半波を担当しています．マイコンの 2 番ピント 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．

Fig. 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，

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

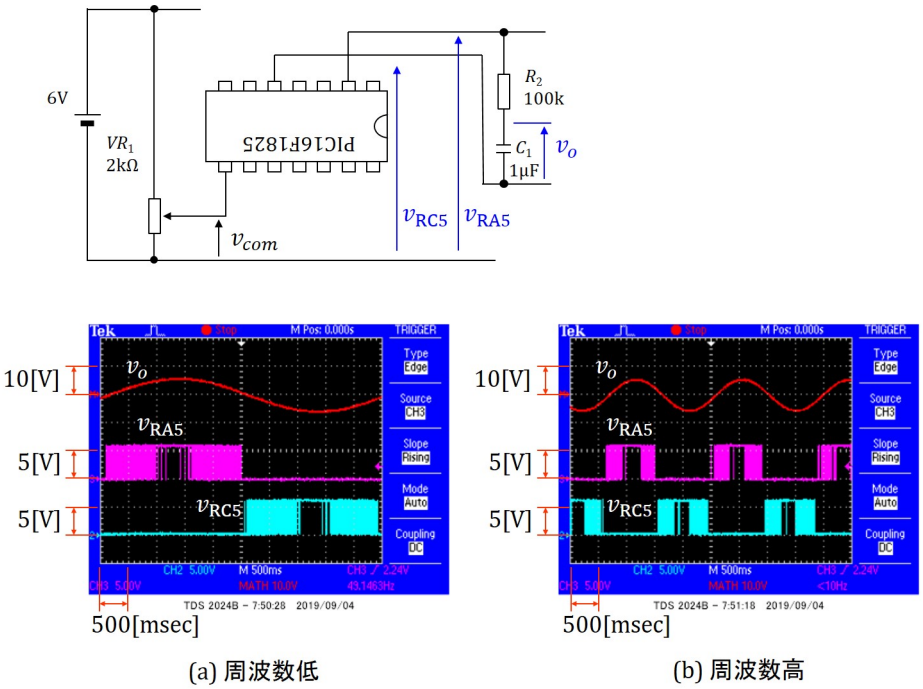


図　2.12: PWM 波形と正弦波形

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．

Fig. 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

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

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，Fig. 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 $\overline{v}_{\text{RA5}} - \overline{v}_{\text{RC5}}$ に近い値に均されます．

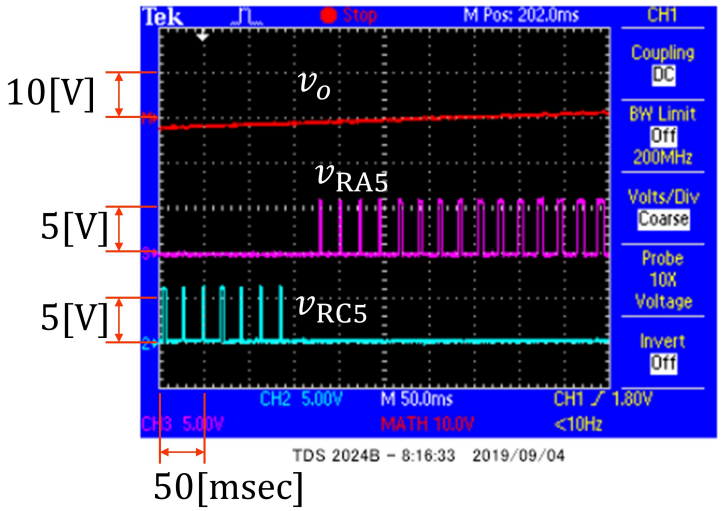


図　2.13: PWM 波形と正弦波形（時間軸拡大図）

Fig. 2.13 は Fig. 2.12 の時間軸を拡大した波形例です．Fig. 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

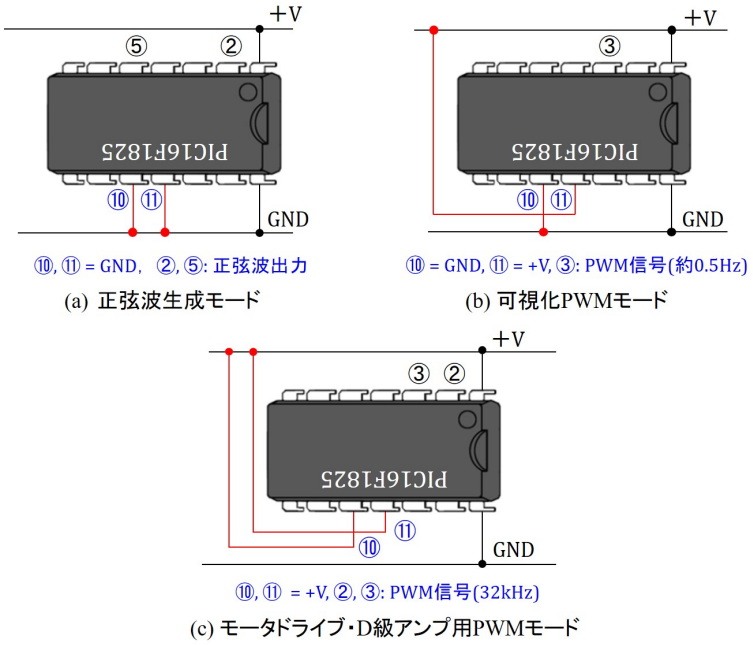


図 2.14: 3 種類のモード設定立体配線図

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 Fig. 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．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 モードを解説します．

Fig. 2.15 は可視化 PWM モードの動作波形例です．同図 (a) が通流率が高い場合，(b) が低い場合です．PWM 制御法の基本概念は Fig. 2.10 に述べたものと同じです．通流率を

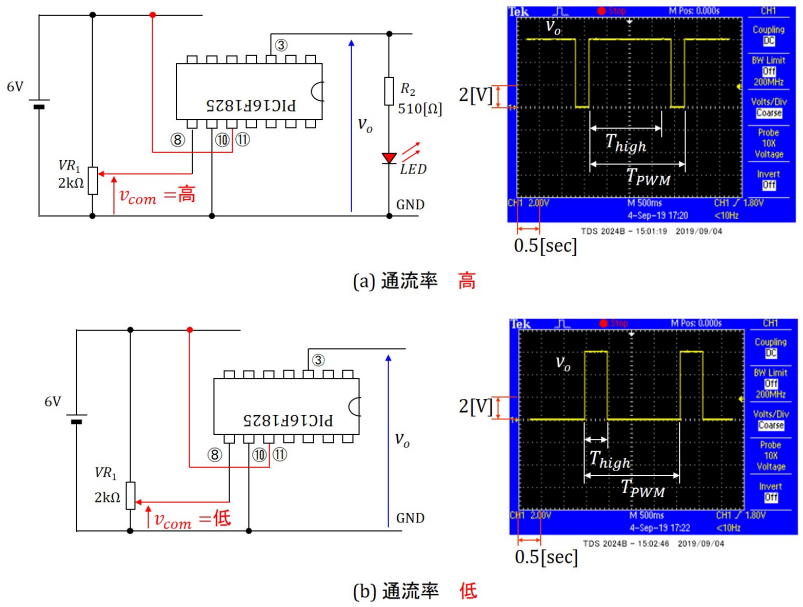


図　2.15: 可視化 PWM モードの動作波形例

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} が高いときに通流率 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

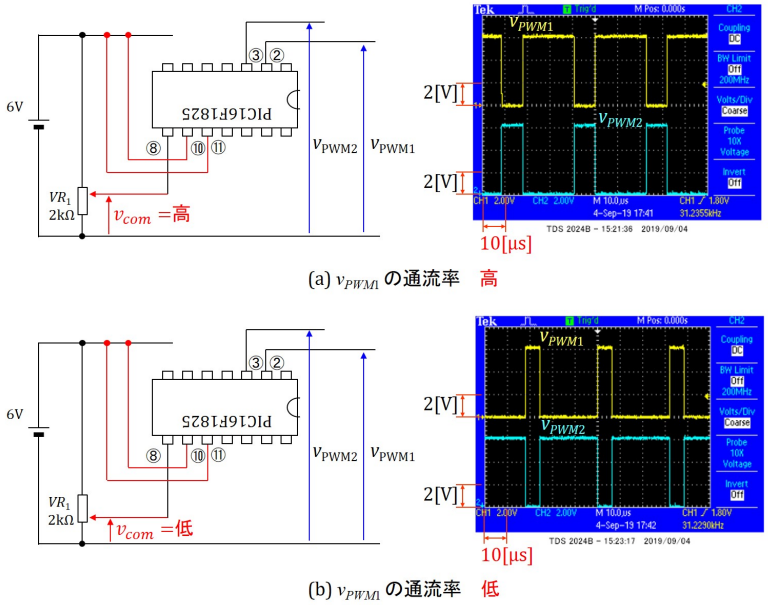


図 2.16: モータドライブ・D 級アンプ用 PWM モードの動作波形例

Fig. 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} の通流率が高い場合，(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} の通流率 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

本章では 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 コンパイラのダウンロード，インストール方法と使用方法の概要を紹介します．

MPLAB® X IDE は Microchip 社のホームページ → Design Support → Development Tools → Overview → MPLAB® X IDE → Downloads → MPLAB® X IDE v5.25 (2019 年 9 月時点) とたどることでインストーラ (MPLABX-v5.25-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) のフォルダ内に Microchip という名前のフォルダ，ドキュメントフォルダ内に MPLABXProjects という名前のフォルダが作られています．

同様に，MPLAB® XC8 コンパイラは Microchip 社のホームページ → Design Support → Development Tools → Overview → MPLAB® XC → Downloads → MPLAB® XC8 Compiler v2.10 (2019 年 9 月時点) とたどることでインストーラ (xc8-v2.10-full-install-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，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，Microchip フォルダ内に xc8¥v2.10 という名前のフォルダが作られています．v2.10¥pic¥include フォルダ内に本稿で用いるヘッダファイル xc.h がダウンロードされています．

本稿で解説するソースコードは，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_for_Web_up.zip にあります．ダウンロード，解凍し

てお使いください.

3.2　New Project の作成方法

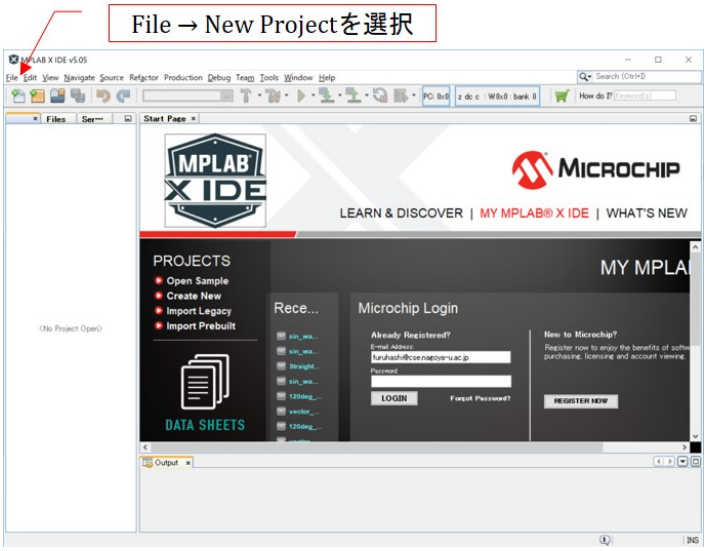


図　3.1: MPLAB® X IDE の立ち上げと New Project の設定

第 4 章で詳述するファイルを用いて，MPLAB® X IDE による編集，マイコンへの書き込み方法を記します．

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 3.1 の画面が立ち上がったら，File → New Project を選択します．次に図 3.2 のように進み，デバイスとして PIC16F1825 を選択します．

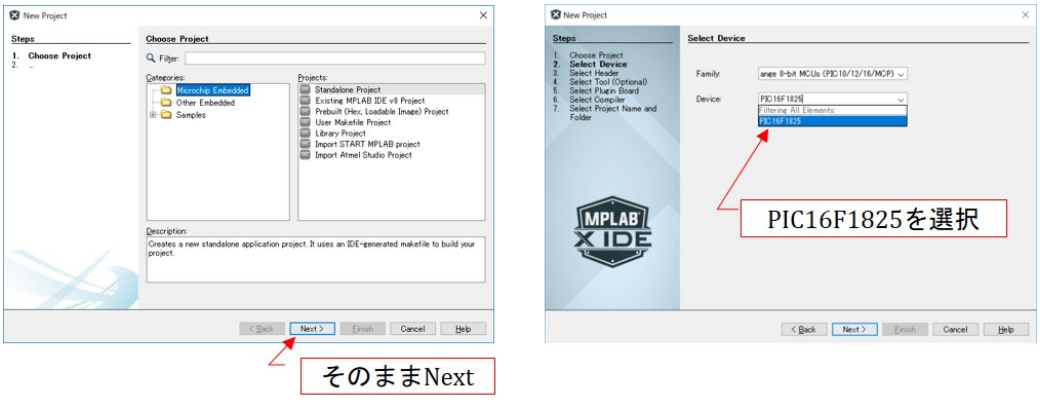


図　3.2: MPLAB® X IDE: Device 選択

その後は図 3.3 のようにデバッグツールとして Snap を選択し，コンパイラに XC8(vx.xx)....を選択します．図 3.4 は次に表示される画面です．あらかじめ作っておいたフォルダ（例

えば，MPLABXProjects フォルダ内に PIC16F1825 という名前のフォルダを作っておく）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 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.

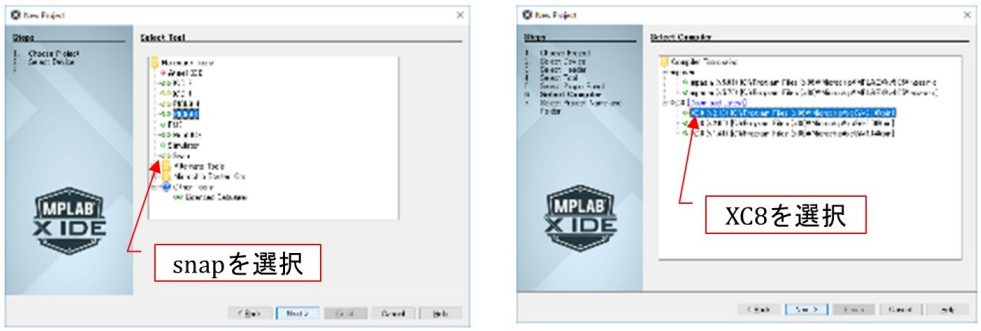


図 3.3: MPLAB® X IDE: tool, compiler 選択

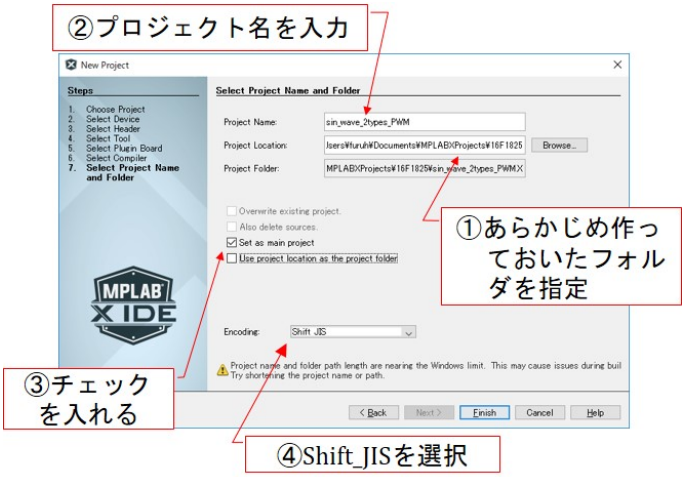


図 3.4: MPLAB® X IDE: project name, folder, 言語選択

以上が完了した段階で,（上の例では PIC16F1825 のフォルダ内に sin_wave_2types_PWM という名前の）フォルダが作られています．図 3.5 のように，このフォルダの中へ

パワーエレクトロニクスノート

からダウンロードしておいた「sin_wave_2types_PWM_for_Web_up」フォルダ内のソースファイル (main.c) および関連ファイルの入っているフォルダ (PIC16F1825_Files) をコピーします.

次に，これらのファイルをプロジェクトに追加します．図 3.6 のように，Source Files のフォルダを右クリックし，ADD Existing Item を選択します．そして，同図右のよう

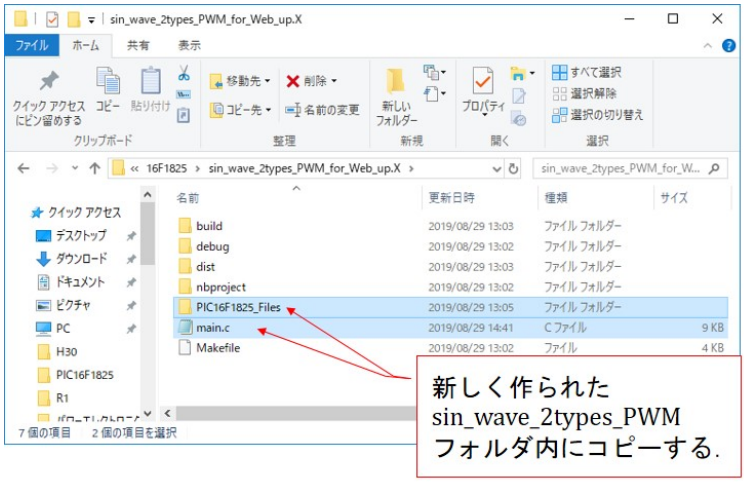


図 3.5: ソースファイル，関連フォルダの sin_wave_2types_PWM フォルダ内へのコピー

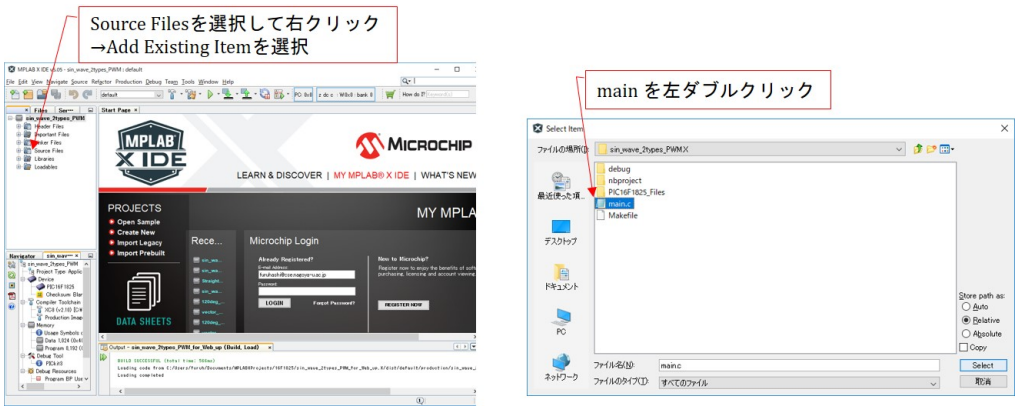


図 3.6: MPLAB® X IDE: Source file の追加

に main.c を選択します．次に，図 3.7 のように，Linker Files を選択して右クリックし，ADD Existing Item を選択します．そして，同図右のように，set_ad_converter.c, set_osc.c, set_pwm.c, set_timer1.c, set_timer2.c を選択します．

最後に，図 3.8 の画面のように，Header Files のフォルダを右クリックして，ヘッダファイル pic16f1825_s.h を追加します．

図 3.7: MPLAB® X IDE: Linker files の追加

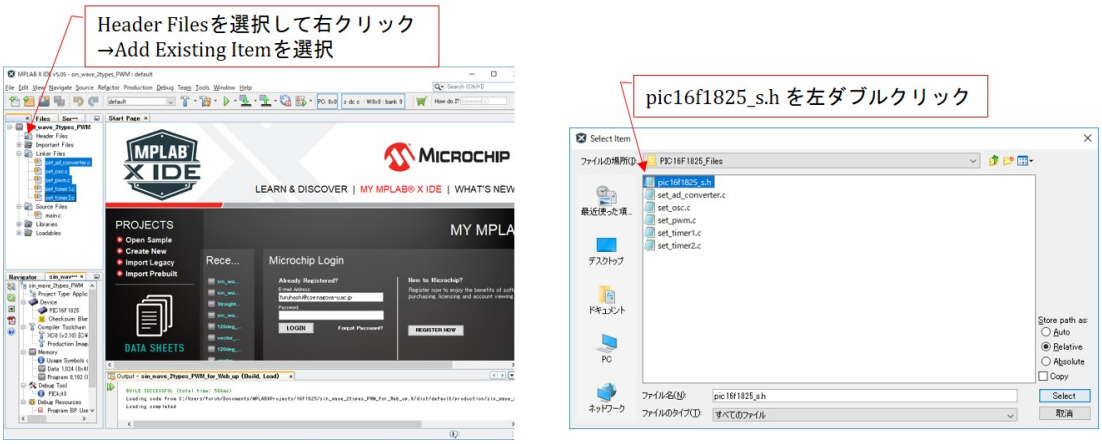


図 3.8: MPLAB® X IDE: Header の追加

3.3　プログラムのマイコンへの書き込み

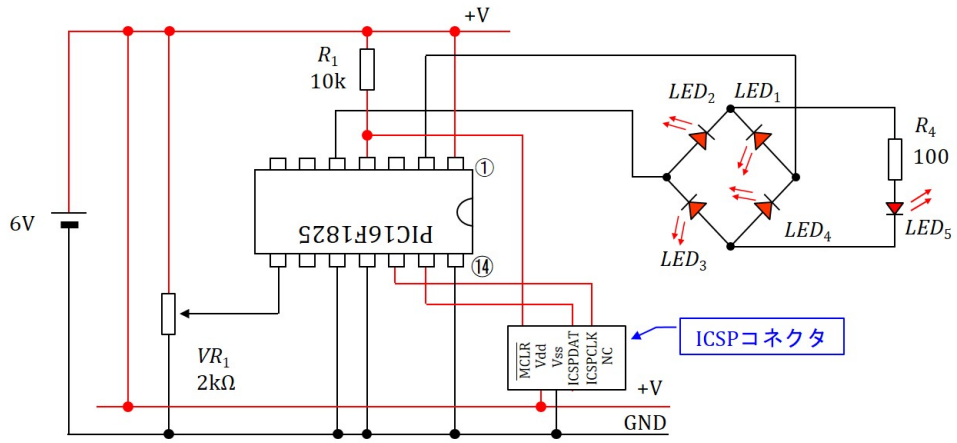


図　3.9: In-Circuit Debugger/Prgrammer によるプログラム書き込み用回路

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．Fig. 3.9 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．Fig. 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．コネクタの各ピンの名称は，これらピンとつながるマイコン側のピンの名称 ($\overline{\text{MCLR}}$, ICSPDAT, ICSPCLK) を記してあります．全波整流回路は LED $LED_1 \sim LED_5$ と抵抗 R_4 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．

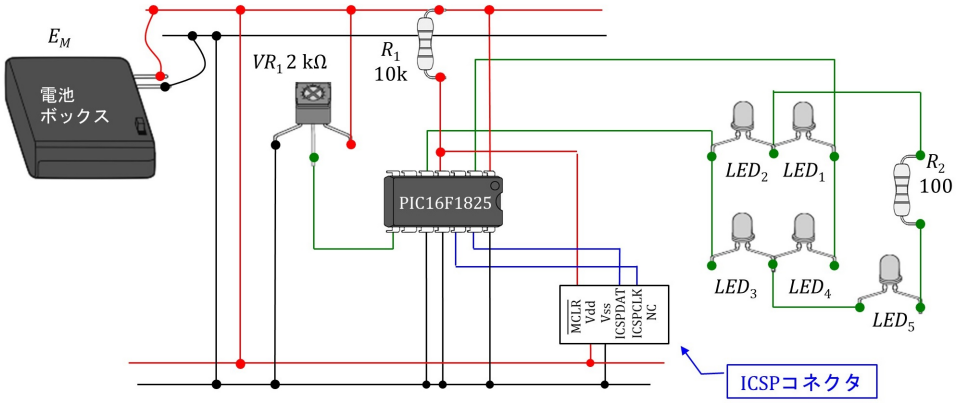


図 3.10: In-circuit Prgrammer によるプログラム書き込み用回路の立体配線図

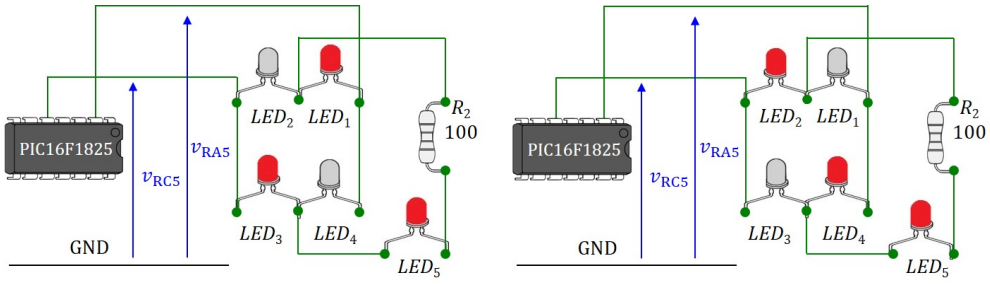


図 3.11: プログラム書き込み成功したときの回路動作

Fig. 3.10 はプログラム書き込み用回路の立体配線図です．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．Fig. 3.11 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．

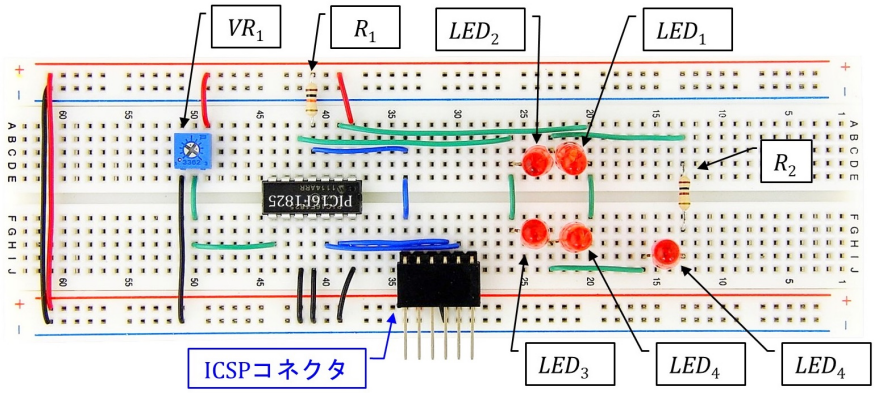


図 3.12: In-circuit Prgrammer によるプログラム書き込み用回路の写真

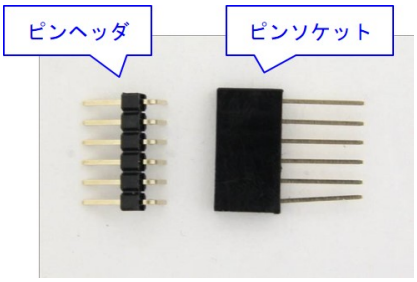
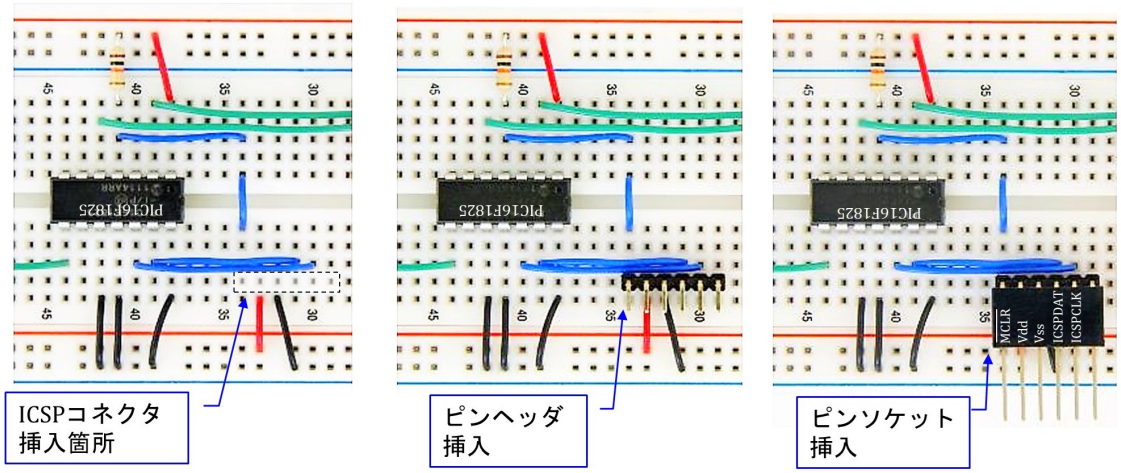


図 3.13: ピンヘッダとピンソケット

Fig. 3.12 はプログラム書き込み用回路の写真です．ICSP コネクタは，Fig. 3.13 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 3.14: ICSP コネクタの製作

Fig.3.14 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 個の穴にピンヘッドを挿入します．同図 (b) がピンヘッダを挿入した写真です．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．

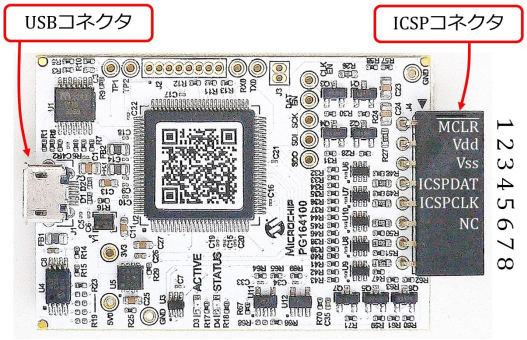


図 3.15: MPLAB® Snap

Fig. 3.15 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.3.14 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．

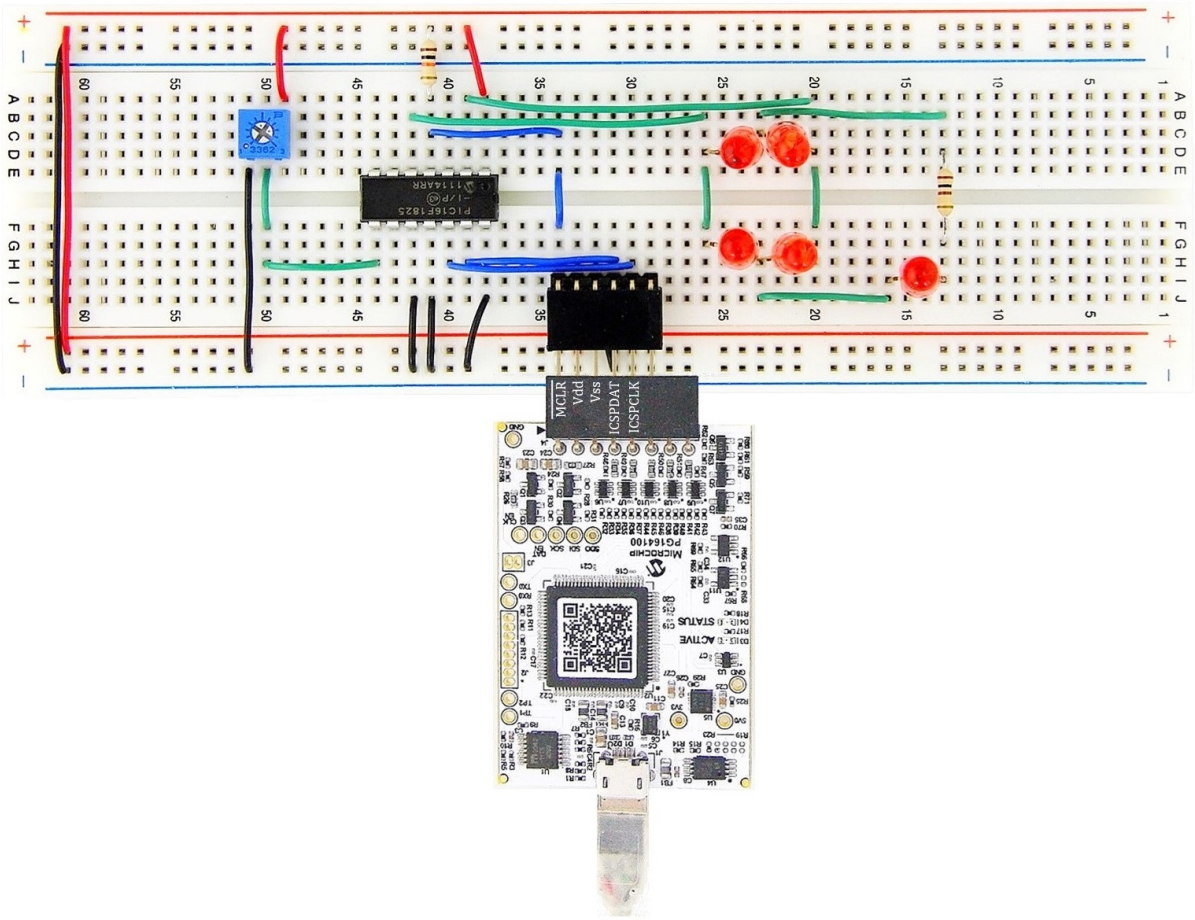
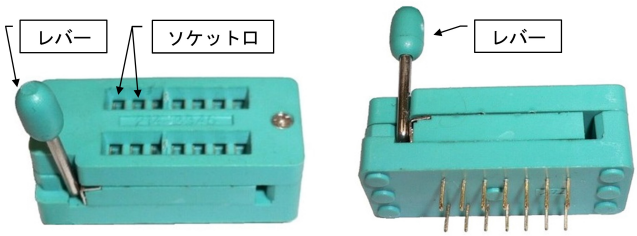


図 3.16: ICSP コネクタに MPLAB® Snap を接続

Fig. 3.16 は Fig.3.14 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．



(a) 上面と側面　　(a) 側面と下面

図 3.17: ゼロプレッシャーソケット

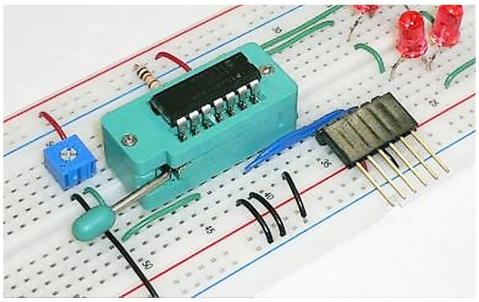


図 3.18: ゼロプレッシャーソケット使用によるマイコン挿抜の容易化

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．Fig. 3.17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．Fig. 3.18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．

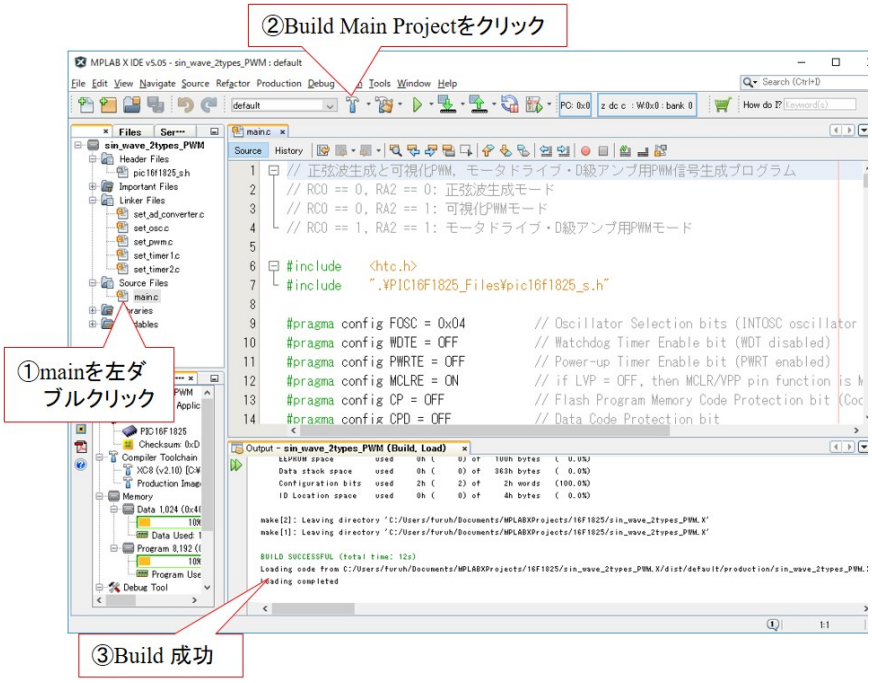


図　3.19: MPLAB® X IDE: Build

以上で MPLAB® X IDE によるプログラムの編集，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 もしくは PICkit™4 を接続し，MPLAB® Snap PICkit™4 とパソコンを USB ケーブルで接続します．

図 3.19 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．そして，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)

のメッセージが出れば，全ての準備が完了です．

Fig. 3.20 のように ▷ 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します．同図下の画面のように

Programming/Verify complete

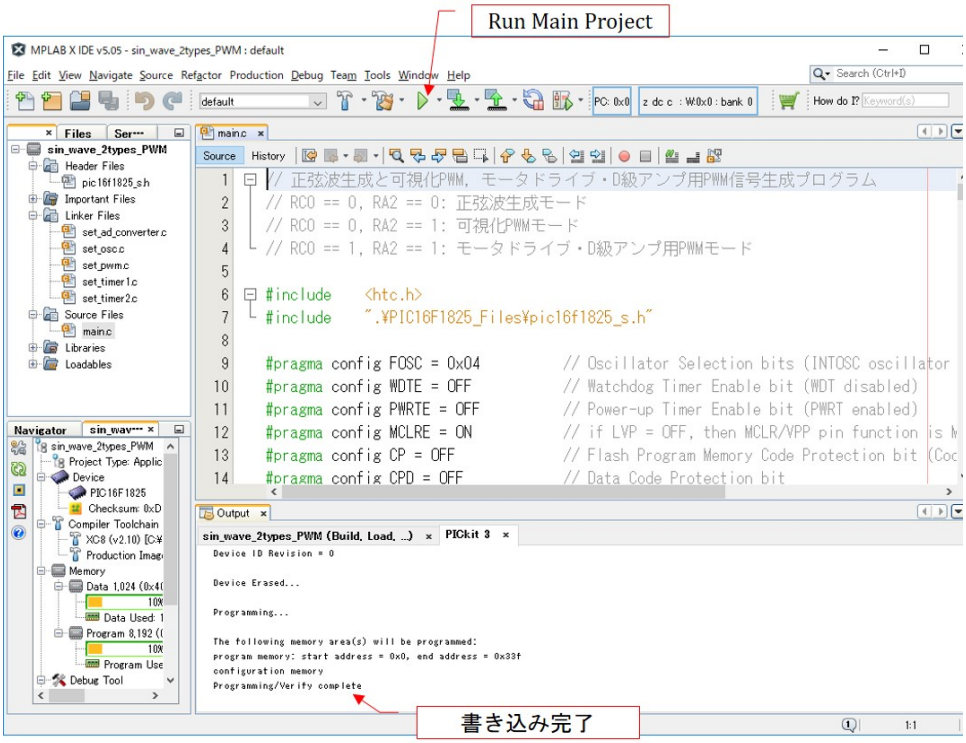


図 3.20: MPLAB® X IDE: Build とマイコンへの書き込み

が表示されれば，書き込みが完了し，Fig. 3.11 の様に LED がゆっくりと明るさを変化させるはずです．可変抵抗器上面のつまみをネジ回しで回すと，LED の明かりの変化速度が変わります．

3.4　デバッガの使用方法

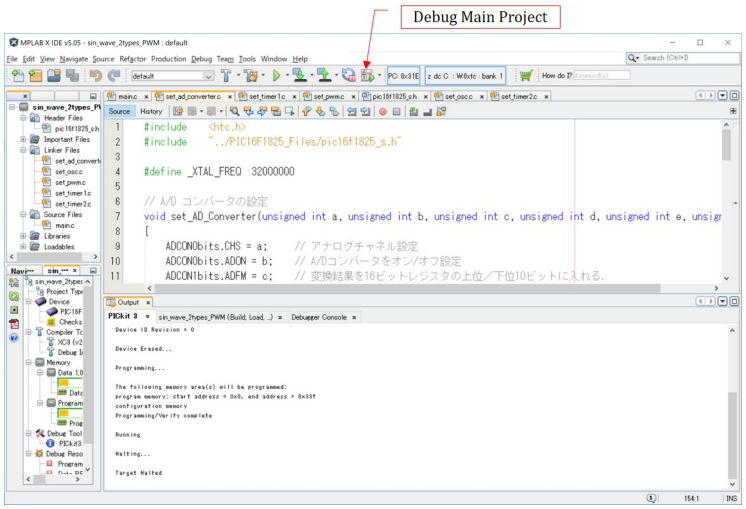


図　3.21: デバッガの起動

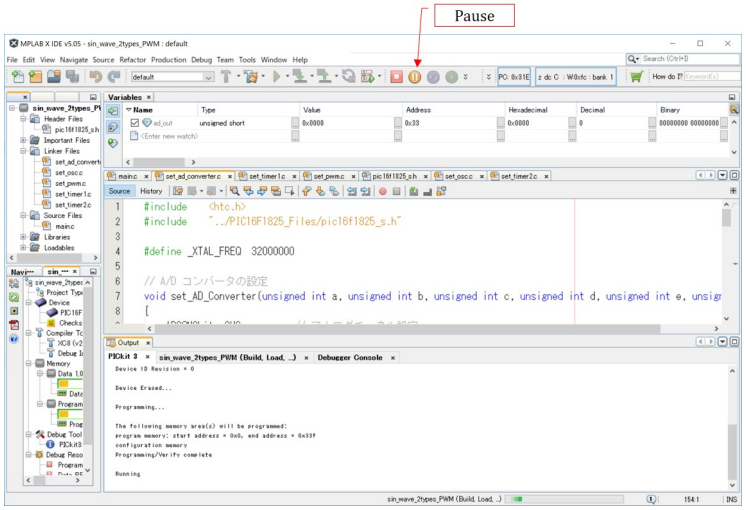


図　3.22: デバッガの一時停止

プログラムを改変したいときにはデバッガが便利です．Fig. 3.21 の Debug Main Project 1 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．そして，Fig. 3.22 の 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．そして，この停止時のグローバル変数の値，レジスタの値などを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

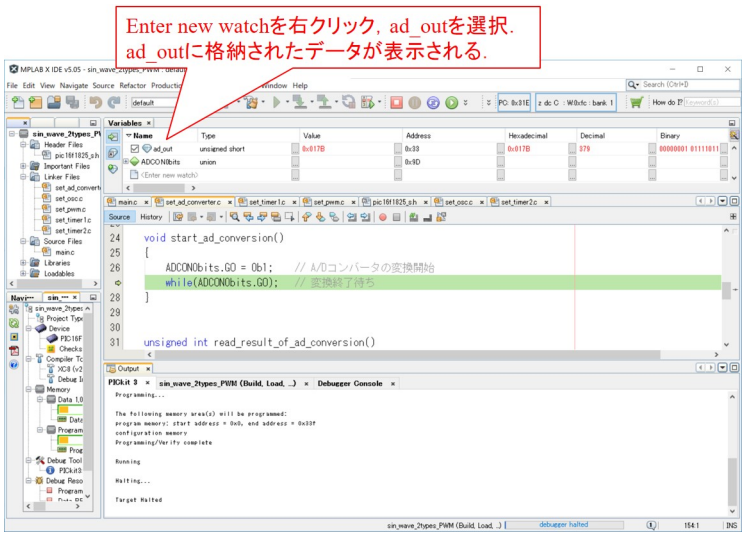


図 3.23: 表示データの指定

圧を A/D 変換して 10 ビットの値で読み出し，グローバル変数 ad_out に格納します．図 3.23 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．

第 4 章

プログラム

本章では sin_wave_2types_PWM.c のプログラムを解説します．本稿で使用する関数，ヘッダファイルは全て

モータドライブノート　I. PIC16F1825 による DC モータの回転数制御

にて詳細に記述してありますので，こちらを参照して下さい．

4.1　ヘッダファイル

```
#include     <xc.h>
#include     ".¥PIC16F1825_Files¥pic16f1825_s.h"
```

図　4.1: ヘッダファイルのインクルード

Fig. 4.1 はヘッダファイルのインクルードです．pic16f1825_s.h は筆者オリジナルのヘッダファイルです．上述の「I. PIC16F1825 による DC モータの回転数制御」にて，オシレータ，タイマー 1, 2，A/D 変換，PWM 等の各モジュール設定用関数を作成し，その引数に分かり易い表現を採用しました．そして，その表現とレジスタの数値との対応表をヘッダファイル pic16f1825_s.h にまとめてあります．

4.2 デバイスコンフィギュレーション

```
#include     <xc.h>
#include     ".¥PIC16F1825_Files¥pic16f1825_s.h"


#pragma config FOSC = 0x04          // 内蔵オシレータ オン，RA5（2番ピン）をクロック入力用ではなく，I/O用に設定
#pragma config WDTE = OFF           // ウォッチドッグタイマ オフ
#pragma config PWRTE = OFF          // パワアップタイマ オフ
#pragma config MCLRE = ON           // 4番ピンをMCLR用に設定
#pragma config CP = OFF             // コード保護 オフ
#pragma config CPD = OFF            // データコード保護 オフ
#pragma config BOREN = OFF          // 低電圧リセット オフ
#pragma config CLKOUTEN = OFF       // クロック信号出力ピン（3番ピン）オフ
#pragma config IESO = OFF           // 電源立ち上げ時の内蔵オシレータ→外部オシレータ切替 オフ
#pragma config FCMEN = OFF          // フェイルセーフクロックモニタ オフ
#pragma config WRT = OFF            // フラッシュメモリ書き込み保護 オフ
#pragma config PLLEN = ON           // PLL オン
#pragma config STVREN = OFF         // スタックオーバー／アンダーフローリセット オフ
#pragma config BORV = HI            // 低電圧リセット電圧を2.7[V]に設定．(LOW=1.9 [V])
#pragma config LVP = ON             // 低電圧（例えば3.3 [V]）でのプログラミング オン
                                    // (MPLAB SnapはLVP ONでのみ書き込み可)
```

図 4.2: デバイスコンフィギュレーション

図 4.2 はデバイスコンフィギュレーションです．本稿では

FOSC = INTOSC : 内蔵オシレータオン, RA5（2 番ピン）を I/O 用に設定

MCLRE = ON : $\overline{\text{MCLR}}$ピン（4 番ピン）による強制リセットを可能とする

PLLEN = ON : 内蔵オシレータにより得られたクロックを PLL により 4 倍にする

BORV = HI : 低電圧リセット電圧を 2.7[V] に設定

LVP = ON : 低電圧 (例えば 3.3[V]) でのプログラミングオン

のみ設定しています．実験室レベルでは，これで十分です．MPLAB Snap は LVP = ON でのみ，書き込み可能なので注意してください．

4.3 メイン関数

```
void main()
{
    TRISA = 0x04;               // RA2(11番ピン)を入力ポートに設定
    TRISC = 0x05;               // RC2(8番ピン), RC0(10番ピン)を入力ポートに設定
    ANSELA = 0x00;
    ANSELC = 0x04;              // RC2(8番ピン)をアナログ入力に設定

    //オシレータの設定
    set_osc(Int_OSC_Freq_8MHz, SysClockSource_detmd_by_Config);
                                // 内蔵オシレータ8MHzでFOSC=32MHzと設定する.

    //  タイマー1の設定
    set_timer1(TMR1_clock_source_FOSC, T1Clock_PreScale_1_1, TMR1_ON);
                                // タイマ1のクロックソース, 分周率を設定して, タイマ1をオンとする.
    set_interrupt_by_timer1();  // タイマ1による割り込みを可とする.

    // A/Dコンバータの設定
    set_AD_Converter(select_AN6, AD_ON, right_justified, clock_1_32, neg_ref_VSS, pos_ref_FVR);

    // FVRの設定
    set_FVR(FVR_ON, ad_ref_4_096);

    if((RC0 == 1) && (RA2 == 1))   // モータドライブ & D級アンプ用PWM
    {
        // PWMモジュールの設定
        set_PWM(Half_Bridge_with_P2A_P2B, P2A_B_C_D_active_high, P2B_to_RA4, P2A_to_RA5, based_on_timer2);

        // タイマ2の設定
        set_timer2(set_Postscaler_1_1, timer2_on, set_Prescaler_1_1, PWM_period); //スイッチング周波数  約32kHz
    }

    for(;;)
        continue;       // 無限ループ
}
```

図 4.3: メイン関数

図 4.3 はメイン関数です.

$$\begin{aligned} \mathrm{TRISA} &= 0\mathrm{x}04; \\ \mathrm{TRISC} &= \mathrm{Ox}05; \end{aligned} \tag{4.1}$$

により，RA2(11 番ピン)，RC2(8 番ピン), RC0(10 番ピン) を入力ポートに設定し，さらに，

$$\begin{aligned} \mathrm{ANSELA} &= 0\mathrm{x}00; \\ \mathrm{ANSELC} &= 0\mathrm{x}04; \end{aligned} \tag{4.2}$$

により，RA2(11 番ピン)，RC0(10 番ピン) をディジタル入力，RC2(8 番ピン) をアナロ

グ入力に設定しています．

set_osc(Int_OSC_Freq_8MHz, SysClockSource_detmd_by_Config);

により，内蔵オシレータのクロック (16 [MHz]) をプリスケーラにより 8 [MHz] とし，4 逓倍 PLL により 32 [MHz] としてシステムクロック FOSC を 32 [MHz] に設定します．set_osc() 関数は，set_osc.c ファイル内で定義され，引数の用語 (Int_OSC_Freq_8MHz, SysClockSource_detmd_by_Config）はヘッダファイル pic16f1825_s.h 内で定義されています．詳細はモータドライブノート　I. PIC16F1825 による DC モータの回転数制御を参照して下さい．以降の関数及び用語の定義も同様です．

set_timer1(TMR1_clock_source_FOSC, T1Clock_PreScale_1_1, TMR1_ON);

により，タイマ 1 のクロックソースをシステムクロック FOSC とし，クロックの分周率を 1:1 に設定して，タイマ 1 をオンとします．

set_interrupt_by_timer1();

により，タイマ 1 による割り込み可とします．

set_AD_Converter(select_AN6, AD_ON, right_justified, clock_1_32, neg_ref_VSS,
pos_ref_FVR);

により，A/D 変換モジュールを設定します．AN6(8 番ピン) を A/D 変換モジュールの入力とし，A/D 変換モジュールをオンとします．このマイコンの A/D 変換結果は 10 ビットです．結果の格納用レジスタは 16 ビットなので，right_justified により右寄せで（下 10 ビットに）結果を格納します．クロックソースは FOSC で固定です．A/D 変換器の推奨クロック周波数は 0.25〜1 [MHz] です．そこで，FOSC = 32 [MHz] としたので，clock_1_32 により,AD 変換器のクロックを FOSC/32 = 1 [MHz] に設定します．A/D 変換器の基準電圧は，－側を電源の－側 (VSS) とし，＋側を FVR とします．FVR は

set_FVR(FVR_ON, ad_ref_4_096);

により，4.096 [V] を選択します．

RC0 と RC2 の入力値により次の 3 通りの場合分けをします．

RC0 = 1, RC2 = 1 のとき：モータドライブ & D 級アンプ用 PWM モード
RC0 = 0, RC2 = 0 のとき：正弦波生成モード
RC0 = 0, RC2 = 1 のとき：可視化 PWM モード (4.3)

モータドライブ & D 級アンプ用 PWM モードのときのみ PWM モジュールの設定をします．

```
set_PWM(Half_Bridge_with_P2A_P2B, P2A_B_C_D_active_high, P2B_to_RA4,
        P2A_to_RA5, based_on_timer2);
```

により，PWM モジュールを設定します．ハーフブリッジの PWM 信号を P2A，P2B に出力します．P2A, P2B は正論理とします．P2B を RA4(3 番ピン) に，P2A を RA5(5 番ピン) に割り当てます．PWM モジュールのタイマとしてタイマ 2 を選択します．

```
set_timer2(set_Postscaler_1_1, timer2_on, set_Prescaler_1_1, PWM_period);
```

により，タイマ 2 のポストスケーラを 1:1 にし，タイマ 2 をオンとし，プリスケーラを 1:1 にします．デバイスコンフィギュレーションの次の行で

```
#define    PWM_period    0xFF
```

により，PWM_period $=$ $0xFF_{(16\text{ 進数})}$ $=$ $255_{(10\text{ 進数})}$ と定義しています．PIC16F1825 のデータシートより，タイマ 2 のクロックは FOSC/4 です．PWM 周期 T_{TPWM} を

$$\begin{aligned} T_{PWM} &= \frac{\mathrm{FOSC}}{4} \times \frac{1}{256} \\ &= 32[\mu s] \end{aligned} \tag{4.4}$$

と設定します．詳細はモータドライブノート　I. PIC16F1825 による DC モータの回転数制御の PWM モジュールの項を参照して下さい．

```
for(;;)
continue;
```

は，何もしない無限ループです．この間，タイマ 1 による割り込みで，タイマ 1 割り込みルーチンが起動されます．

4.4 タイマ 1 割り込み関数

```
unsigned short     ad_out;

static void __interrupt()        // Timer1 による割り込み処理ルーチン
timer1_isr(void)
{
      unsigned short        Int_Data, Duty;

      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

      //A/D変換
      start_ad_conversion();
      ad_out = read_result_of_ad_conversion();

      if((RC0 == 1) && (RA2 == 1)){          // モータドライブ & D級アンプ用PWM
            ・・・
      }
      else if((RC0 == 0) && (RA2 == 0)) {    // サイン波形生成
            ・・・
      }
      else if((RC0 == 0) && (RA2 == 1)){     // 0.5Hz PWM
            ・・・
      }

      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 4.4: タイマ 1 割り込み関数

図 4.4 はタイマ 1 割り込み関数です．timer1_isr() はこの関数名です．どのような名前を選んでもよいので，timer1_isr(timer1 Interrupt Service Routine) としました．

RC3 = 1;

により，RC3(7 番ピン) に 1 (VDD) を出力します．そして，このルーチンの最後で

RC3 = 0;

により，RC3 に　0 (VSS) を出力します．これにより，割り込み処理ルーチンの所要時

間を計測することができます.

if(RC0 == x) && (RA2 == y)

により，(4.3) 式の各モードのプログラムを実行します.

4.4.1　モータドライブ・D 級アンプ用 PWM モード

```
if((RC0 == 1) && (RA2 == 1))   // モータドライブ & D級アンプ用PWMモード
{
    set_timer1_count_down_ini_num(0x0190);
        // タイマ1のカウントダウン値の設定.
        // 入力値を初期値としてカウントダウンを行い，0で割り込みを発生することとなる.
        // AD変換器のサンプリング周期とPWM周期を同期させるためにタイマ1の割り込み周期を調整
        // サンプリング周波数　約31.2kHz
    clear_interrupt_flag_of_timer1();
        // タイマ1の割り込みフラグを0にして，次のタイマ1による割り込みを受付け可とする

    set_PWM_duty_cycle(ad_out);     // PWM通流率の設定
}
```

図　4.5: タイマ 1 割り込み関数_モータドライブ & D 級アンプ用 PWM

図 4.5 はモータドライブと D 級アンプ用 PWM モードのプログラムです.

set_timer1_count_down_ini_num(0x0190);

により，タイマ 1 のカウントダウン値を設定しています．タイマ 1 は 0x0190 からカウントダウンを始めて，0 となったときに再び timer1_isr() を起動（することに相当する動作を）します．タイマ 1 による割り込み周期 T_{TM1} と (4.4) 式の PWM 周期 T_{PWM} を同じにするために，0x0190 の数値を試行により決めました．これにより，A/D 変換のサンプリング周波数 f_{SMAP} は

$$\begin{aligned} f_{SAMP} &= \frac{1}{T_{TM1}} \\ &= \frac{1}{T_{PWM}} \\ &\approx \frac{1}{32[\mu\mathrm{s}]} \\ &\approx 31.2[\mathrm{kHz}] \end{aligned} \tag{4.5}$$

となります.

clear_interrupt_flag_of_timer1();

により，割り込みフラグを 0 にクリアして，次の割り込みを受け付けられるようにしておきます.

set_PWM_duty_cycle_out();

により，A/D 変換結果 ad_out を PWM の通流率に設定します.

4.4.2 正弦波生成モード

図 4.6 は正弦波生成モードのプログラムです．定数の定義と変数の宣言も併せて載せてあります.

sin_data[sin_div+1]

に正弦波データを 61 個格納します．図 4.7 の通り 60 個で正弦波の 1/4 周期分です．プログラムではこのデータをつなぎ合わせて 1 周期の正弦波形を作ります.

Int_Data = MaxInterval_of_PWM_step - ad_out*2

により，タイマ 1 による割り込み間隔を決定します.

#define MaxInterval_of_PWM_step 0x0820

と定義しています．A/D 変換器は 10 ビットなので

$$0 \leq \text{ad\_out} \leq 1023 \tag{4.6}$$

の範囲を変化します．よって，Int_Data は

$$34 \leq \text{Int\_Data} \leq 2080 \tag{4.7}$$

の範囲を変化します．最小値は試行により 34 としました．これより小さい場合は，割り込み時間間隔が短くなりすぎてプログラムが誤動作します.

```
unsigned short    ad_out;

#define           sin_div     60                  // 1/4周期
#define           sin_div2  120                   // 1/2周期
char              sin_data[sin_div+1]             // 正弦波データ
                    ={0,7,13,20,27,33,40,46,53,60,66,72,79,85,91,98,104,110,116,122,128,133,139,144,150,
                      155,160,166,171,176,180,185,190,194,198,202,206,210,214,217,221,224,227,230,233,
                      236,238,240,243,244,246,248,249,251,252,253,254,254,255,255,255};
#define           MaxInterval_of_PWM_step   0x0820  // PWMステップの最長時間
char              sin_step = 0;                   //正弦波データの番地カウンタ
unsigned short    pwm_step = 0;                   // PWMのデューティ比決定用カウンタ
char              flag_sign = 0;                  // 0: 正の半波，1: 負の半波

else if((RC0 == 0) && (RA2 == 0))                 // 正弦波生成モード
{
    Int_Data = MaxInterval_of_PWM_step - ad_out*2;    // PWMステップの時間決定

    set_timer1_count_down_ini_num(Int_Data);      // タイマ1のカウントダウン値の設定.
    clear_interrupt_flag_of_timer1();             // 次の割り込みを受付け可とする.

    if(sin_step < sin_div){                       // 正弦波データによりデューティ比決定
        Duty = sin_data[sin_step];                // 1周期の0--1/4，1/2--3/4の期間のデューティ比
    } else if(sin_step < sin_div2){
        Duty = sin_data[sin_div2 - sin_step];     // 1周期の1/4--1/2，3/4--1の期間のデューティ比
    }
    if(pwm_step <= Duty){             // PWM波形生成
        if(flag_sign < 1){            // 正の半波
            RA5 = 1;
        }else{                        // 負の半波
            RC5 = 1;
        }
    }else{
        RA5 = 0;
        RC5 = 0;
    }
    pwm_step += 1;                    // PWMステップ +1

    if(pwm_step >= 255){              // PWMステップ数最大値 = 255
        pwm_step = 0;                 // PWMステップ リセット
        sin_step += 1;                //正弦波データの番地を +1  (1/4周期sin_div, ½周期sin_div2)
        if(sin_step>=sin_div2){       // 正弦波の正負設定
            sin_step = 0;             // 正弦波データの番地リセット
            if(flag_sign < 1){
                flag_sign = 1;        // 負の半波へ移行
            }else{
                flag_sign = 0;        // 正の半波へ移行
            }
        }
    }
}
```

図 4.6: タイマ１割り込み関数_正弦波生成

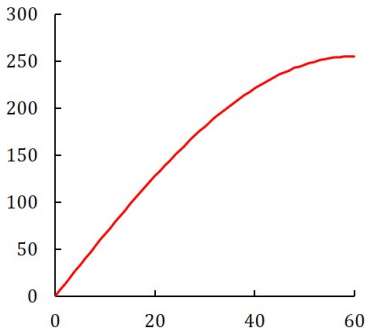


図 4.7: 正弦波データ

set_timer1_count_down_ini_num(Int_Data); // タイマ 1 のカウントダウン値の設定.
clear_interrupt_flag_of_timer1();

により，次の割り込みタイミングを設定して，タイマ 1 による再割り込みを可とします.

```
if(sin_step < sin_div){              // 正弦波データにより通流率決定
    Duty = sin_data[sin_step];    // 0–1/4，1/2–3/4 の期間の通流率
}else if(sin_step < sin_div2){
    Duty = sin_data[sin_div2 - sin_step];//1/4–1/2，3/4–1 の期間の通流率
}
```

により，正弦波データ 1 周期分のデータを順次読み出して Duty に格納します．図 4.7 の 1/4 周期分のデータを左から順に読み出した後，次の 1/4 周期は sin_div2 - sin_step により右から順に読み出します．正弦波形の正の半周期と負の半周期では同じデータを使います．Duty は次のプログラムで PWM の通流率を決定します.

```
if(pwm_step <= Duty){       // PWM 波形生成
    if(flag_sign < 1){          // 正の半波
        RA5 = 1;
    }else{                      // 負の半波
        RC5 = 1;
    }
}else{
    RA5 = 0;
```

```
        RC5 = 0;
    }
    pwm_step += 1;                  // PWM ステップ +1
    if(pwm_step >= 255){            // PWM ステップ数最大値 = 255
        pwm_step = 0;               // PWM ステップ リセット
        ...
    }
```

図 4.8 はこのプログラムによる PWM 波形生成の様子を示します．pwm_step ≤ Duty の間は RA5（正の半波のとき）もしくは RC5（負の半波のとき）に 1 (VDD) を出力し，pwm_step > Duty にて RA5 = RC5 = 0 (VSS) を出力します．pwm_step は割り込み毎に 1 ずつ増加して，255 に達したときに 0 にリセットします．RA5 および RC5 の波形が PWM 波形です．

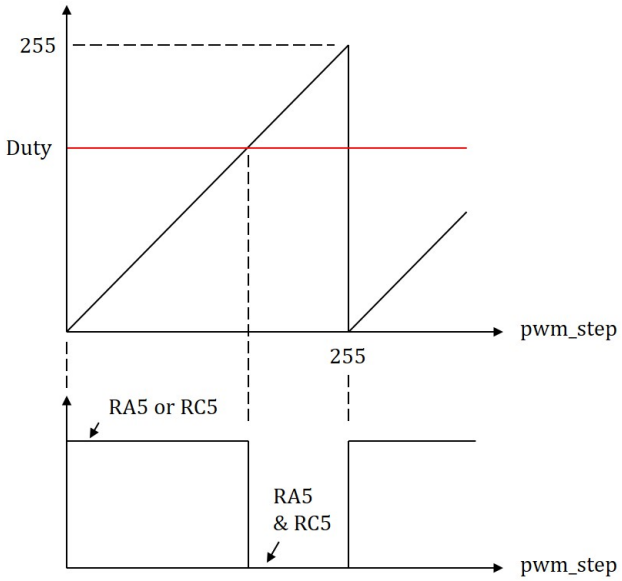


図　4.8: PWM 波形生成の様子

正の半波では RA5 のみに PWM 波形を出力し，負の半波では RC5 のみに PWM 波形を出力しています．これにより RA5 と RC5 間の電圧は − VDD ～ VDD となります．よく知られている PWM 波形では，RA5 と RC5 を相補的（一方が VDD/VSS を出力しているときに，もう一方は VSS/VDD を出力するよう）に用います．しかし，相補的に用いた場合の RA5-RC5 間の電圧範囲は　0 ～ VDD と狭くなってしまいます．

```
if(pwm_step >= 255){              // PWM ステップ数最大値 = 255
    pwm_step = 0;                 // PWM ステップ リセット
    sin_step += 1;                //正弦波データの番地を +1
    if(sin_step >= sin_div2){     // 正弦波の正負設定
        sin_step = 0;             // 正弦波データの番地リセット
        if(flag_sign < 1){
            flag_sign = 1;        // 負の半波へ移行
        }else{
            flag_sign = 0;        // 正の半波へ移行
        }
    }
}
```

により，pwm_step が 255 に達したとき，sin_step を 1 つ増加させます．これにより次の割り込みで通流率が更新されます．そして，sin_step が sin_div2 = 120（正弦波半周期分のデータ数）に達したとき，sin_step をリセットします．同時に正弦波の正負フラグ (flag_sign) を反転します．

PWM 波形から正弦波形が得られる様子は，2.1.2 項にて述べてあります．この項を参照して下さい．

4.4.3 可視化 PWM モード

図 4.9 は可視化 PWM モードのプログラムです．

```
#define  Interval_of_TMR1_Interrupt  0xFFFF
```

によりタイマ 1 による割り込み周期 T_{TMR1} を

$$\begin{aligned} T_{TMR1} &= \frac{2^{16}}{32[\mathrm{MHz}} + \alpha \\ &\approx 2.05[\mathrm{ms}] + \alpha \end{aligned} \tag{4.8}$$

と設定します．α は主に A/D 変換に要する時間です．T_{TMR1} の実測値は約 2.07 [ms] でした．

```
pwm_step ≤ ad_out
```

```
unsigned short      ad_out;
unsigned short      pwm_step = 0;

#define   Interval_of_TMR1_Interrupt   0xFFFF                    // 0.5Hz PWM波形生成用

else if((RC0 == 0) && (RA2 == 1))     // 可視化PWMモード
{
    set_timer1_count_down_ini_num(Interval_of_TMR1_Interrupt);    // タイマ1のカウントダウン値の設定
    clear_interrupt_flag_of_timer1();                             // 次の割り込みを受付け可とする.

    if(pwm_step <= ad_out)            // PWM波形生成
    {
      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 4.9: タイマ 1 割り込み関数_可視化 PWM

の間，RA4(3 番ピン) に 1 (VDD) を出力し，

$$\text{pwm\_step} = 1024$$

で，pwm_step をリセットします．これにより PWM 周期 T_{PWM} は

$$\begin{aligned} T_{PWM} &= T_{TMR1} \times 1024 \\ &\approx 2.1[\text{sec}] \end{aligned} \tag{4.9}$$

となります．

2.2.1 項に実験結果を示してあります．周期が約 2.1 秒なので，可変抵抗器 VR_1 のつまみを回すことで，通流率の変化を LED の点滅間隔の変化で見ることができます．

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

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.