PICマイコンによるDCモータの回転数制御（第3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

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

古橋　武

第3版のまえがき

PIC16F1825 を用いた DC モータの回転数制御の記事を 2012 年 7 月に掲載してから 11 年半が経ちました．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 1 月の現時点でも，訪問者が絶えません．PIC マイコンのよい点の 1 つに，Microchip 社が古い型の製品であっても，決して生産停止にしないことが挙げられます．拙稿の製作例は，2024 年の現在でも製作できます．マイコンによるモータ制御の基礎を学ぶのに，拙稿が今でもお役に立てていると思える根拠です．

第 3 版では，MCC(MPLAB® Code Configurator) 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．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．

拙稿（第 2 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．第 2 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．拙稿第 2 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．

改訂プログラムを本書と同じ Web ページ

モータドライブノート

に掲載します．ご活用下さい．

2024.2

著者記す

第２版のまえがき

PIC16F1825 を用いて DC モータの回転数制御回路を組み，その回路と制御プログラムの解説書を 2012 年から UP してきました．これまでに 2 万回を超える~~ダウンロード~~ヒットがありました．多くの方にお読み（ご覧）いただき，著者として嬉しい限りです．

初版で使用した統合開発環境とコンパイラがすっかり古くなったので，該当箇所を最新のものに書き換えました．初版では統合開発環境に MPLAB IDE を，コンパイラに HI-TECH C® Compiler を使用しました．Microchip 社が無償提供している最新の統合開発環境は MPLAB®X IDE です．また，同社は HI-TECH C Compiler の提供を終了し，MPLAB® XC8 C Compiler を後継コンパイラとして無償提供しています．2019 年 9 月時点のバージョンはそれぞれ v5.25, v2.10 です．そこで，本書の該当箇所を全て MPLAB®X IDE v5.25, MPLAB® XC8 C Compiler v2.10 を用いて書き換えました．

コンパイラの変更に伴い，プログラムを一部修正しなければなりませんでした．改訂プログラムを本書と同じ Web ページ

モータドライブノート

に掲載します．ご活用下さい．

2019.9
2021.2 （再改訂）
2022.5 （再々改訂）
著者記す

初版のまえがき

ブレッドボードでマイコンによるモータ制御回路が組めないかと考えてみた．ブレッドボードはハンダづけを必要としないので，初学者が製作体験をするにはうってつけである．そう考えていた筆者はPIC® マイコンと出会うことができた．PICマイコンにはブレッドボードに差し込めるDIPタイプが多種提供されている．さっそく，PIC12F, 16F, 18F, dsPIC30F, dsPIC33Fの各種タイプについて試してみたところ，いずれも容易に組めることが分かった．ハンダづけ作業を伴わないので回路の組み替えが簡単であり，浮かんだ着想を即座に試すことができる．

世にモータ制御に関する良書は多い．これらは二つのタイプに分類できる．

(1) 理論中心に書かれたもの．

(2) 製作事例集

いずれも重要であるが，両者の間にはギャップがある．理論中心の工学書で学んだ学生は，現実のモータ制御回路をどのように組めばよいのかほとんど分からない．一方，製作事例集では具体的な製作のノウハウは書かれてはいるが，理論との接点がほとんど見られない．

本稿の狙いは以下の通りである．

(1) モータ制御回路の製作事例を紹介する．

(2) 背景となるモータ制御の理論を紹介する．

製作事例で解説するマイコンのプログラムは全て本稿と同じWebページよりダウンロード可能である．本文中にはプログラムを載せた図の図説にフォルダ名をマゼンタ色で示してある．

本稿は大学の電気工学科等にてパワーエレクトロニクスを履修した学生を対象にしている．PWMインバータなどについて一通りのことを教科書と板書による講義を通して学んだ学生に対して，モータ制御という応用問題を提供することを狙っている．本稿にてモータ制御回路の題材を選ぶに当たって，心がけたことは以下の通りである．

(1) 確実に動くこと．

(2) 部品がネット通販により入手できること．

(3) ブレッドボードで（ハンダ付けを極力しないで）作れること．

選んだ題材と用いたマイコンは以下の通りである．

(1) 直流モータの回転数制御
PIC16F1825

(2) 直流モータの電流マイナーループ制御
PIC33FJ128MC802

(3) ブラシレスモータのベクトル制御
PIC33FJ128MC802

なお，PIC16F1825のコンパイラにはMicrochip社が無償で提供しているHI-TECH C® Compiler for PIC 10/12/16 MCUs LITE v9.83 を使用する．HI-TECH Cコンパイラは ANSI Cに準拠したコンパイラであるため，C言語の基本的文法がそのまま適用できる．

しかし，このコンパイラにはマイコン内蔵の各種モジュール（タイマ，A/D 変換，PWM など）を設定するための組み込み関数が（2012 年 7 月時点では）提供されていない．各種モジュール用のレジスタの設定は煩雑で，しかも，データシートと首っ引きでないとできない．本稿では，本稿で必要とするモジュール設定用の関数の作成とその引数に判りやすい表現を採用して，その（表現とレジスタの数値との対応表である）ヘッダファイルの作成についても解説する．

本稿によりインバータ回路等を学んだ学生が，さらにモータドライブに興味を持つことができれば，著者のこの上ない喜びである．

2012.7
著者記す

目 次

第1章　PIC16F1825によるDCモータの回転数制御

1.1　DCモータ回転数制御回路の組み立て

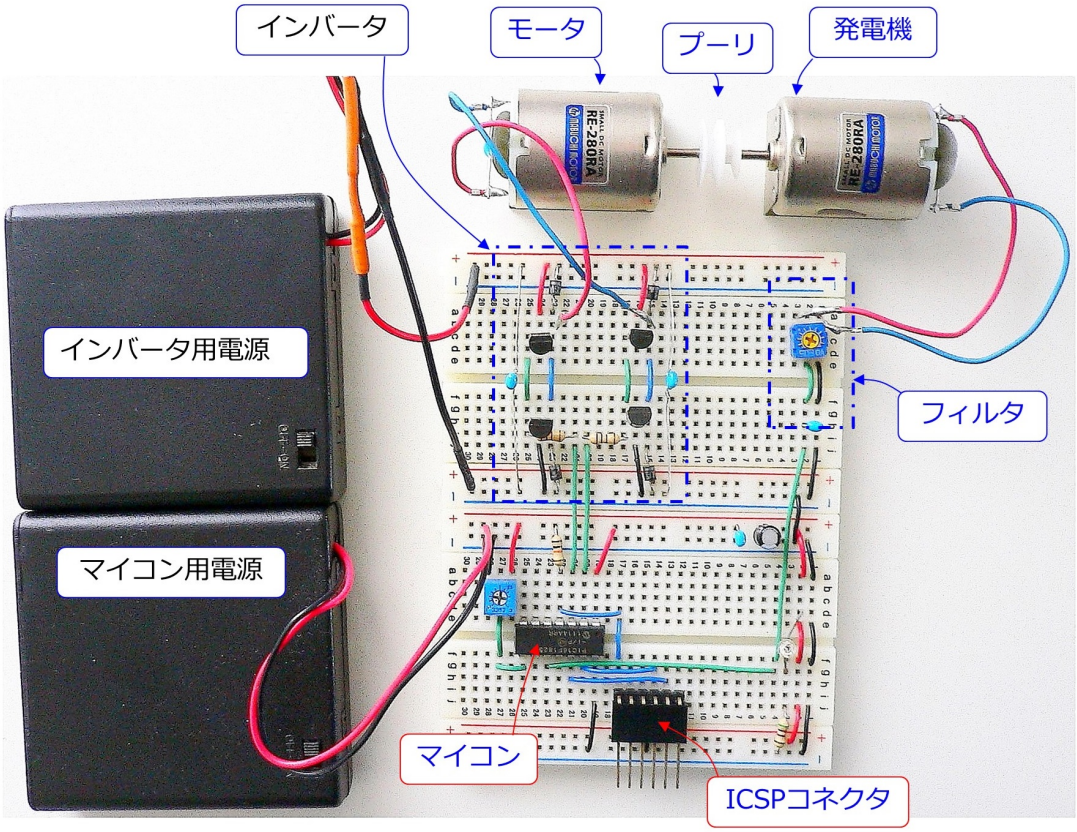


図 1.1: PIC16F1825 を用いた直流 (DC) モータの回転数制御回路（全体写真）

PIC16F1825 を用いた直流 (DC) モータの回転数制御回路の製作例を図 1.1 に示します．制御対象はマブチモータです．この DC モータをインバータにより駆動します．インバータはバイポーラトランジスタを用います．マブチモータをもう一個用意して，モータの軸同士をプーリでつなぎ，発電機として用います．この発電電圧をフィルタを通して回転数検出値としてマイコンに取り込みます．回転数制御に PIC マイコン (PIC16F1825) を用います．マイコン用電源には単 3 の充電池 4 本を用いて 5 [V] の電源とします．インバータ用電源は単 3 の乾電池 4 本 (6 [V])，もしくは充電池 4 本 (5 [V]) を用います．電源を 2 つ

に分けたのは，インバータ回路の発生するノイズによって回転数制御回路の誤動作の可能性を減らすためです．ICSP コネクタにインサーキットデバッガ／プログラマを接続することで，マイコンへのプログラムの書き込みとデバッギングができます．

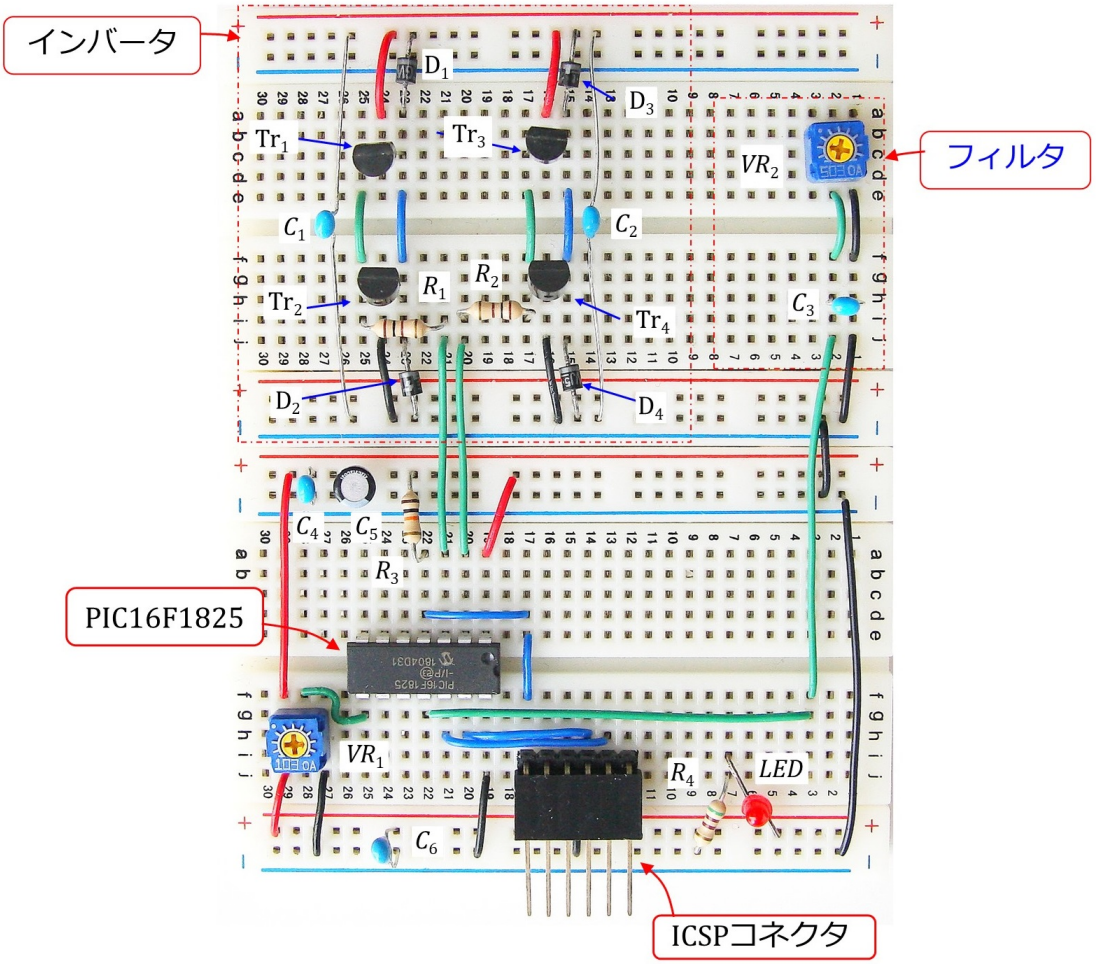


図 1.2: PIC16F1825 を用いた直流 (DC) モータの回転数制御回路（拡大写真）

図 1.2 はブレッドボード上の回路の写真です．写真では配線の様子が分かりにくいので，実体配線図を図 1.3 に示します．

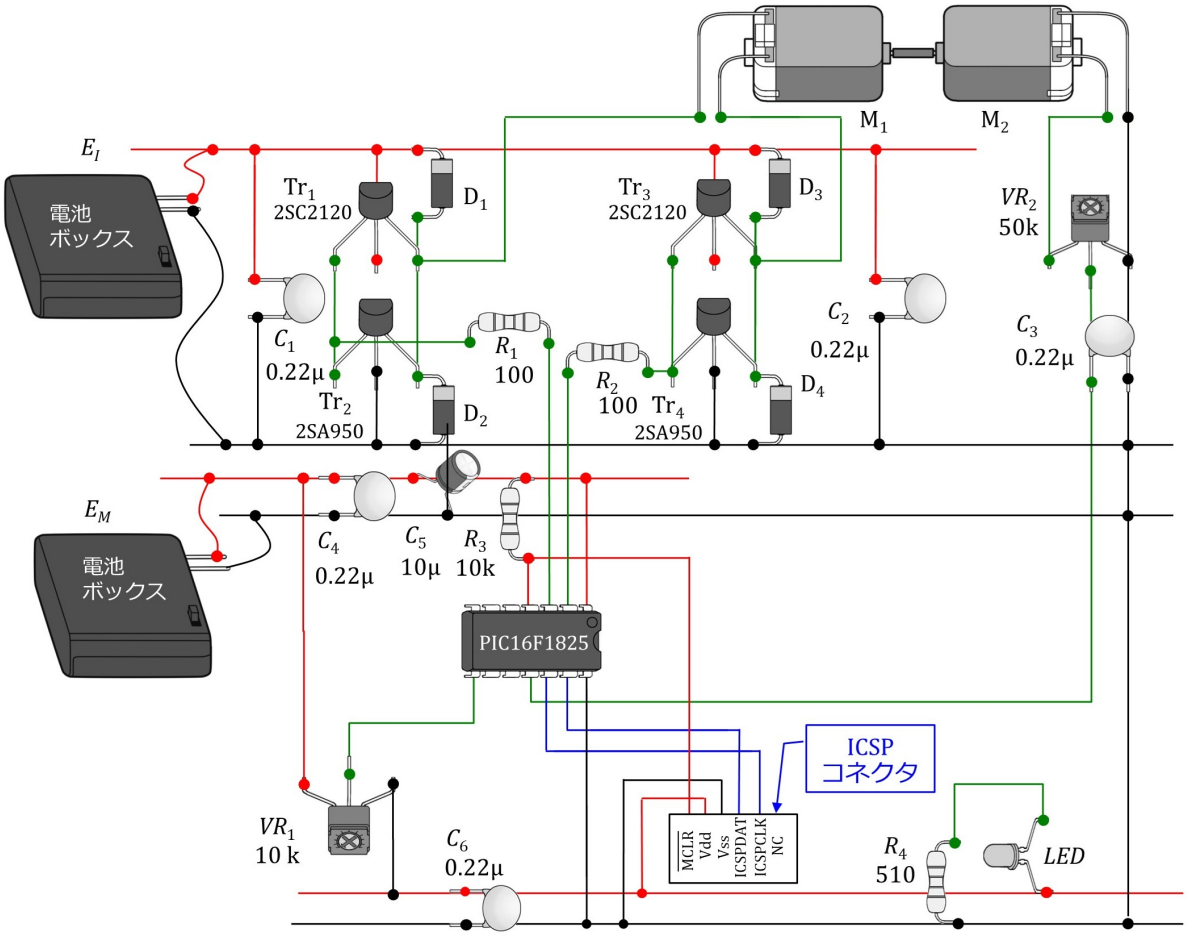


図 1.3: PIC16F1825 を用いた直流 (DC) モータの回転数制御回路実体配線図

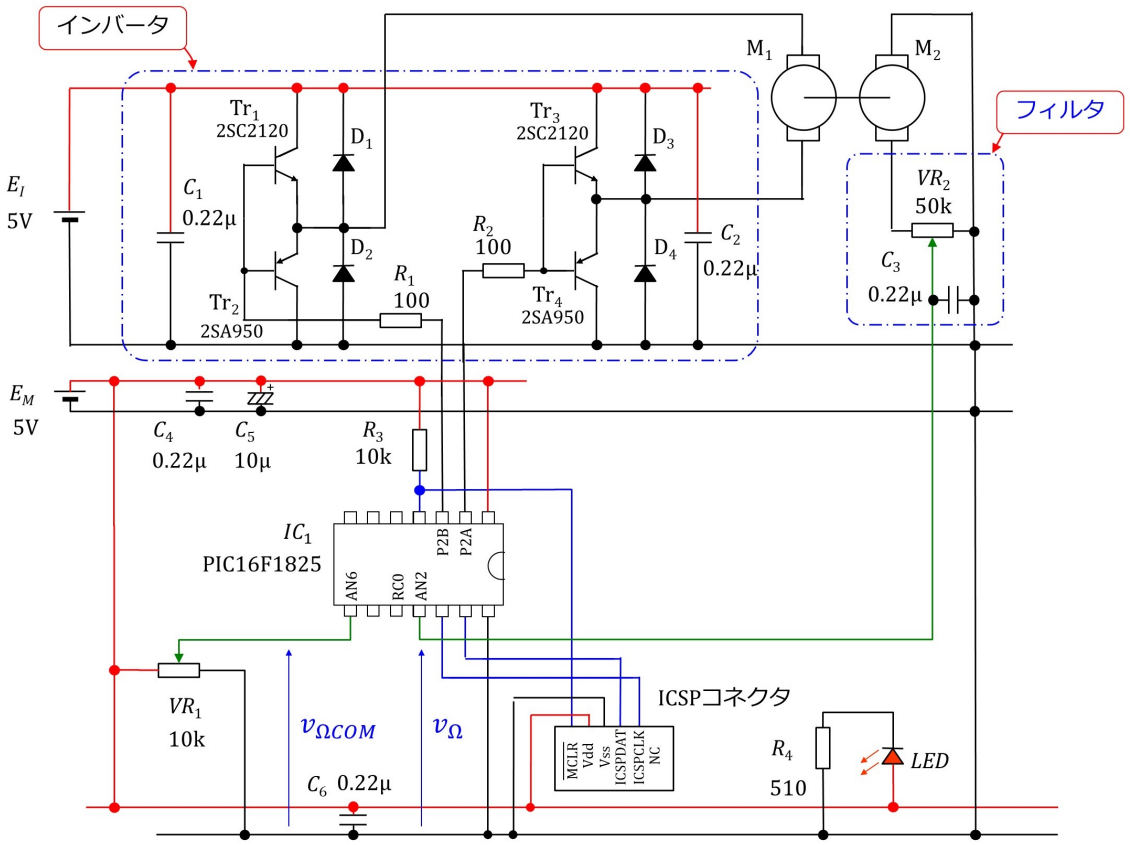


図 1.4: PIC16F1825 を用いた直流 (DC) モータの回転数制御回路図

回路図を図 1.4 に示します．インバータは Tr_1, Tr_3 に NPN 型トランジスタ 2SC2120 を用い，Tr_2, Tr_4 に PNP 型トランジスタ 2SA950 を用います．トランジスタと並列に接続してあるダイオード D_1〜D_4 にはショットキーバリヤダイオードを用います．トランジスタの駆動は PIC マイコンの PWM(Pulse Width Modulation) 制御出力によります．コンデンサ C_1, C_2 はインバータのスイッチング（オン・オフ動作）によるノイズの除去用です．モータ M_2 は発電機として用い，可変抵抗 VR_2 とコンデンサ C_3 はフィルタ回路です．この RC フィルタにより，M_2 の発電電圧に含まれるノイズを除去します．また，コンデンサ C_4, C_6 はノイズ対策用，C_5 は制御電圧安定化用です．

回転数制御回路，インバータ，DC モータ，回転数検出回路などの詳細は以降に詳述します．

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

1.2.1　ICSP コネクタと MPLAB® Snap の接続

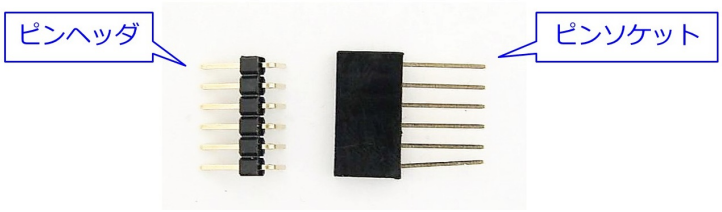


図 1.5: ピンヘッダとピンソケット

1.1 節の DC モータ制御回路のマイコンに DC_Motor_Cont のプログラムを書き込みます．マイコンへのプログラムの書き込みは図 1.2〜1.4 中の ICSP コネクタを介して行います．ICSP コネクタは，図 1.5 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．ピンヘッダは L 型を用います．

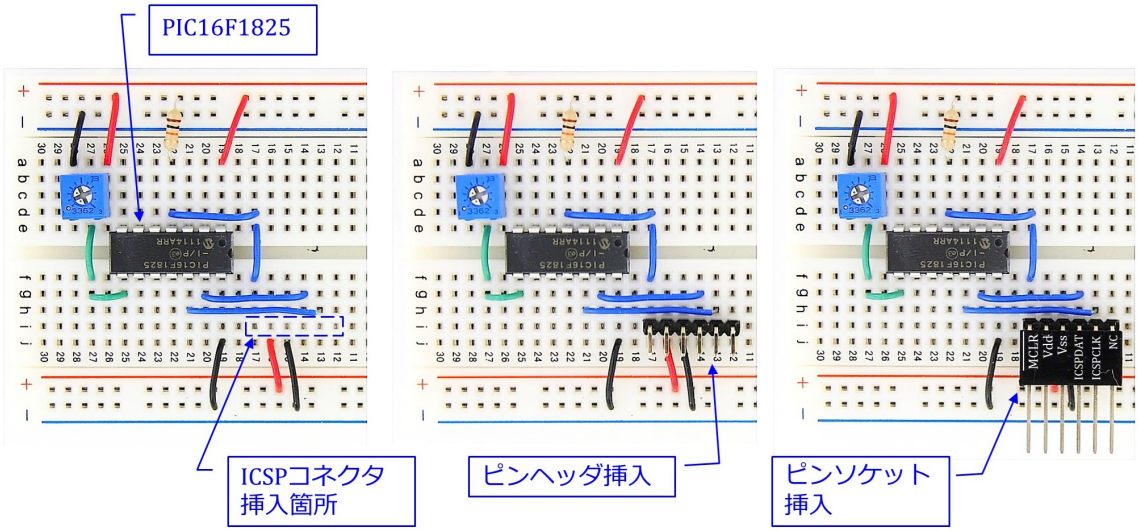


図 1.6: ICSP コネクタの製作

図 1.6 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) がその写真です．このピンヘッダに，同図 (c) のように，ピンソケットを挿入して ICSP コネクタができあがります．ピン・ソケットの上面には，各ピンがつながる先のマイコンの機能を記します．ただし，NC は非接続の意味です．図 1.7 のインサーキット・デバッガ／プログラマ (MPLAB® SNAP) のピン・ソケットのピン配置に合わせてあります．

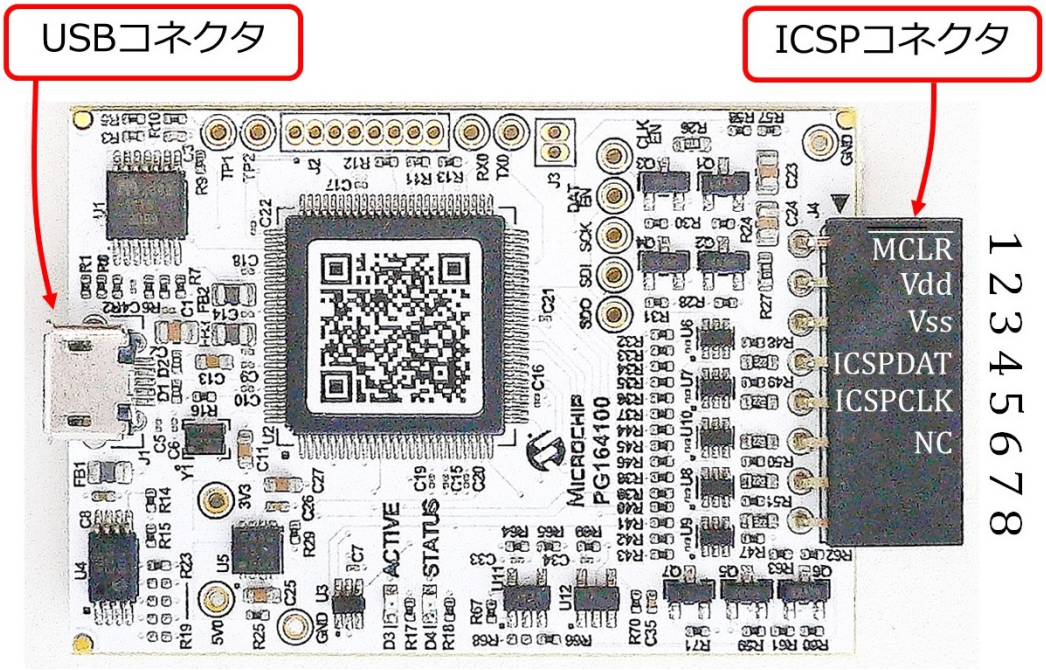


図 1.7: MPLAB® Snap

図 1.7 は MPLAB® Snap の外観とピン配置を示します．ピン穴は 8 個あり，1 ～ 5 番ピンまでが ICSP コネクタ用のメス・ピンです．

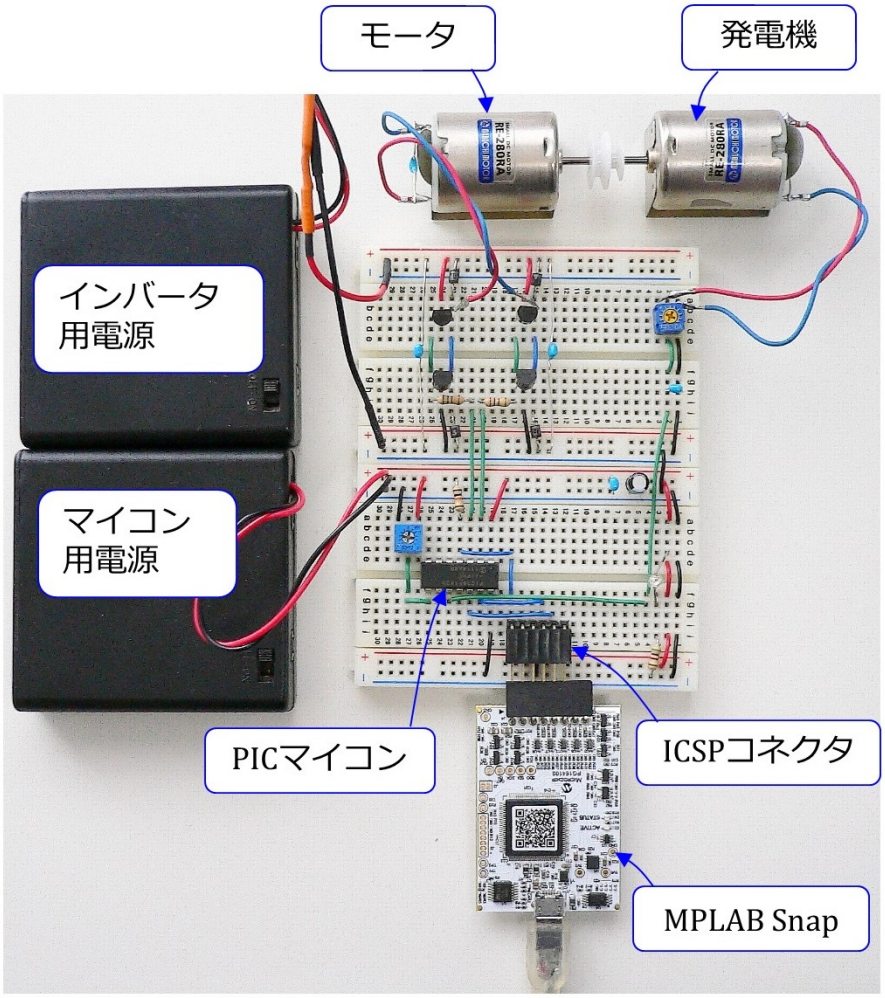


図 1.8: ICSP コネクタに MPLAB® Snap を接続

図 1.8 は ICSP コネクタに MPLAB® Snap を挿入した様子です．写真の向きに見て左端のピンを揃えます．右端の 2 穴は開放のままでよいです．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．この ICSP コネクタには PICkit™4 も，配線を変更すること無く，挿入してプログラムの書き込み／デバッギングができます．

1.2.2　MPLAB® X IDE, XC8 コンパイラのインストール方法

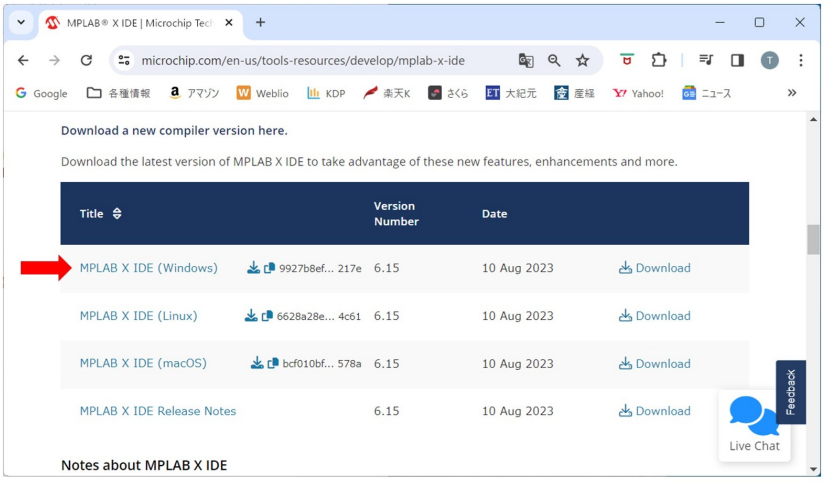


図 1.9: MPLAB®X IDE のダウンロード

本節では 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 → Tools and Resources → Develop → MPLAB® X IDE とたどることで図 1.9 の画面を開くことができます (2024 年 1 月時点)． 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，ユーザー・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．

1.2.3　ダウンロードした Project の読み込み

本稿で解説するソースコードは，本稿と同じ
モータドライブノート
の「PIC16F1825 用 DC モータの回転数制御プログラム（MCC 版）（圧縮ファイル）」に入れてあります．同フォルダをダウンロード・解凍して，「￥DC_Motor_Cont.X」フォルダを MPLABXProjects フォルダ内に置いてください．

「￥DC_Motor_Cont.X」フォルダ内のファイルを用いて，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．また，以下の設定開始前に，図 1.1 の実験回路を作成し，マイコンに 5[V] の電源電圧を印加しておいてください．ただし，インバータ用電源はオフにしておきます．そして，パソコンの USB ポートにインサーキット・デバッガ／プログラマ（例えば MPLAB Snap) をつなぎ，このデバッガ／プログラマをブレッドボード上の ICSP コネクタに挿入してください．

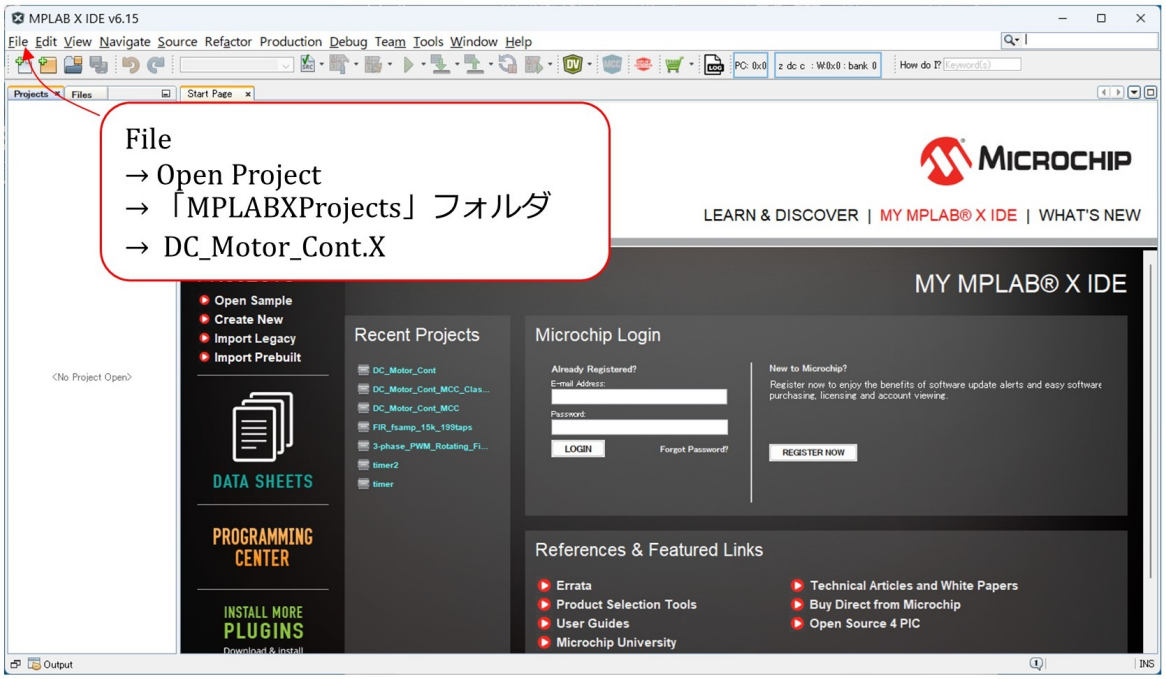


図 1.10: ダウンロードした Prject の読み込み

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 1.10 の画面が立ち上がったら，File → Open Project →「MPLABXProjects」フォルダ → DC_Motor_Cont.X を選択することで，DC_Motor_Cont.X のプロジェクトを MPLAB® X IDE に読み込めます．

図 1.11 は，MPLAB® X IDE に読み込んだ，DC_Motor_Cont 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 def_of_constants.h, limiter.c ファイルからなります．main.c 内に DC モータ制御のための PI 制御関数を書き込みました．

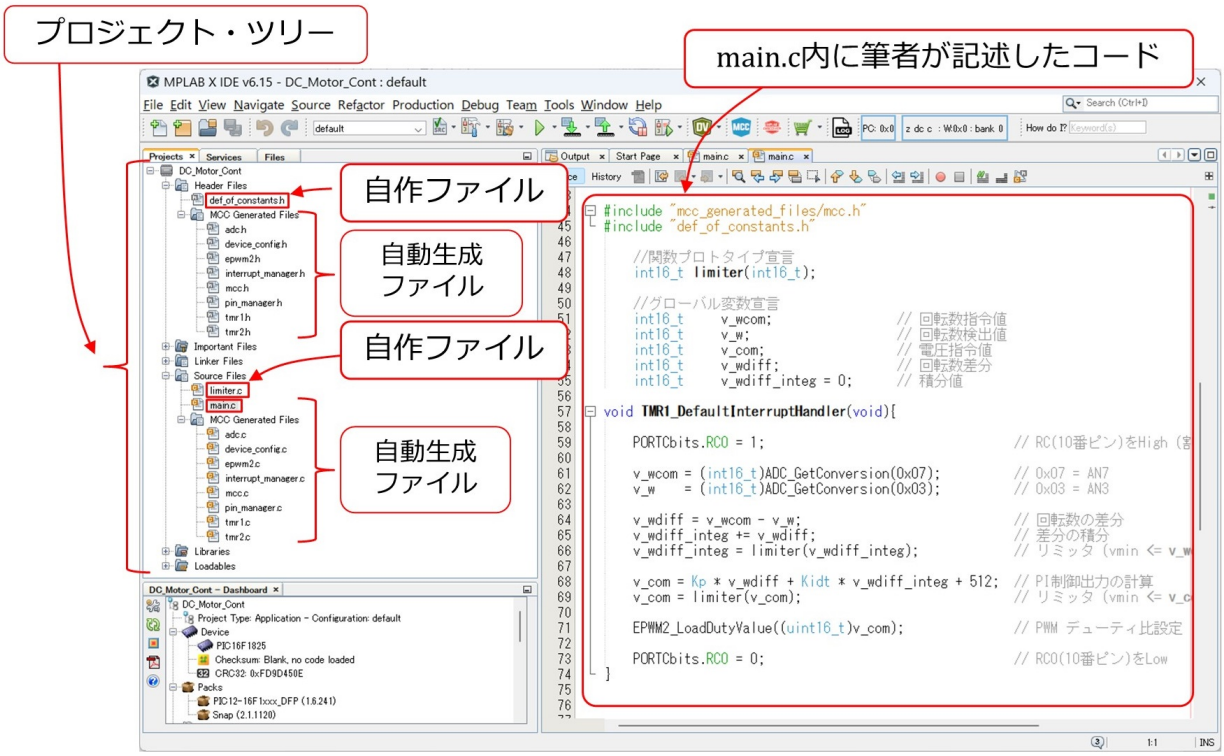


図 1.11: DC_Motor_Cont のプロジェクトツリー

ファイル名を左ダブルクリック（マウスの左ボタンを 2 回クリック）することで，右側のエリアにファイルを開くことができます．図では，main.c 内のコードを開いています．筆者の作成した PI 制御関数です．

なお,「PIC16F1825 用 DC モータの回転数制御プログラム（MCC 版）（圧縮ファイル）」には，以下の 4 つのプロジェクトが入っています．各プロジェクトと，本稿の各節との対応関係は以下の通りです．

- DC_Motor_Cont_TMR1.X → 1.3 節
- DC_Motor_Cont_ADconv.X → 1.4 節
- DC_Motor_Cont_PWM.X → 1.5 節
- DC_Motor_Cont.X → 1.6 節

各節では，白紙の状態からタイマ 1 による割り込み設定 (1.3 節)，AD 変換モジュール設定 (1.4 節)，PWM モジュール設定 (1.5 節)，PI 制御関数の記述 (1.6 節) と，順次プログラムの機能を拡張して行きます．上記プロジェクトは各節終了時点のものです．参考にしてください．

1.2.4　Build とマイコンへの書き込み

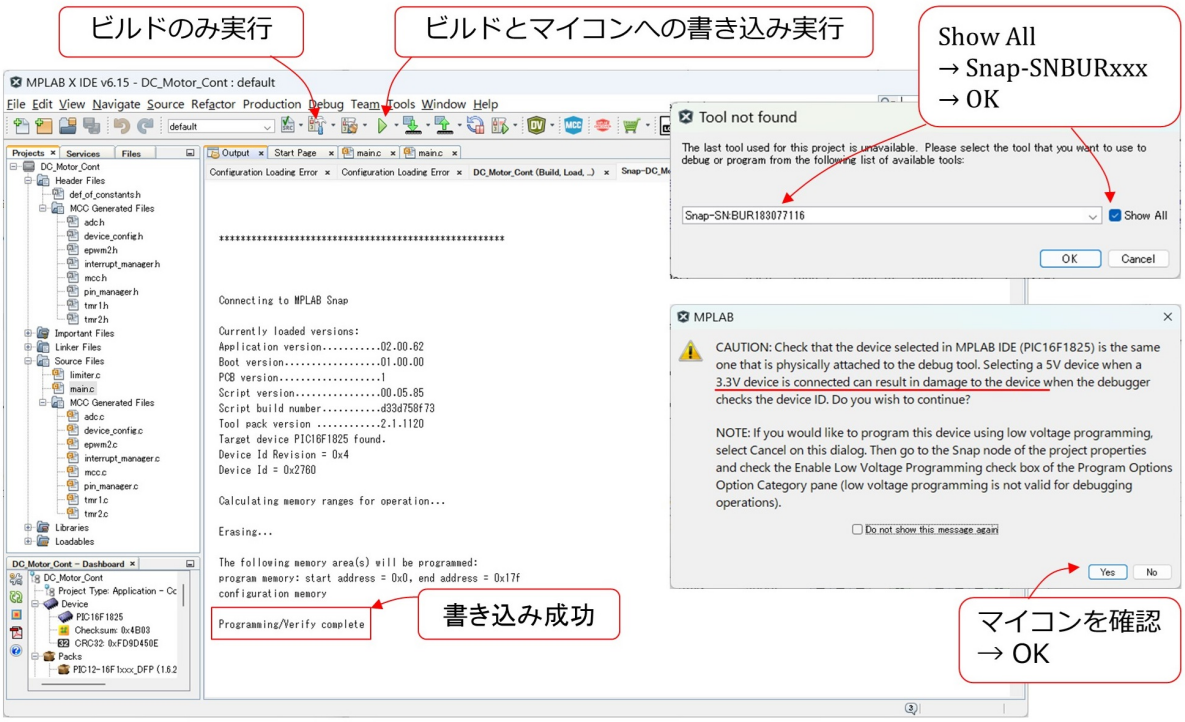


図 1.12: ビルドとマイコンへの書き込み

マイコンに書き込みを行わないでプログラムをデバッグする場合は，ビルド（Build）のみを実行します．図 1.12 の金槌ボタンの上にカーソルを置くと，“Build for Debugging” の文字が表示されます．このボタンを左クリックすることで，ビルドのみを実行できます．文法的なエラーが無ければ，中央エリア内に BUILD SICCESSFUL のメッセージが表示されます．

プログラムをマイコンに書き込む場合は，ブレッドボード上の ICSP コネクタにプログラマ／デバッガを接続し，USB ケーブルでパソコンとこのプログラマ／デバッガをつなぎます．そして，マイコンの電源を入れます．インバータ用の電源は入れないでおきます．図 1.12 の ▷ ボタンの上にカーソルを置くと，“Run Project” の文字が表示されます．この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を実行できます．

Tool not found のメッセージが出たら，Show All にチェックを入れて，プル・ダウン・メニューから Snap-SNBURxxxxx を選択し，OK ボタンをクリックします．

次に，マイコンが PIC16F1825 であることの確認と電源電圧に関する注意喚起 (CAUTION) が表示されます．データ・シートによると PIC16F1825 の電源電圧最大値は 6.5 V です．電源電圧が 5 V であることを確認して，Yes ボタンを左クリックしてください．Programming/Verify complete が表示されれば，書き込み完了です．

マイコン内のプログラムは，書き込み完了と同時に実行されます．インバータ用電源のスイッチを入れるとモータが回り出します．可変抵抗器 VR_1 上面のつまみをネジ回しで

回すと，モータの回転数が変わります．

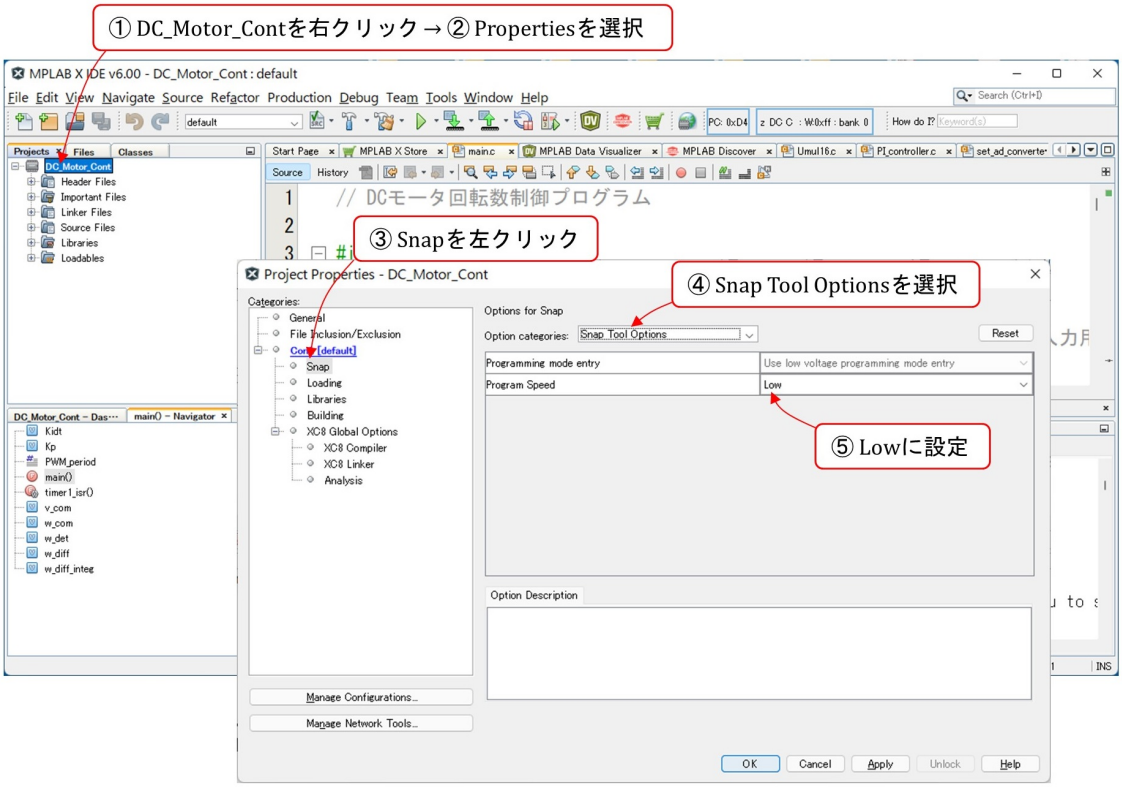


図 1.13: Program Speed を Low に設定する手順

もし，つぎのエラーメッセージが出た場合には

You have set the program speed to Normal. The circuit on your board may require you to slow the speed down. Please change the setting in the tool properties to low and try the operation again.

図 1.13 の手順に従って，Program Speed を Low に設定してください．

1.2.5　New Project の作成方法

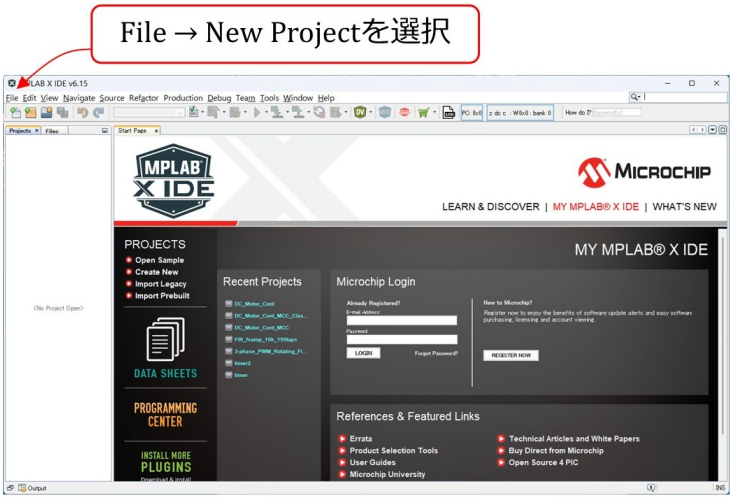


図 1.14: New Project の作成開始

ダウンロードしたプロジェクトではなく，全く新しいプロジェクトを立ち上げる手順について説明します．MPLAB® X IDE のアイコンを左ダブルクリックすることで，統合開発環境を立ち上げられます．図 1.14 の画面が立ち上がったら，File → New Project を選択します．

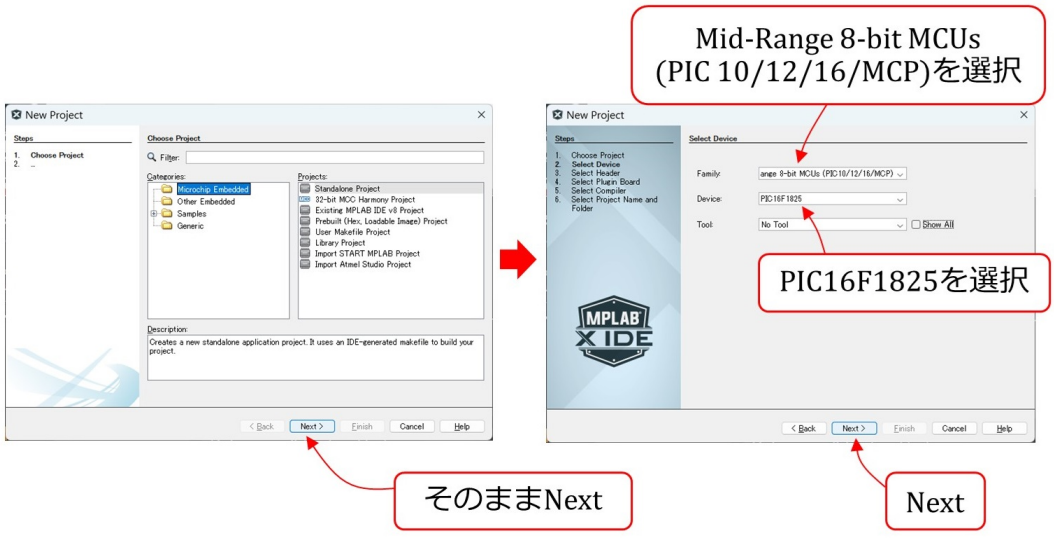


図 1.15: Device 選択

次に図 1.15 のように進み，Device に PIC16F1825 を選択します．

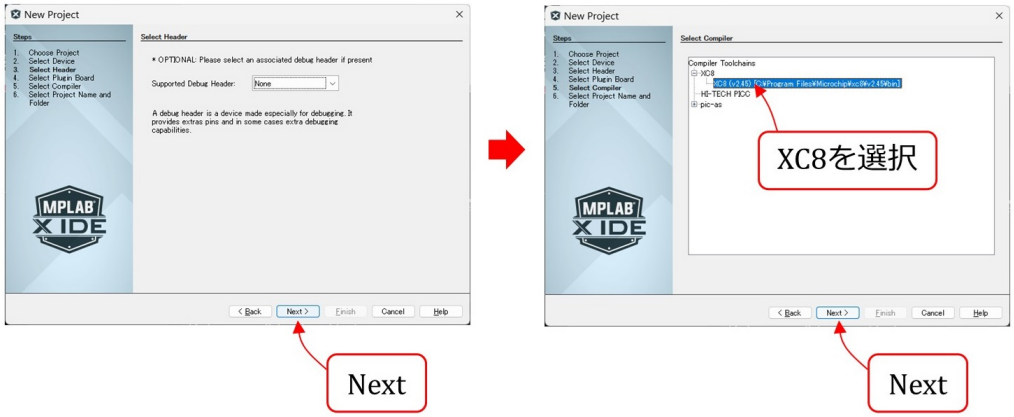


図 1.16: Compiler 選択

その後は図 1.16 のようにコンパイラ (Compiler) に XC8(v2.45) を選択します．

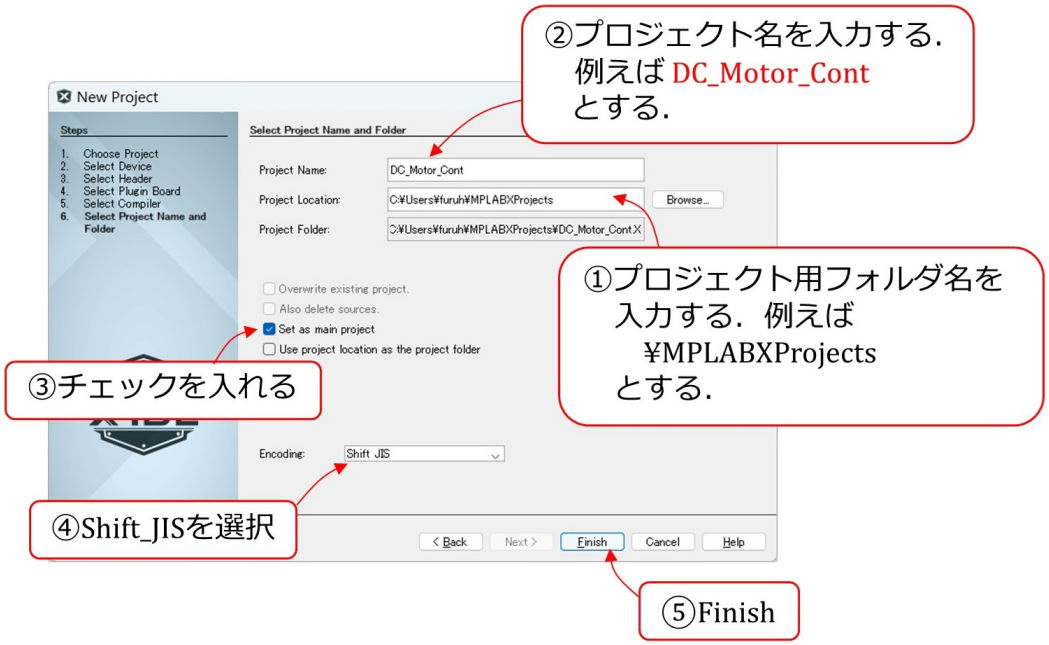


図 1.17: Project Name, コード選択

図 1.17 は次に表示される画面です．Project Location に新しいプロジェクト用フォルダ名を入力します．例えば，¥MPLABXProjects とします．そして，Project Name を，例えば，DC_Motor_Cont とします．文字コード (Encoding) に Shift_JIS を選択して，Finish ボタンをクリックします．

1.3　タイマ 1 による割り込み

1.3.1　タイマ 1 割り込み実験回路

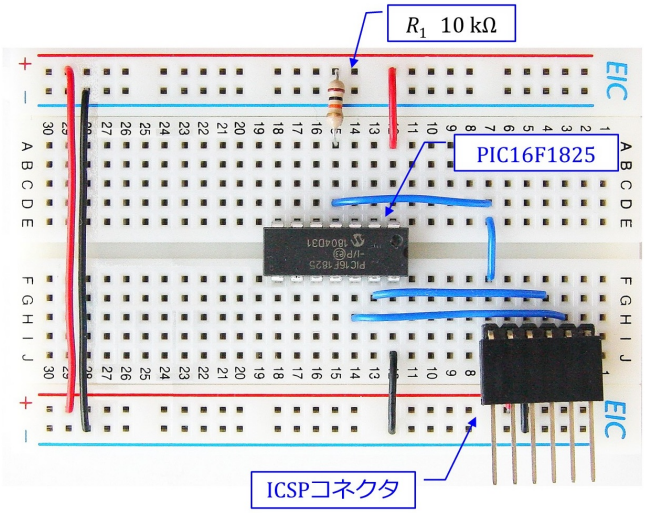


図 1.18: PIC16F1825 を用いたタイマ 1 割り込み実験回路の配線例

回転数制御に必要な PIC マイコン内のモジュールはタイマモジュール，A/D(Analog/Digital) 変換モジュールと PWM モジュールです．本節ではまず，タイマ 1 モジュールの設定方法について解説します．また，マイコンの周辺部品について解説します．

図 1.18 はブレッドボード上に製作した実験回路です．PIC16F1825 を用いて，内蔵のタイマ 1 モジュールによる割り込みの実験を行うための回路です．主な部品は PIC マイコン (PIC16F1825)，ICSP コネクタとブレッドボードです．ブレッドボードは回路製作に際してハンダ付けを要しないため，回路の製作，改変が容易であり，回路の試作に便利です．ICSP コネクタはマイコンへのプログラムの書き込みとデバッグ用であり，本書ではこのコネクタに MPLAB® Snap を接続して，プログラムの書き込みとデバッグを行います．

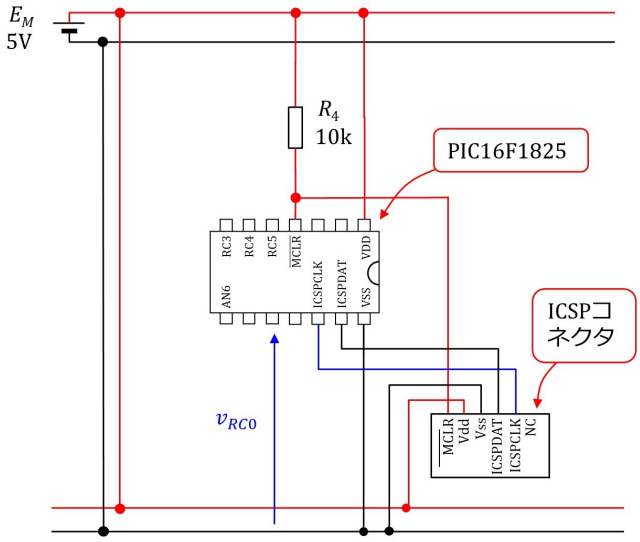


図 1.19: PIC16F1825 を用いたタイマ 1 割り込み実験回路

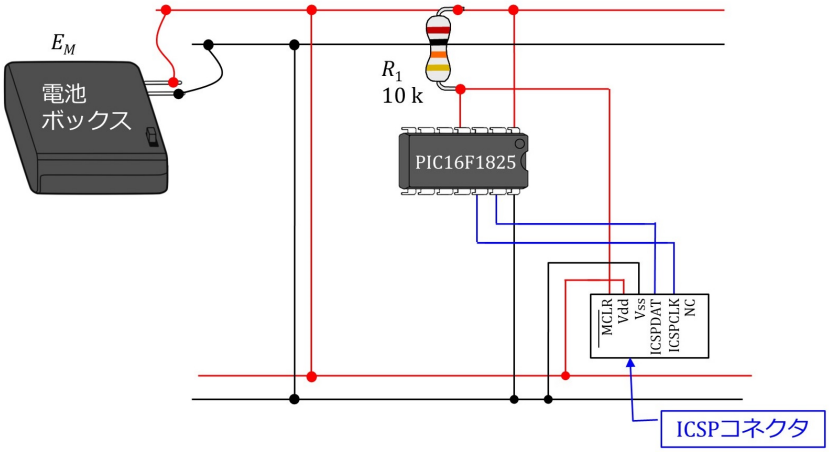


図 1.20: PIC16F1825 を用いたタイマ 1 和鋳込み実験回路の実体配線図

図 1.19 は実験回路の回路図，図 1.20 は実体配線図です．

表 1.1: 部品表

（2024年1月時点）

品名	型式	個数	単価	値段	入手先の例
PICマイコン	PIC16F1825-I/P	1	250	250	秋月電子通商
抵抗	510Ω, 1/4W　100個入り	1	100	100	〃
〃	10kΩ, 1/4W　100個入り	1	100	100	〃
半固定ボリューム	10kΩ	1	50	50	〃
LED	3mm 赤	1	10	10	〃
ピンヘッダ	L型オス, 1×6 (6P)	1	10	10	〃
ピンソケット	1×6 (6P) リード長10mm	1	30	30	〃
ブレッドボード	EIC-801	1	370	370	〃
電池ボックス	単3×4本 リード線・フタ・スイッチ付き	1	160	160	〃
すずめっき軟銅単線	ETFE電線パック6色 11m 導体径 0.51mm	1	580	580	〃
インサーキット　デバッガ／プログラマ	MPLAB SNAP	1	7700	7700	〃

総計　9360　円

表 1.1 にこの実験回路に使用する部品およびデバッガ／プログラマをまとめて示します．表には令和 6 年 1 月時点での部品入手先の例を記してあります．いずれもネット通販です．ただし，値段に送料は含まれていません．電源には充電池 4 本を使用して 5 [V] の電圧を得ます．乾電池 3 本で 4.5[V] としても問題なく動きます．以降，回路内の各部品について順次解説し，その後にタイマモジュールの設定と実験用プログラムについて説明します．

1.3.2　部品

PIC マイコン　- PIC16F1825-

表 1.2: PIC16F1825 の主な仕様

クロック周波数	32MHz
プログラムメモリ	8 kWords
データメモリ	1 kbytes
10ビット AD変換器	8 ch
PWM出力	ハーフ/フル ブリッジ
8/16ビット タイマー	4/1個
電源電圧	1.8～5.5 V

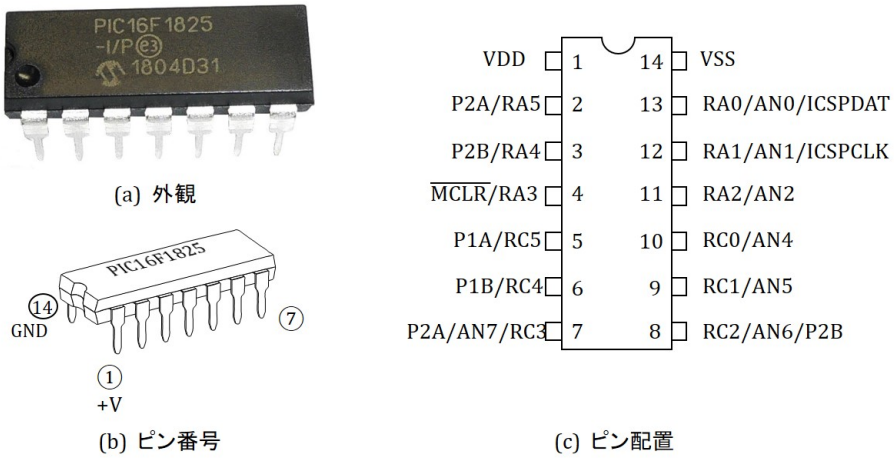
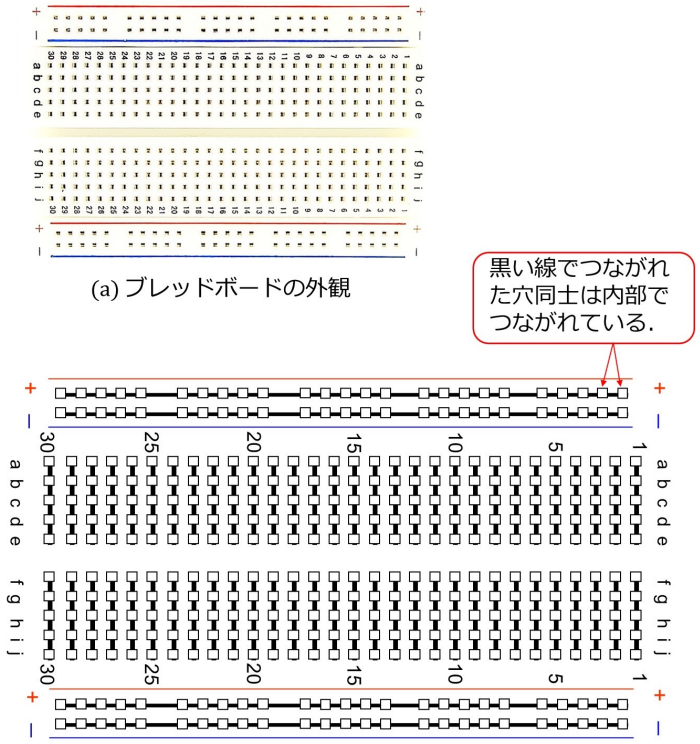


図 1.21: PIC16F1825-I/P のピン配置

PIC マイコン (PIC16F1825) の主な仕様を表 1.2 に示します．PIC16F1xxx シリーズのマイコンは 2010 年から発売されたシリーズです．

図 1.21 には PIC16F1825-I/P の外観とピン番号の見方，およびピン配置を示します．このマイコンは 14 個の電極を持ち，電極はピンと呼ばれます．ピン番号はマイコンを同図 (b) のように見たとき，凹みの左下のピンを 1 番として反時計回りに付けられています．ピン配置図には PIC16F1825 のピンの機能のうち，本章に関係するものを記してあります．VDD に電池の＋側を接続し，VSS に電池の－側を接続します．また，P2A/RA5 のように/記号で併記された機能は，プログラムの設定により使い分けできることを意味します．各機能についてはプログラムの解説の際に記します．詳細は Microchip 社の Web ページから無料でダウンロードできるデータシート “PIC16F1825 Data Sheet” を参照してください．



(a) ブレッドボードの外観

(b) ブレッドボードの穴のつながりの様子

図 1.22: ブレッドボード

ブレッドボード

図 1.22 はブレッドボードの外観とボード内部での穴同士のつながりの様子を示します．このボードでは縦に 10 個並んだ穴が 30 列あり，5 個ずつがそれぞれ内部で接続されています．また，横に 25 個並んだ穴が 4 行あり，各行の穴がそれぞれ接続されています．接続されている穴同士が黒い線でつながれています．図 1.18 の製作回路は一番上の行と下から 2 番目の行を＋電源（電池の＋側）とし，上から 2 番目の行と一番下の行をグラウンド（電池の－側）として使用しています．

抵抗

図 1.23 は抵抗の (a) 外観と (b) 記号および (c) カラーコード表を示します．図は 510 [Ω], 1/4 [W] の抵抗の例です．抵抗値は色の帯で表されます．10 色の色が 0〜9 の数字に対応付けられています．この対応の規則がカラーコードです．憶え方のひとつに「く（黒）ち（茶）あ（赤）だ（橙）き（黄），み（緑）あ（青）む（紫）は（灰）し（白）」があります．すなわち,「くちあ」という滝と「みあむ」という橋があると憶えておくと，おかしな名前の滝と橋ですが，記憶にとどめておくことができます．同図 (a) の抵抗には緑 (5),

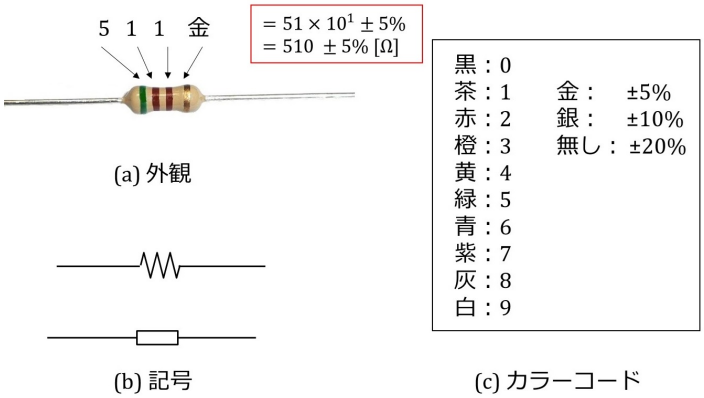


図 1.23: 抵抗

茶 (1)，茶 (1)，金 (±5%) の色の帯が付されています．抵抗値は

$$\begin{aligned} R &= 51 \times 10^1 \ [\Omega] \\ &= 510 \ [\Omega] \end{aligned} \tag{1.1}$$

です．最後の金色はこの抵抗値の精度が ±5% であることを意味します．

1.3.3　タイマ 1 割り込み実験プログラムのブロック図

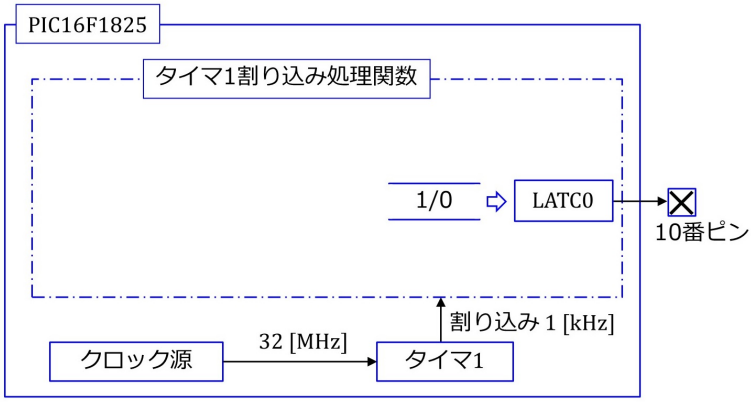
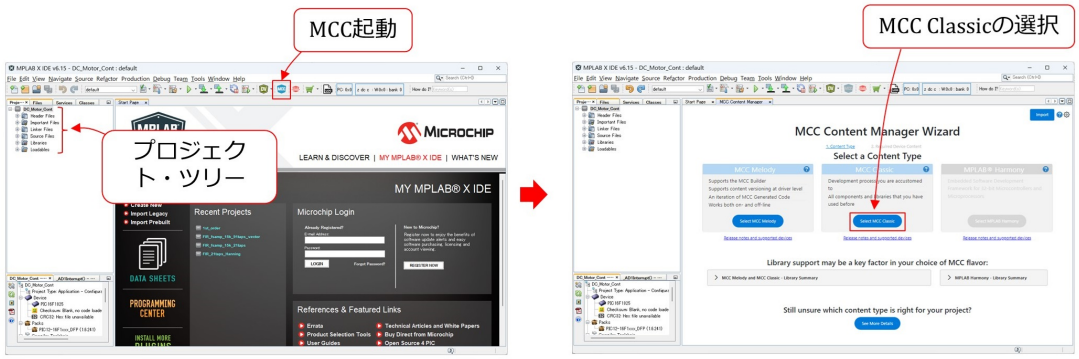


図 1.24: タイマ 1 割り込み実験プログラムのブロック図

図 1.24 はタイマ 1 割り込み実験プログラムのブロック図です．タイマ 1 モジュールはクロック源の 32 MHz のクロックを基に，1 kHz の繰り返し周波数で，CPU に割り込みをかけます．タイマ 1 割り込み処理関数は，割り込みの度に，10 番ピンに “High” の電圧を出力し，直後に “Low” の電圧を出力します．

1.3.4　MCC の起動



(a) MCCの起動　　(b) MCC Classicの選択

図 1.25: MCC の起動

図 1.17 までの手順により，全く新しいプロジェクトの設定が完了すると，図 1.25(a) の画面になります．画面左上には DC_Motor_Cont のプロジェクトツリーが表示されます．この時点では，ツリー内のフォルダは空です．MCC 起動ボタンを左クリックします．図 1.25(b) の画面が現れたら，MCC の Content Type を選びます．PIC16F1825 の場合は MCC Melody と MCC Classic が選べます．令和 6 年 1 月の時点では，MCC Melody の

PIC16F1825 用の機能はまだ十分に整備されていなかったため，MCC Classic を選択しました．Melody は Classic の発展版なので，やがて，使い勝手の良いものへと開発が進むことと推測します．

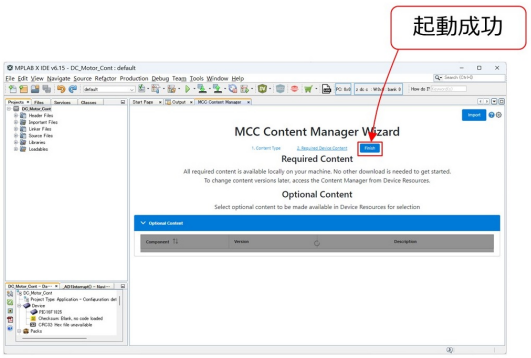


図 1.26: MCC の起動成功

図 1.26 の画面に切り替わったら，Finish ボタンをクリックします．

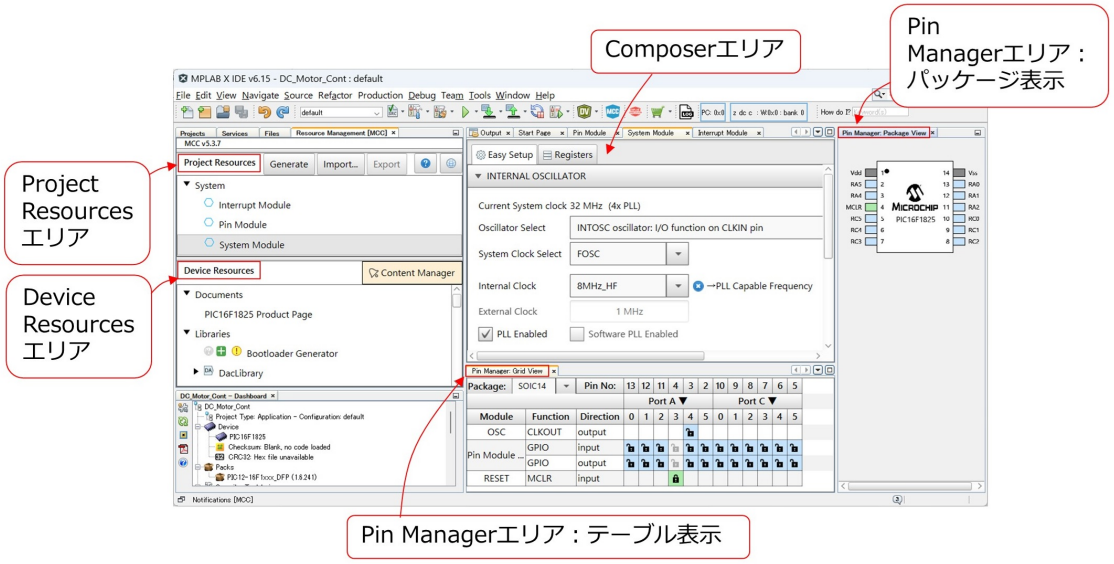


図 1.27: MCC の各エリアの名称

各種周辺モジュールの設定に先立ち，MCC ウィンドウ画面内の各枠の名称を紹介します．図 1.27 に主なものを示します．左上の枠が Project Resources エリア，左中が Device Resources エリア，中央上が Composer エリア，中央下が Pin Manager エリア：テーブル表示，右が Pin Manager エリア：パッケージ表示です．

1.3.5　システムモジュール設定

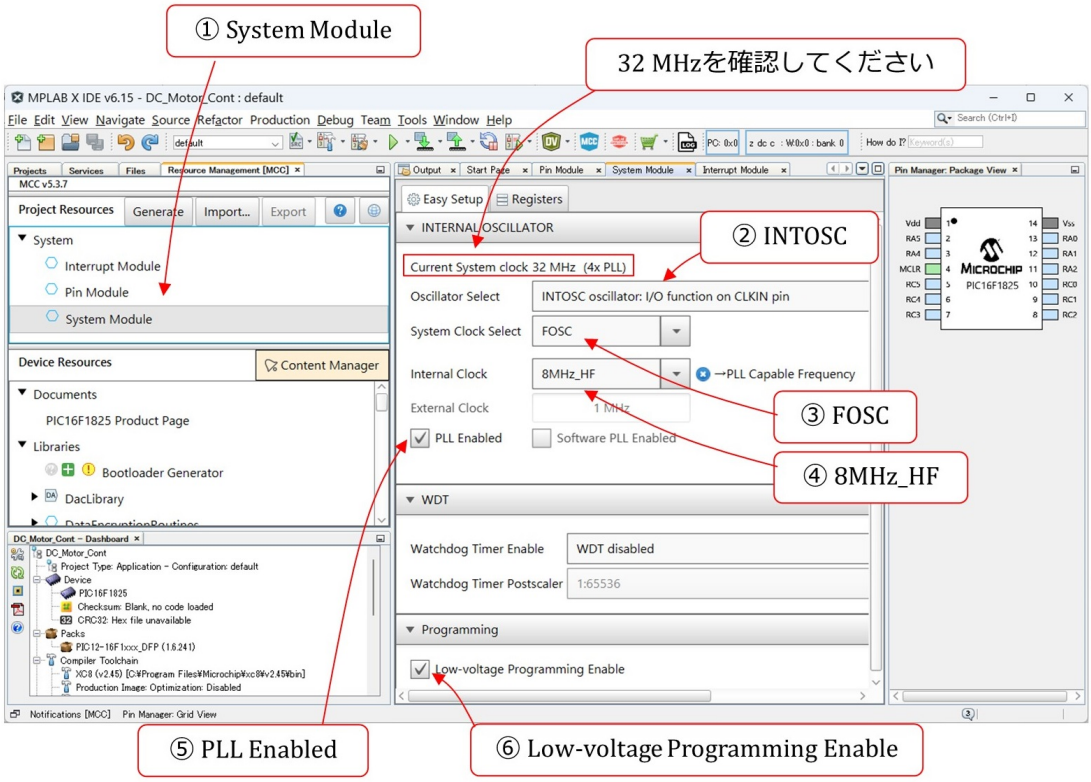


図 1.28: システムモジュールの設定

まず，System Module の設定から始めます．図 1.28 のように，Project Resources エリア内の System Module の文字を左クリックします．すると，Composer エリアに INTERNAL OSCILLATOR 等の設定画面が開かれ，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．選択肢には 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) 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．

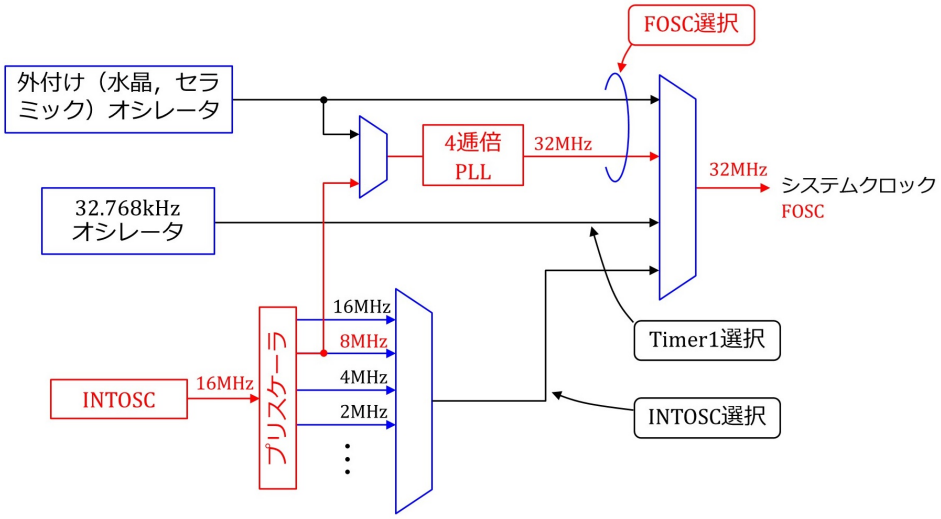


図 1.29: クロックソースのブロック図

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 1.29 は，クロックソースのブロック図です．上記設定により定まった，システムクロックの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，システムクロックは PLL 出力の 32MHz クロックになります．

1.3.6　タイマ 1 モジュールの設定

図 1.30 はタイマ 1 の設定画面です．Device Resources エリアの TRM1 を選択すると，Project Resources エリアに TMR1 の文字が現れ，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．

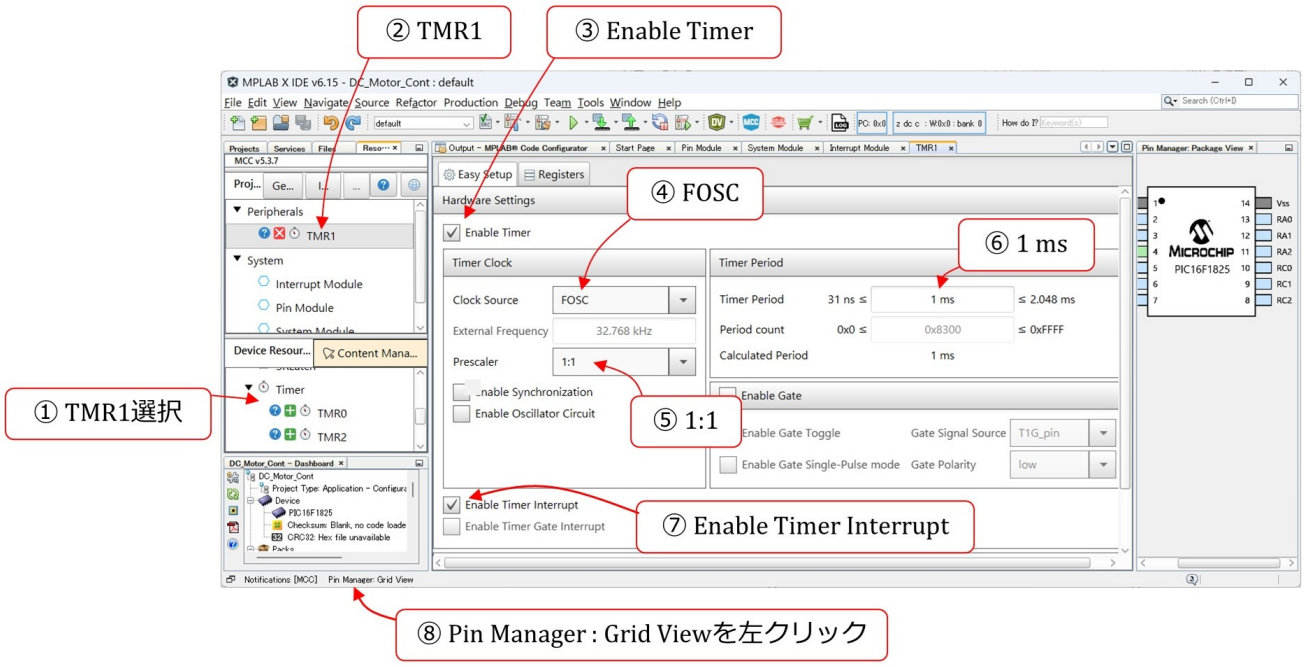


図 1.30: タイマ 1 モジュールの設定

(3) Enable Timer を左クリック

タイマ 1 ・モジュールを起動します.

(4) 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.

(5) 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.

(6) 1 ms に設定

タイマ 1 による割り込み周期を 1 [ms] に設定します．TMR1 レジスタには，初期値として 0x8300 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

$$\begin{aligned} 2^{16} - 0x8300 &= 2^{16} - 8 \times\ 16^3 - 3 \times\ 16^2 \\ &= 32000 \end{aligned} \tag{1.2}$$

です．FOSC = 32 MHz なので，オーバーフローまでの所要時間は 1 ms です.

(7) 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.

(8) Pin Manager: Grid View を左クリック

Pin Manager エリア：テーブル表示を開きます．

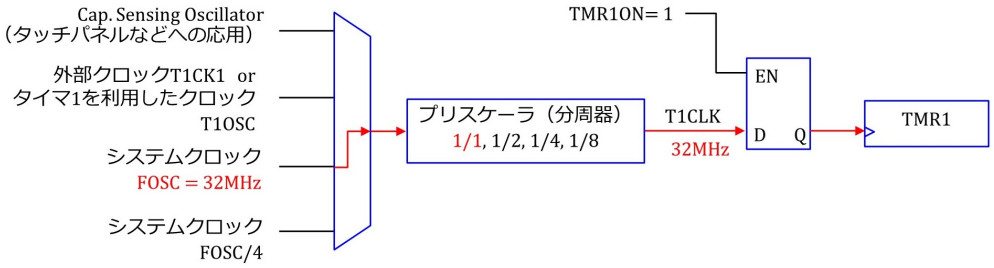


図 1.31: タイマ 1 のクロック源のブロック図

図 1.31 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．

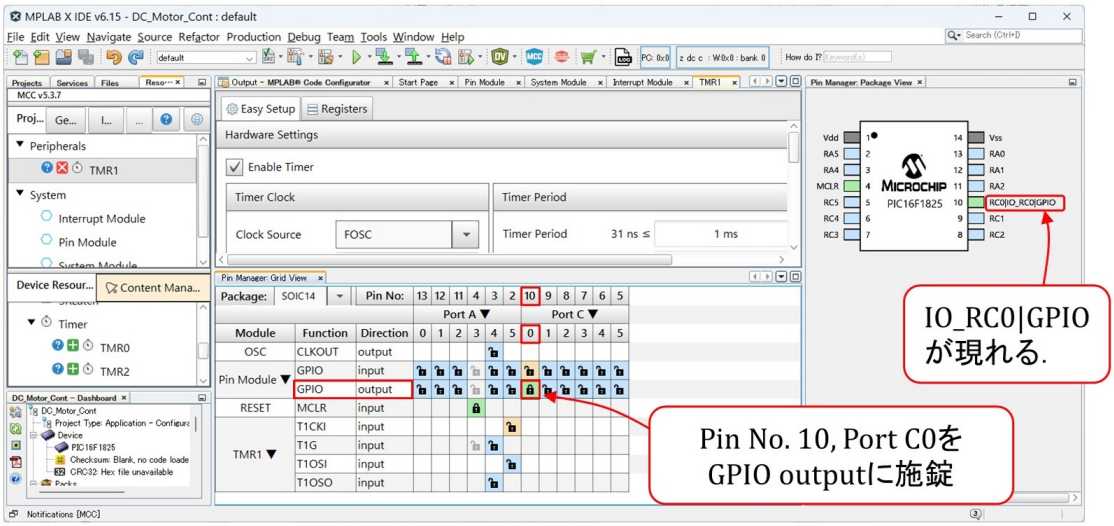


図 1.32: PORT C0(10 番ピン) をディジタル出力 RC0 に設定

図 1.32 は，PORT C0(10 番ピン) をディジタル出力 RC0 に設定している画面です．タイマ 1 による割り込みが実際に起きているかをモニタするために，10 番ピンに “High/Low” 信号を出力するための準備です．上記手順のステップ 8 で，Pin Manager エリア：テーブル表示を開きます．そして，PORT C0(10 番ピン) の列において，GPIO (General Purpose Input/Output) output のグリッドを施錠します．すると，右の Pin Manager エリア：パッケージ表示において，10 番ピンに IO_RC0|GPIO の文字が現れます．

1.3.7　タイマ 1 モジュール設定関数の自動生成

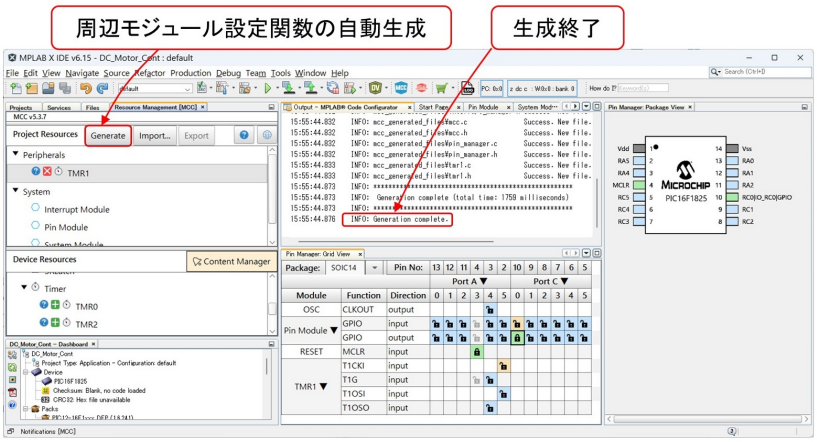


図 1.33: 周辺モジュール設定関数の自動生成

前項にて，タイマ 1 モジュールの設定を終えたので，タイマ 1 モジュール設定関数の自動生成を実行します．図 1.33 はその自動生成を行った画面です．Project Resources エリアの Generate ボタンを左クリックします．中央上の Composer エリアに “Generation complete.” の文字が現れれば，生成終了です．

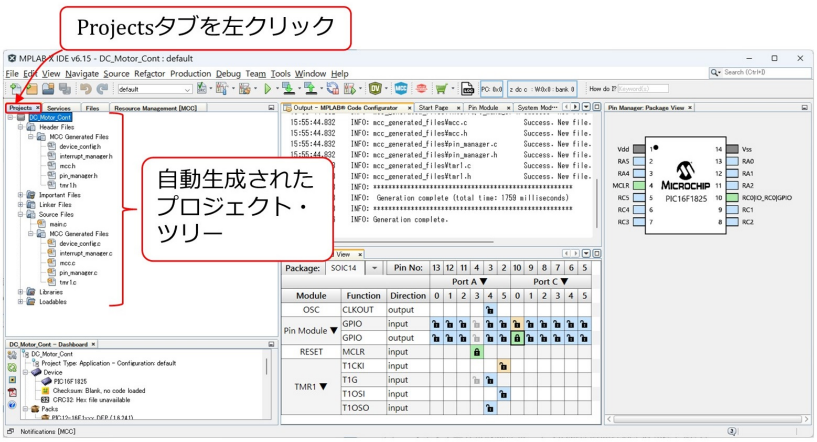


図 1.34: 自動生成されたプロジェクトツリー

図 1.34 のように，Project Resources エリアの Projects タブを左クリックすると，同エリア内に自動生成されたプロジェクトツリーが表示されます．

1.3.8　追加の設定

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 1.35 は，GIE と PEIE を

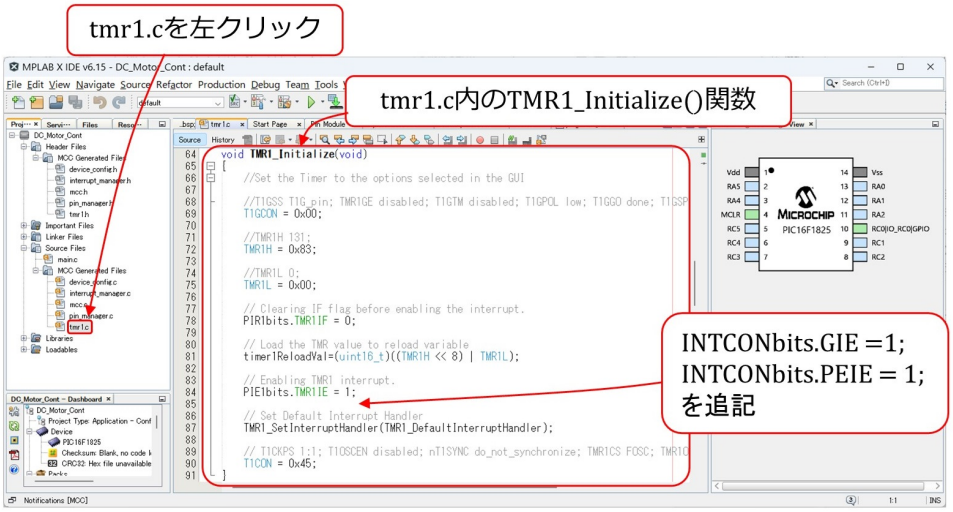


図 1.35: 追加の設定 (GIE, PEIE)

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

$$\begin{aligned} \text{INTCONbits.GIE} &= 1; \\ \text{INTCONbits.PEIE} &= 1; \end{aligned} \tag{1.3}$$

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．

1.3.9　タイマ 1 割り込み処理関数の記述

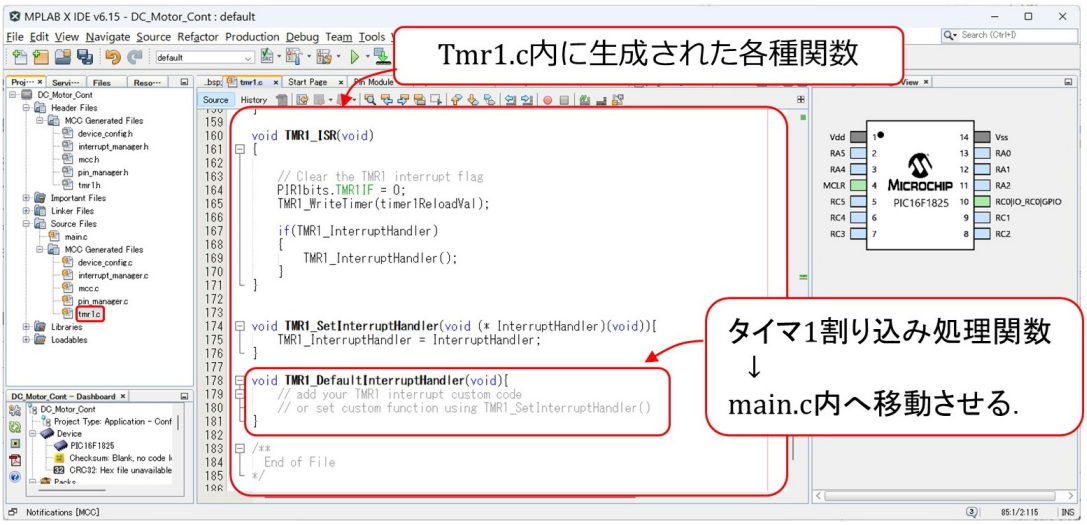


図 1.36: tmr1.c 内のタイマ 1 割り込み処理関数

タイマ 1 割り込み処理関数の「ひな形」が，MCC により自動生成されています．図 1.36 は，tmr1.c 内に自動生成されたタイマ 1 割り込み処理関数 (TMR1_DefaulInterruptHandler() 関数) です．図 1.35 の TMR1_Initialize() 関数を下へスクロールしていくと，この関数が現れます．以下に抜き書きします．

```
void TMR1_DefaultInterruptHandler(void){
  // add your TMR1 interrupt custom code
  // or set custom function using TMR1_SetInterruptHandler()
}
```

コメントにあるように，この割り込み処理関数内にユーザ独自のコードを記述できます．ユーザ独自の割り込み処理関数を設定することもできます．また，関数の名称を TMR1_DefaultInterruptHandler から，別の名前に変えることもできます．名称変更は，tmr1.h ファイル内の 1 箇所，tmr1.c ファイル内の 2 箇所にある TMR1_DefaultInterrupt-Handler を書き換えることでできます．

この関数を main.c ファイルに移動させ（切り取り & ペーストし）ます．

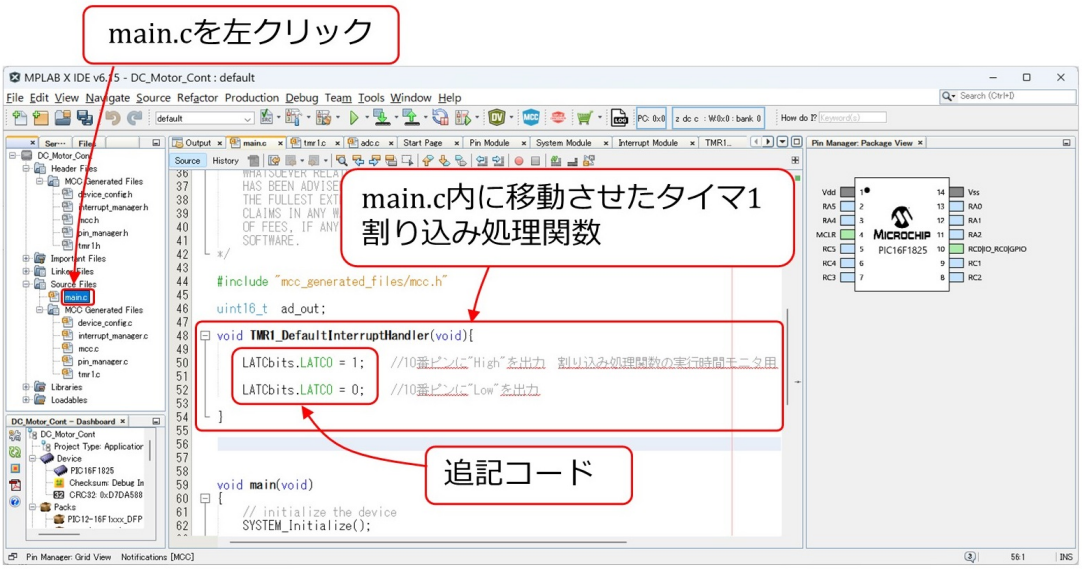


図 1.37: main.c ファイル内の TMR1 割り込み処理関数

図 1.37 は，main.c ファイル内に移動させたタイマ 1 割り込み処理関数です．関数名は，MCC により自動生成された TMR1_DefaultInterruptHandler() 関数をそのまま用いています．

```
void TMR1_DefaultInterruptHandler(void){
    LATCbits.LATC0 = 1; //10 番ピンに”High”を出力
    LATCbits.LATC0 = 0; //10 番ピンに”Low”を出力
}
```

割り込み処理内容は，LATC0 = 1 により，10 番ピンに “High” を出力し，直後に LATC0 = 0 により，“Low” を出力するだけです．タイマ 1 による割り込み処理が，図 1.30 のステップ 6 にて設定した，割り込み周期 1 [ms] で実行されることの確認を目的としています．

1.3.10　ビルド

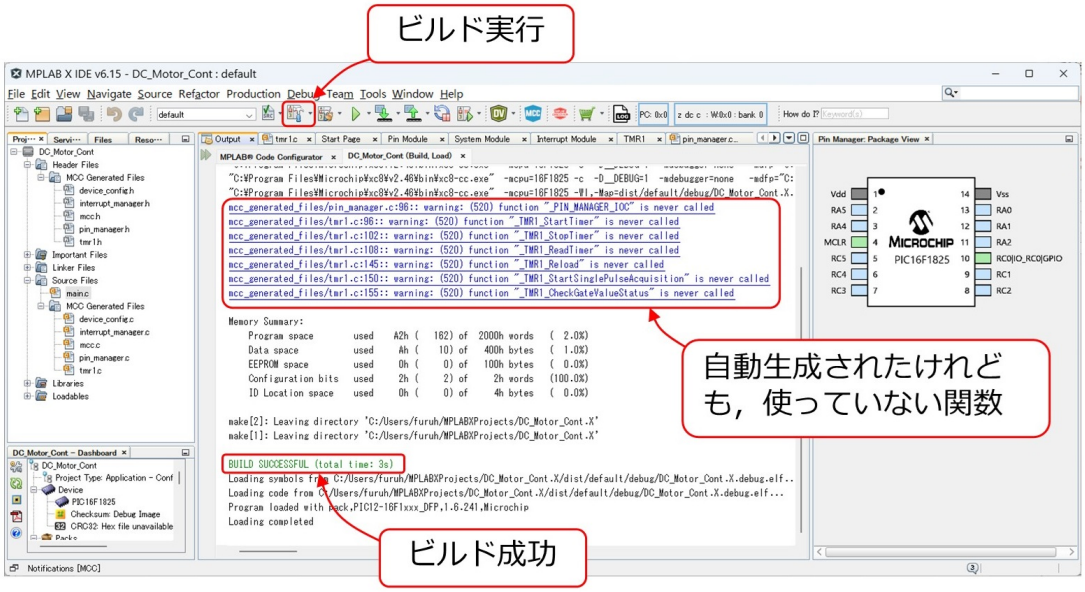


図 1.38: TMR1 割り込み処理プログラムのビルド

タイマ 1 割り込み処理関数の設定と記述ができたので，ビルドを実行します．マイコンをつながない状態でプログラムをデバッグする場合は，ビルドだけを行います．図 1.38 はビルドの実行の様子を示します．金槌ボタンの上にカーソルを置くと，”Build for Debugging”の文字が表示されます．このボタンを左クリックすることで，ビルドのみを実行できます．文法的なエラーが無ければ，中央エリア内に BUILD SICCESSFUL のメッセージが表示されます．

途中に，function “xxx‘’ is never called のメッセージが表示されています．これらの関数は MCC により自動生成された関数です．生成されはしましたが，筆者が使用していないので警告としてこれらのメッセージが出されています．プログラムの実行に支障は無いので，このまま放置しておきます．

1.3.11　ビルドとマイコンへの書き込み

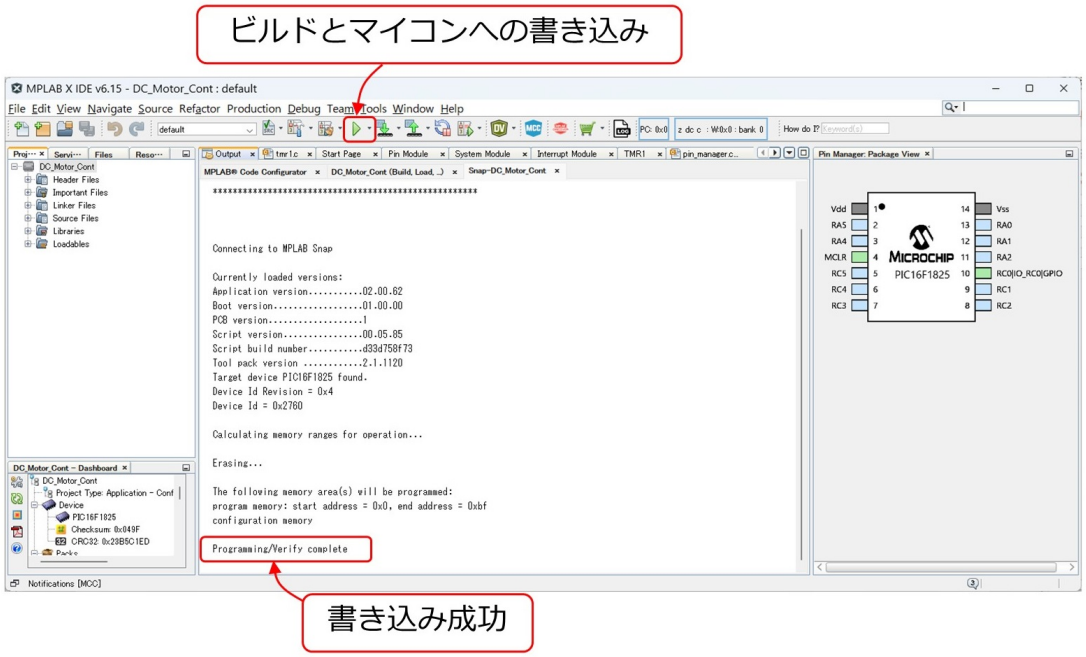
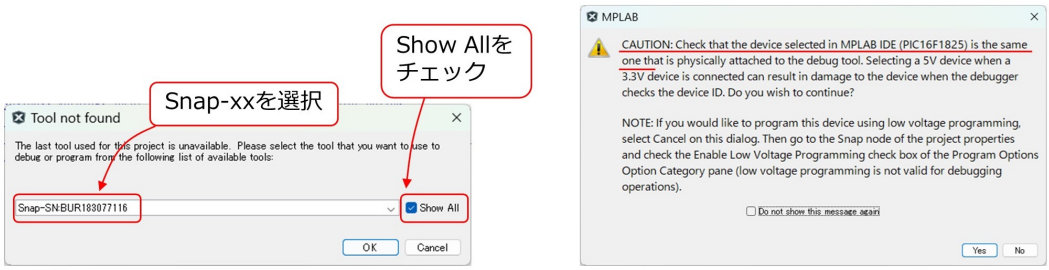


図 1.39: TMR1 割り込み処理プログラムのビルドとマイコンへの書き込み

文法的ミスの無いことがわかったので，いよいよマイコンへのプログラムの書き込みです．ブレッドボード上の ICSP コネクタにプログラマ／デバッガを接続し，USB ケーブルでパソコンとこのプログラマ／デバッガをつなぎます．そして，マイコンの電源を入れます．図 1.39 の▷ボタンの上にカーソルを置くと，”Run Project ”の文字が表示されます．この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を実行できます．



(a) プログラマ／デバッガの指定要求　　(b) マイコン接続の確認要求

図 1.40: 書き込みツールの指定要求とマイコン接続の確認要求

図 1.40 のように，Tool not found のメッセージが出たら，Show All にチェックを入れて，プルダウンメニューから Snap-SNBURxxxxx を選択し，OK ボタンをクリックします．次

に，マイコン (PIC16F1825) の接続確認と電源電圧に関する注意喚起 (CAUTION) が表示されます．データ・シートによると PIC16F1825 の電源電圧最大値は 6.5 V です．電源電圧が 5 V であることを確認して，Yes ボタンを左クリックしてください．Programming/Verify complete が表示されれば，書き込み完了です．

マイコン内のプログラムは，書き込み完了と同時に実行されます．

1.3.12　タイマ 1 割り込み処理関数の実行結果

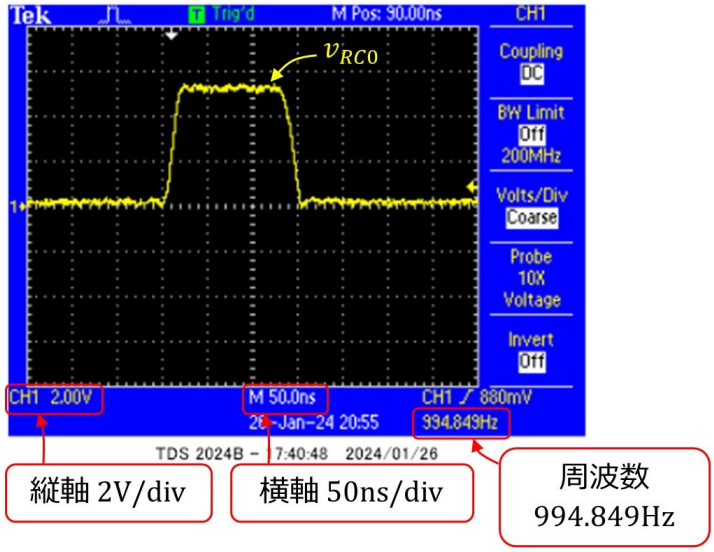


図 1.41: タイマ 1 による割り込みの実験波形

図 1.41 は，タイマ 1 割り込み処理プログラムを PIC16F1825 に書き込み･実行させたときの，RC0(10 番ピン）の電圧波形です．テクトロにクス社製のオシロスコープ画面のスナップショットです．縦軸は 2 [V/div]，横軸は 50 [ns/div] です．このオシロスコープは周波数カウンタの機能も持っていて，計測結果が画面右下に表示されています．繰り返し周波数は 994.849 [Hz] でした．図 1.30 のステップ 6 にて設定した，割り込み周期 1 [ms] より少し長い周期となりました．また，タイマ 1 割り込み処理関数 (TMR1_DefaultInterruptHandler()) 関数の LATC0 =1 により “High” が出力されてから，LATC0 = 0 により “Low” となるまでの時間は約 150 [ns] でした．

1.4　A-D 変換

1.4.1　A-D 変換実験回路

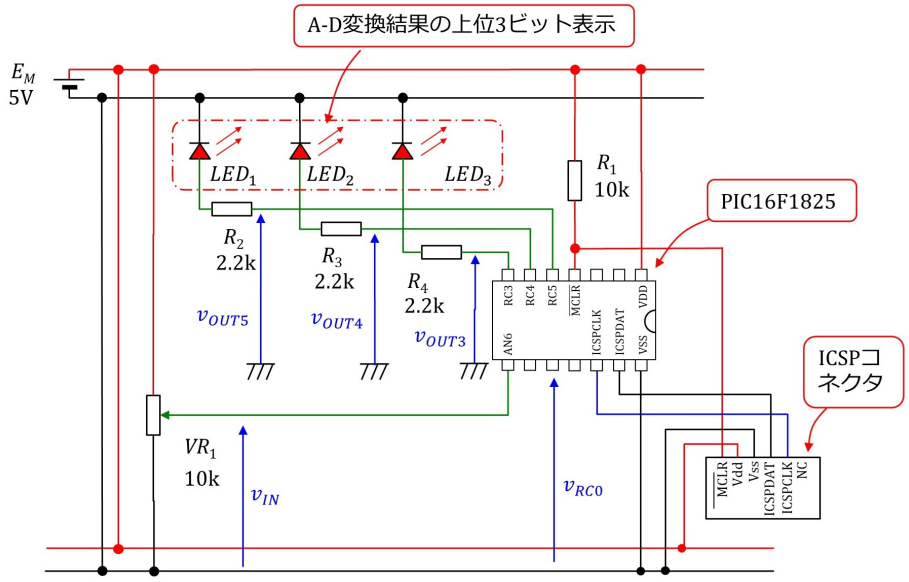


図 1.42: A-D 変換実験回路

図 1.42 は A-D 変換を実験するための回路です．図 1.19 の回路に，可変抵抗 VR_1，抵抗 $R_2 \sim R_4$ と発光ダイオード $LED_1 \sim LED_3$ を追加しました．

可変抵抗により電圧 v_{IN} を変えます．マイコン内部では 8 番ピンに A-D 変換モジュールをつないで，v_{IN} を読み込みます．そして，プログラムにより，A-D 変換結果の上位 3 ビットを，最上位ビットから順に LED_1, LED_2, LED_3 に表示します．

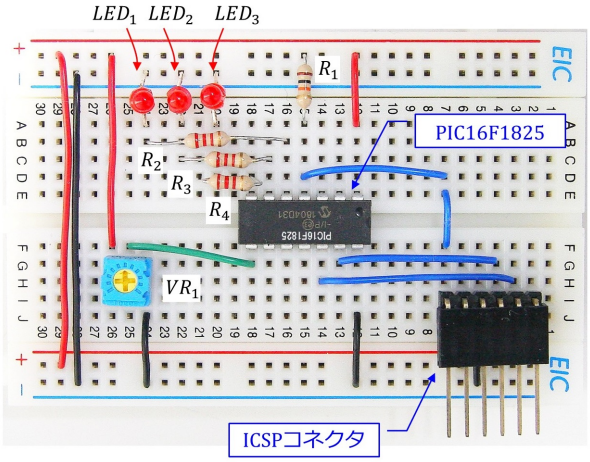
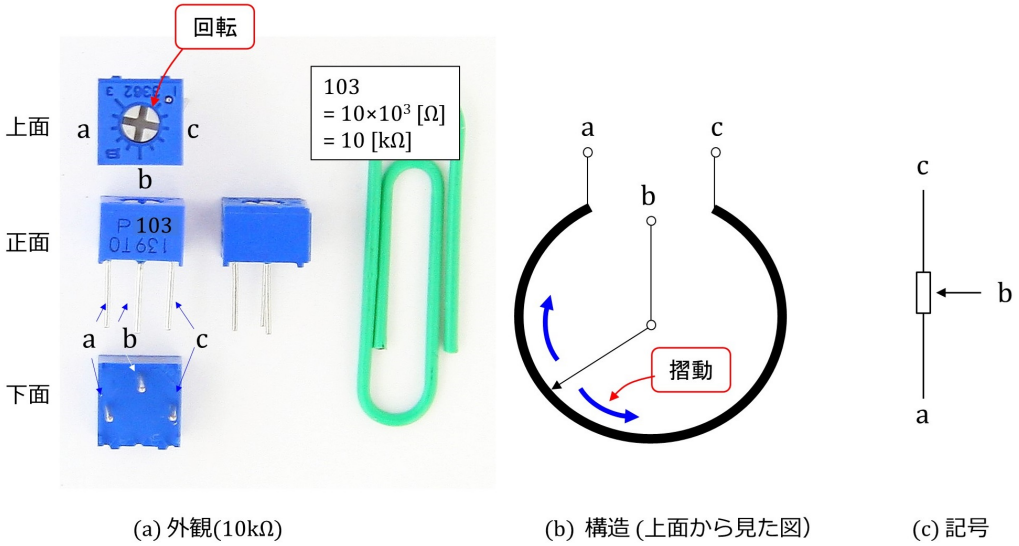


図 1.43: A-D 変換実験回路の配線例

図 1.43 は，A-D 変換実験回路の配線例です．

1.4.2　部品

可変抵抗



(a) 外観(10kΩ)　(b) 構造 (上面から見た図)　(c) 記号

図 1.44: 可変抵抗

図 1.44 は可変抵抗の (a) 外観と (b) 内部構造および (c) 記号を示します．図は 10 [kΩ] の可変抵抗の例です．抵抗正面に印字された 103 の数字は抵抗値です．この値を R とすると，

$$\begin{aligned} R &= 10 \times 10^3 \ [\Omega] \\ &= 10 \ [\mathrm{k}\Omega] \end{aligned} \tag{1.4}$$

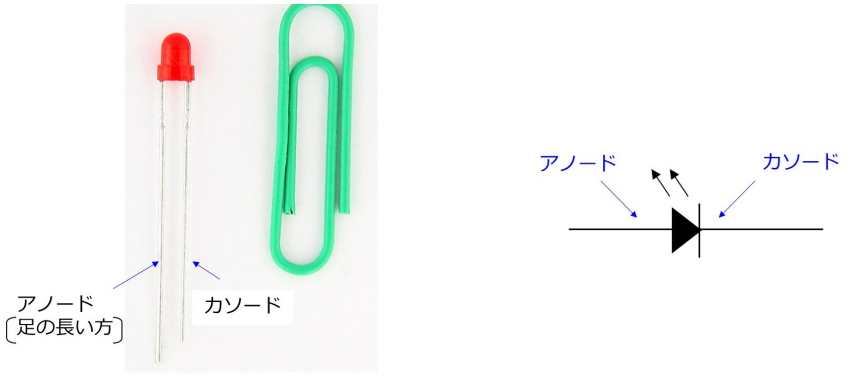
です．この数字が 511 であれば

$$\begin{aligned} R &= 51 \times 10^1 \ [\Omega] \\ &= 510 \ [\Omega] \end{aligned} \tag{1.5}$$

です．

この可変抵抗には電極が 3 つあり，それぞれ a, b, c の記号を付けてあります．同図 (b) のように，a, c 電極は円上の抵抗線に接続され，b 電極が可動部に接続されています．a-c 間の抵抗値は 10[kΩ] です，電極 b は，抵抗上面の＋の溝のついた可動部分を回転させることで，抵抗線上を摺動します．これにより a-b 間，b-c 間の抵抗値をそれぞれ 0〜10[kΩ], 10〜0[kΩ] の範囲で変化させられます．

図 1.19 ではマイコンの 8 番ピンに可変抵抗 VR_1 の b 電極が接続されています．．したがって，マイコンの 8 番ピンには電源電圧の 5[V] から 0[V] の範囲の電圧が印加されます．後述するように，マイコンの内部では 8 番ピンには A/D 変換モジュールの入力端子が割り当てられます．



(a) LED (発光ダイオード)の外観　(b) LED(発光ダイオード)の記号

図 1.45: LED（発光ダイオード）

LED（発光ダイオード）

図 1.45 は LED（Light Emitting Diode: 発光ダイオード）の (a) 外観と (b) 記号です．写真は円筒部分の直径が 3mm の LED の例です．足の長い電極がアノードと呼ばれ，短い電極がカソードと呼ばれます．

図 1.42 の回路では，v_{OUTx} =“High” $(x = 3, 4, 5)$ のときに LED が点灯します．赤色 LED の通電時のアノード・カソード間電圧（オン電圧と呼ばれる）は約 1.9 [V] です．LED を明るく光らすには数 [mA] の電流を流す必要があるので，抵抗値 R_2 は

$$R_2 \approx \frac{5 - 1.9[\mathrm{V}]}{6[\mathrm{mA}]} = 520 \quad [\Omega] \tag{1.6}$$

と求められます．

1.4.3　A-D 変換実験用プログラムのブロック図

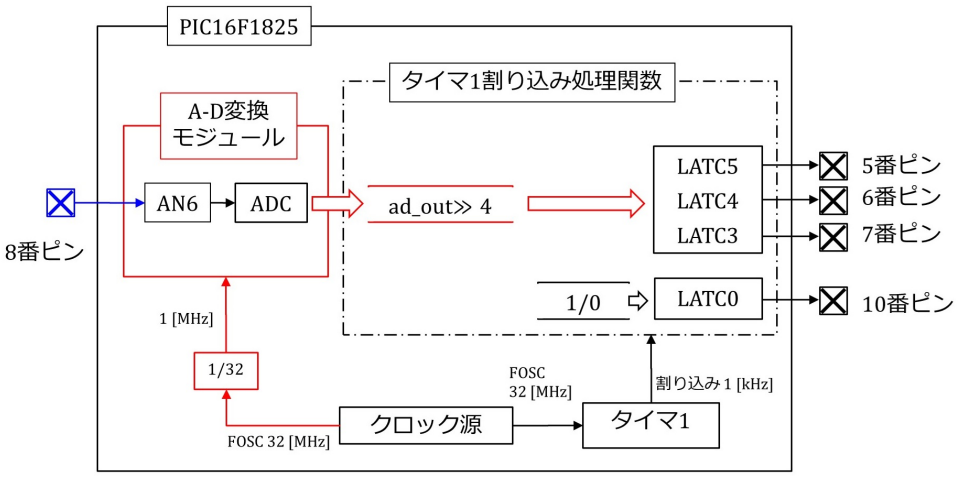


図 1.46: A-D 変換実験用プログラムのブロック図

図 1.46 は A-D 変換実験用プログラムのブロック図です．図 1.24 のタイマ 1 割り込み実験用プログラムのブロック図に，A-D 変換モジュールを追加し，タイマ 1 割り込み処理関数内には，A-D 変換結果 (10 ビット) の上位 3 ビットを 5 ～ 7 番ピンに出力するコードを追加しています．PIC16F1825 の A-D 変換モジュールでは，動作クロックの上限が 1 MHz なので，システムクロックの 32 MHz を 1 ／ 32 に分周して 1 MHz を得ています．

1.4.4　A-D 変換モジュールの設定

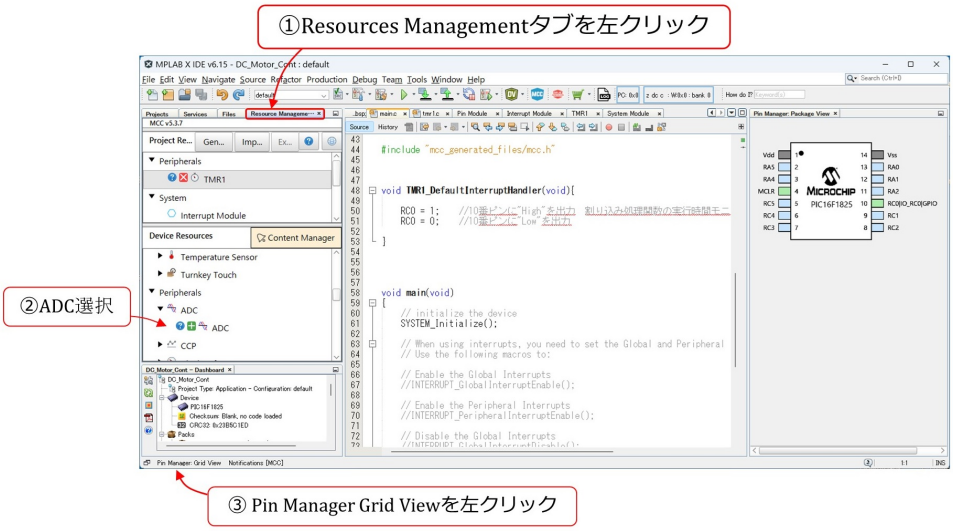


図 1.47: A-D 変換モジュールの選択

前節の DC_Motor_Cont のプロジェクトに A-D 変換モジュールを追加します．図 1.47 は，A-D 変換モジュールを選択している画面です．Resources Management タブを左クリックすると，Project Resources と Device Resources のエリアを再開できます．Device Resources エリアの ADC を左ダブルクリックします．ADC の下にもう 1 つ ADC の文字が現れます．＋ボタンを左クリックします．

また，ウィンドウ左下の Pin Manager Grid View を左クリックして，ピンテーブルを表示しておきます．もし，Pin Manager Grid View の文字が見当たらない場合は，メニューの Window から MPLAB Code Composer v5 → Show All により表示できます．

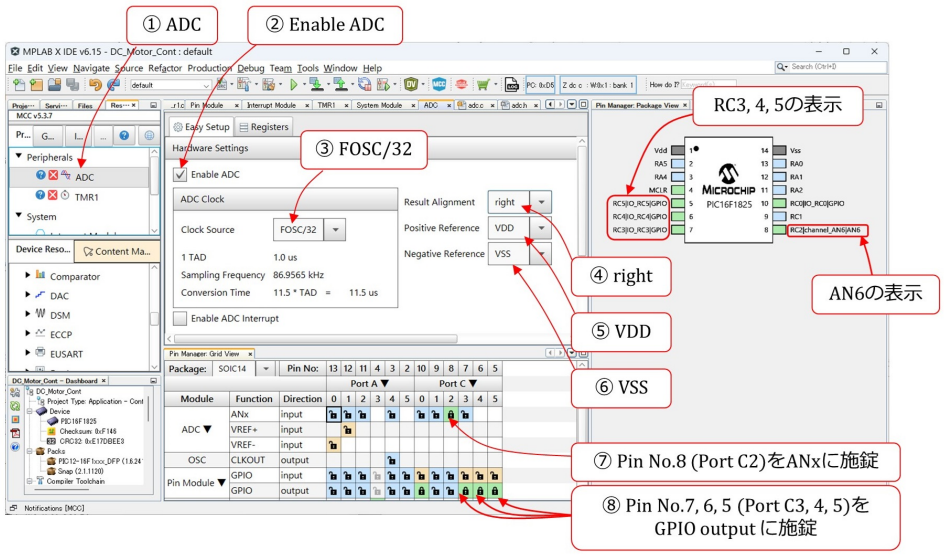


図 1.48: A-D 変換モジュールの設定

図 1.48 のように，Project Resources エリアの Peripherals 欄に ADC の文字が表示され，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，設定を行ってください．

(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 No. 8(Port C2) の ANx 欄を左クリック

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．

(8) Pin No. 7, 6, 5(Port C3, 4, 5) の GPIO output 欄をそれぞれ左クリック

A-D 変換結果を 7, 6, 5 番ピン (Port C3, 4, 5) に出力するために，それぞれをディジタル出力に設定します．

以上の設定が済んだら，Project Resources エリアの Generate ボタンを左クリックします．すると，adc.h ファイルと adc.c ファイルが自動生成されます．

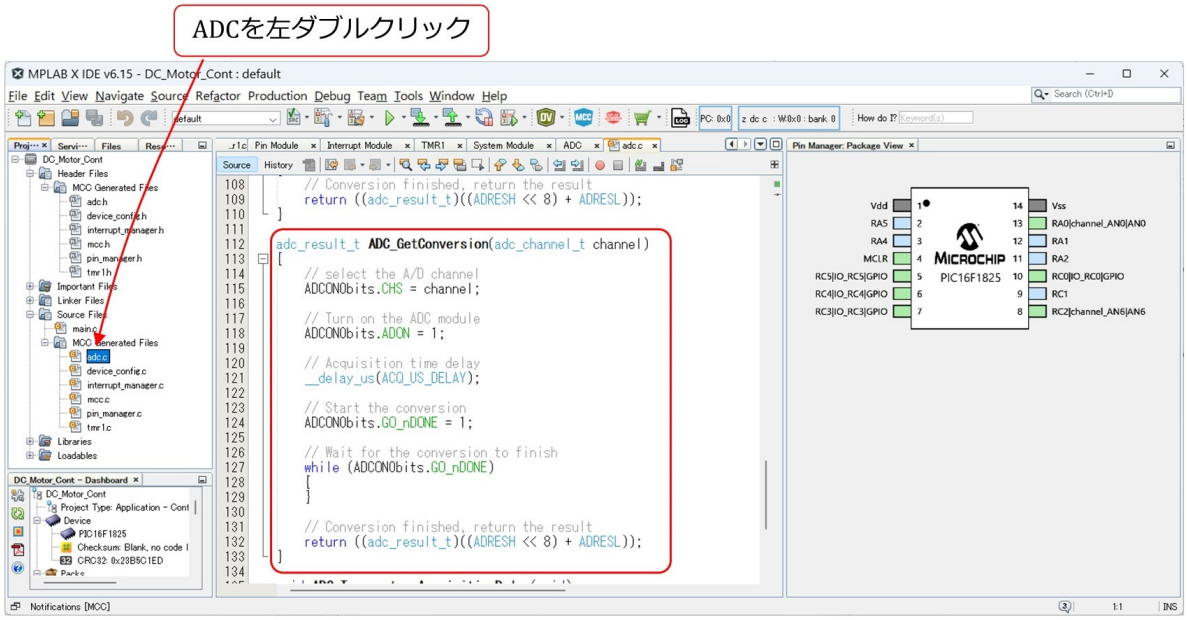


図 1.49: A-D 変換モジュールのプログラム

図 1.49 は，MCC により自動生成された adc.c ファイル内の ADC_GetConversion() 関数を示します．この関数を実行させれば，引数で指定したチャネル (ANx) からアナログ値を読み込んで，A-D 変換を実行し，変換結果を返り値とします．

図 1.50 は，A-D 変換モジュール設定結果のブロック図です．図 1.49 の ADC_GetConversion() 関数で使われているビット名，レジスタ名を記してあります．以下に各用語を説明します．

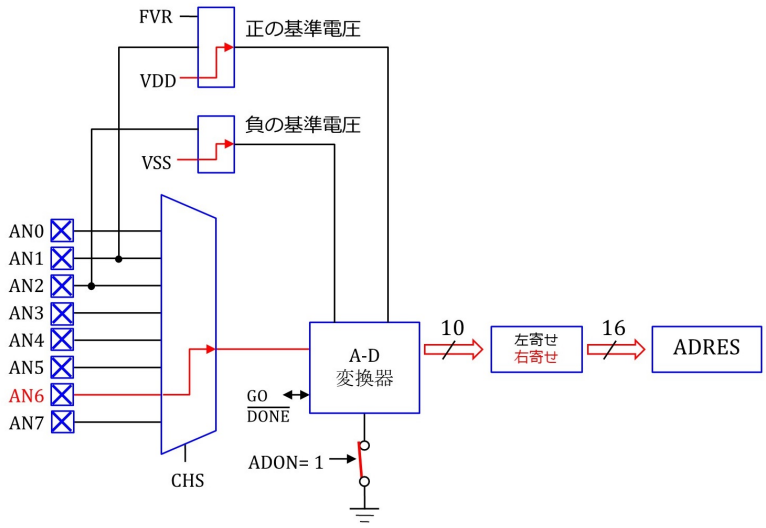


図 1.50: A-D 変換モジュールのブロック図

- CHS

  入力チャネルを選定します．この設定では AN6 を入力とします．

- ADON

  A-D 変換モジュールをオンにします．

- GO_nDONE

  GO_nDONE = 1 により A-D 変換を開始させます．A-D 変換が終了すると，GO_nDONE = 0 となります．while 文により，A-D 変換が終了するまで待ちます．

- ADRES

  10 ビットの変換結果が 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．

1.4.5　タイマ 1 割り込み処理関数の記述

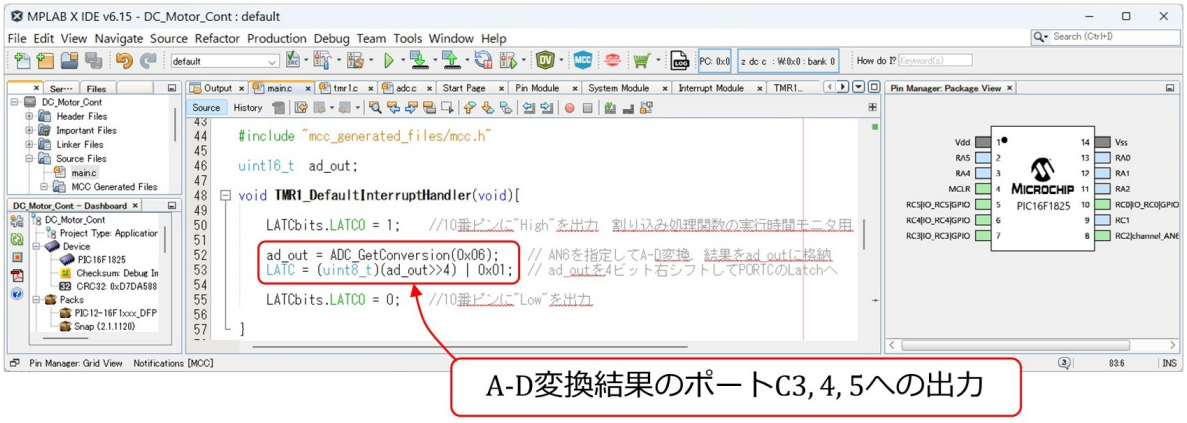


図 1.51: タイマ 1 割り込み処理関数による A-D 変換実行と結果のポート C への出力

図 1.51 は，タイマ 1 割り込み処理関数において，A-D 変換を実行して，変換結果をポート C3(7 番ピン), C4(6 番ピン), C5(5 番ピン) に出力するコードを示します．

```
ad_out = ADC_GetConversion(0x06); // AN6 を指定して A-D 変換.
                                  結果を ad_out に格納
LATC = (uint8_t)(ad_out>>4) | 0x01; // ad_out を 4 ビット右シフトして
                                  PORTC の Latch へ
```

10 ビットの A-D 変換結果は，ad_out の下位 10 ビットに格納されます．ビット番号は最下位が第 0 ビットです．A-D 変換結果の上位 3 ビットは第 9, 8, 7 ビットにあります．これら 3 ビットを，PORTC の第 5, 4, 3 ビット (C5, 4, 3) の入力とするために，ad_out の値を 4 ビット右シフトします．C0 ビットは割り込み処理関数実行のモニタ用で，LATC0 = 1 なので，この値を変更しないように，4 ビット右シフト結果と 0x01 との OR をとります．

1.4.6　A-D 変換実験結果

図 1.52 は，図 1.51 のプログラムによる A-D 変換実験波形例です．オシロスコープ画面のスナップショットです．縦軸は 5 [V/div]，横軸は 10 [ms/div] です．図 1.42 の実験回路において，8 番ピンの入力電圧 v_{IN} に，繰り返し周波数 10 Hz，電圧変化範囲 0 ～ 5[V] の三角波を与え，5, 6, 7 番ピンの電圧 $v_{OUT5}, v_{OUT4}, v_{OUT3}$ を計測した結果です．

A-D 変換結果の上位 3 ビットが，v_{IN} の変化に応じて，2 進数で変化する様子がわかります．

図 1.53 は，図 1.51 のタイマ 1 割り込み処理関数の処理時間の計測結果です．10 番ピンの出力電圧です．処理時間は約 24 [μs] でした．

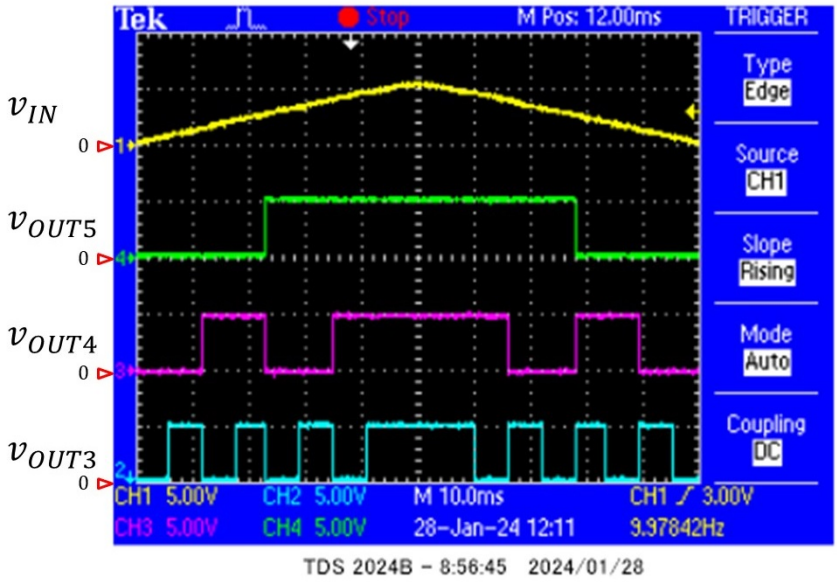


図 1.52: A-D 変換実験波形例

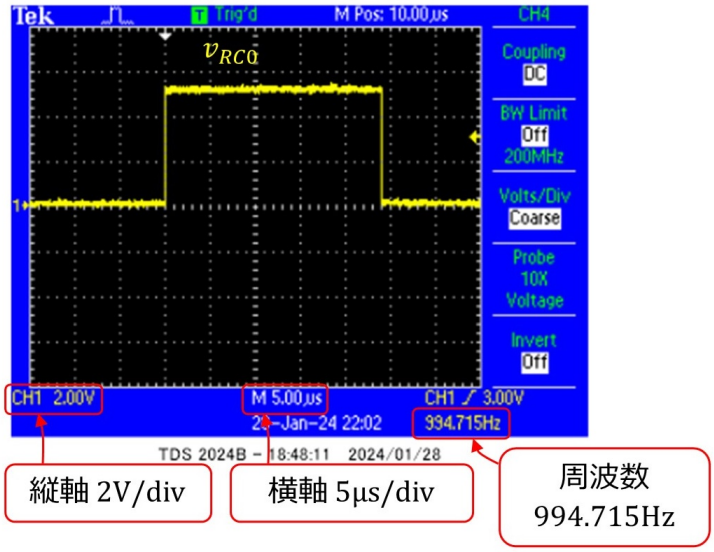


図 1.53: A-D 変換実験実行時のタイマ 1 割り込み処理関数の処理時間

1.5　PWM 制御

1.5.1　PWM 制御実験回路

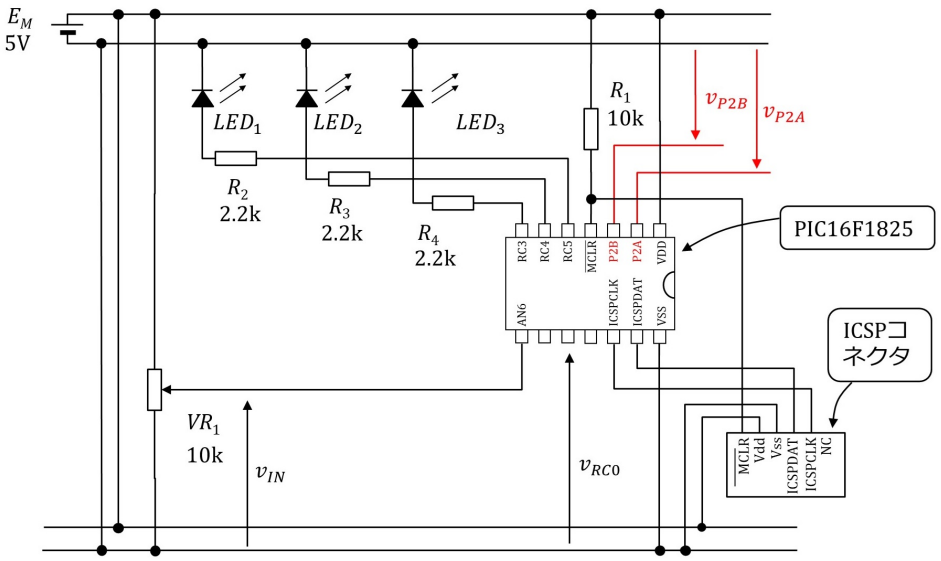


図 1.54: PWM 制御実験回路

図 1.54 は PWM 制御を実験するための回路です．図 1.42 の回路において，PWM 制御電圧 v_{P2A}, v_{P2B} を P2A (2 番ピン)，P2B (3 番ピン) に出力します．

1.5.2　PWM 制御実験用プログラムのブロック図

図 1.55 は PWM 制御プログラムのブロック図です．タイマ 1 割り込み処理関数が，A-D 変換値 ad_out をデューティ比として，PWM モジュール内の CCPR2L レジスタと DC2B ビットに渡します．PWM モジュールは，タイマ 2 を時間ベースとして，CCPR2L と DC2B の値をもとに PWM 波形を生成して，P2A と P2B から出力します．なお，ブロック図では，図 1.54 の 5〜7 番ピンへの A-D 変換結果（上位 3 ビット）出力は省略してあります．

PWM モジュール内では，CCPR2L(8 ビット) レジスタの下位に DC2B(2 ビット) を付け加えて 10 ビットレジスタを作っているため，構成が少し回りくどい（筆者の主観）です．図 1.48 のステップ 4 で，A-D 変換結果を 16 ビットレジスタ ADRES に右寄せで格納させる設定としました．タイマ 1 割り込み処理関数は，その変換結果を読み出して，ad_out に格納します．そして，ad_out の第 9〜2 ビットの 8 ビットと，第 1, 0 ビットの 2 ビットに分けて，それぞれ，CCPR2L レジスタ (8 ビット) と DC2B ビット (2 ビット) に格納します．PWM モジュールは，CCPR2L と DC2B をつなぎ合わせて 10 ビットの値にもどして，ディーティ比の決定を行います．

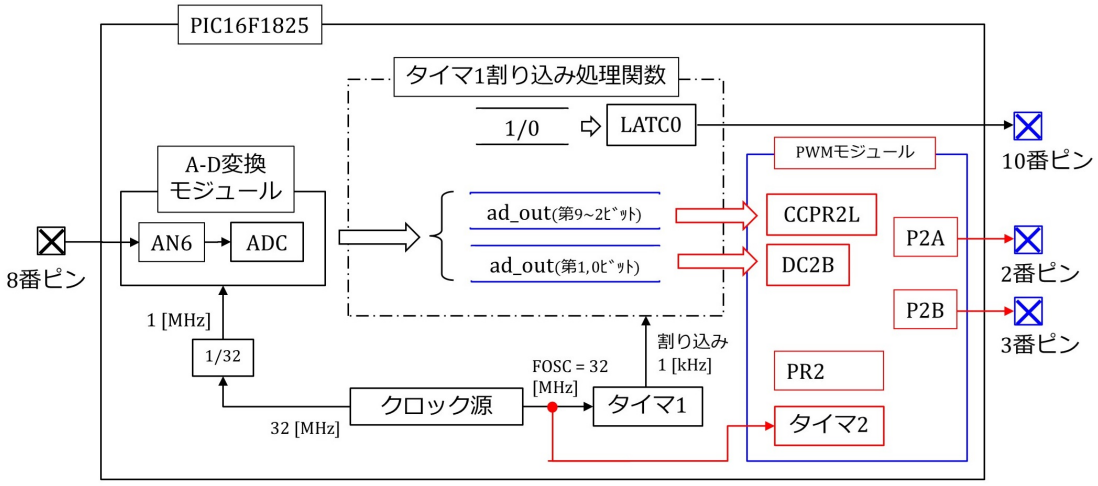


図 1.55: PWM 制御プログラムのブロック図

1.5.3　PWM モジュールのブロック図

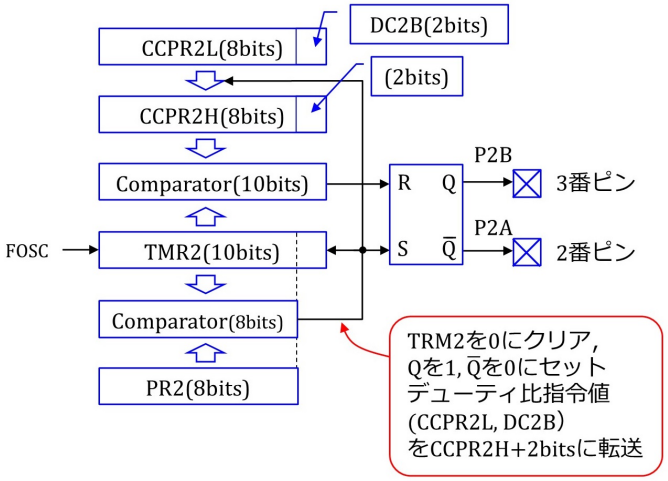


図 1.56: PWM モジュールのブロック図

図 1.56 は PWM モジュールのブロック図です．TMR2 は 8 ビットタイマですが，PWM モジュールのタイムベースとする場合は，2 ビットが追加されて，10 ビットタイマとして機能します．この 10 ビットタイマが FOSC をカウントアップします．比較器 (Comparator) が 2 個あって，上の比較器が CCPR2L と DC2B をつなぎ合わせた値 (CCPR2L × 4 + DC2B) を TMR2 と比較して，一致したら Q を 0，$\bar{Q}$ を 1 にリセットします．下の比較器は，PR2 レジスタ (8 ビット) の値を TMR2 の上位 8 ビットの値と比較して，一致したら Q を 1，$\bar{Q}$ を 0 にセットします．

以上の動作のイメージを図 1.57 に示す．TMR2 レジスタの値が 0 に初期化されたとき，PWM モジュール出力 P2B = Q = 1, P2A = $\overline{Q}$ = 0 とセットされます．TMR2 の値は

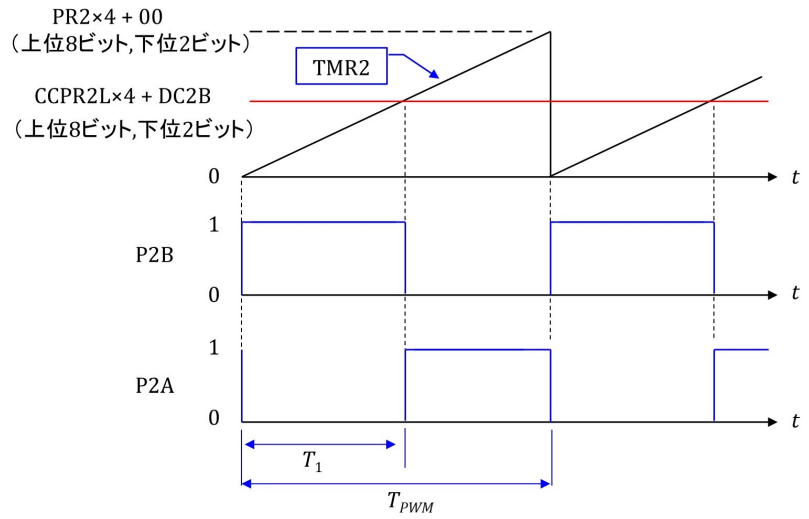


図 1.57: PWM モジュールの動作

FOSC によりカウントアップされます．

$$\mathrm{TMR2} = \mathrm{CCPR2L} \times 4 + \mathrm{DC2B} \tag{1.7}$$

となったとき，$\mathrm{P2B} = \mathrm{Q} = 0$, $\mathrm{P2A} = \overline{\mathrm{Q}} = 1$ にリセットされます．

$$\mathrm{TMR2} = \mathrm{PR2} \times 4 \tag{1.8}$$

となったとき，$\mathrm{TMR2} = 0$，$\mathrm{P2B} = 1$, $\mathrm{P2A} = 0$ とセットされます．そして，TMR2 のカウントアップが再開されます．

以上により，P2B, P2A の値の平均値は $\mathrm{CCPR2L} \times 4 + \mathrm{DC2B}$ の値に比例します．このように P2B, P2A の幅（パルス幅と呼ぶ）を制御する手法を PWM(Pulse Width Modulation：パルス幅変調) 制御法と呼びます．TMR2 の三角波の繰り返し周期 T_{PWM} を PWM 周期と呼びます．この逆数を PWM 周波数 $f_{PWM} = 1/T_{PWM}$ と呼びます．また，$\mathrm{P2B} = 1$ の期間を T_1 とすると $\delta = T_1/T_{PWM}$ をデューティ比と呼びます

FOSC = 32 [MHz] のとき，PR2 = 0xFF とすると，PWM 周期 T_{PWM} は

$$\begin{aligned} T_{PWM} &= \frac{1}{\frac{\mathrm{FOSC}}{256\times4}} \\ &= 32[\mu\mathrm{s}] \end{aligned} \tag{1.9}$$

であり，PWM 周波数 $f_{PWM} = 31.25[\mathrm{kHz}]$ です．また，デューティ比 δ の分解能は $1/2^{10} = 1/1024$ です．

1.5.4　PWM モジュールの設定

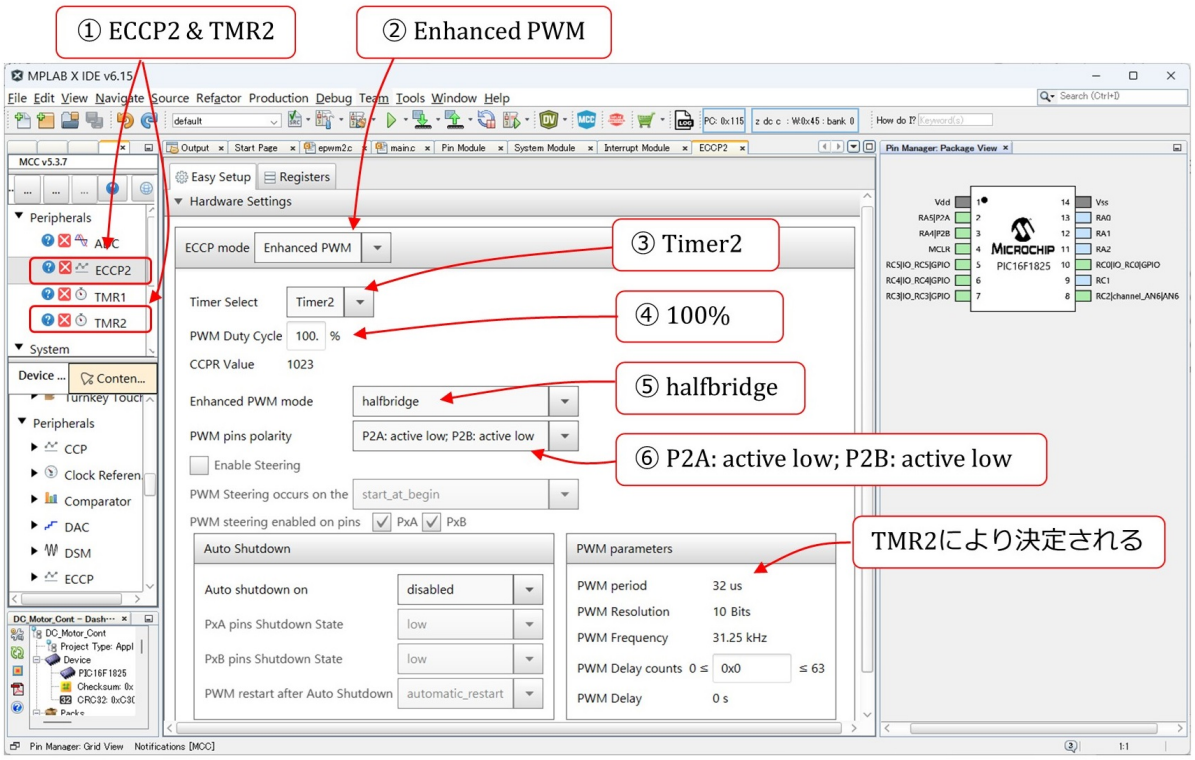


図 1.58: PWM モジュールの設定

前節の DC_Motor_Cont プロジェクトに，PWM モジュールと TMR2 モジュールを追加します．図 1.58 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．TMR2 設定は次項で述べます．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．

(4) PWM Duty Cycle を 100%に設定

これにより，デューティ比 (Duty Cycle) を最大で 100%にできます．TMR2 にて PWM 周期 $T_{PWM} = 32[\mu s]$ と設定すると，CCPR Value が 1023 と表示されます．この値は FOSC = 32 MHz より

$$\begin{aligned} \text{CCPR} &= \text{FOSC} \times T_{PWM} - 1 \\ &= 1023 \end{aligned} \tag{1.10}$$

と計算されます．Duty Cycle 指令値の上限値が 1023 となります．

(5) halfbridge を選択

P2A, P2B を使用します．

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

この設定により，図 1.57 のように，P2A, P2B は，

$$\begin{aligned} \text{TMR2} &< \text{CCPR2L} \times 4 + \text{DC2B} \quad \text{のとき} \quad \text{P2A} = 0 (\text{P2B} = 1) \\ \text{TMR2} &\geq \text{CCPR2L} \times 4 + \text{DC2B} \quad \text{のとき} \quad \text{P2A} = 1 (\text{P2B} = 0) \end{aligned} \tag{1.11}$$

となります．

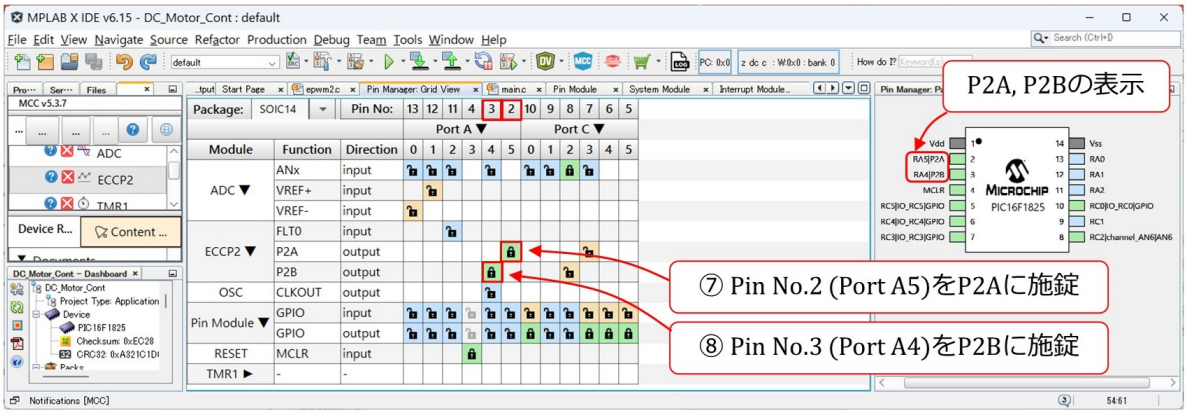


図 1.59: PWM モジュールのピン設定

(7) Pin No. 2 (Port A5) を P2A，Pin No. 3 (Port A4) を P2B に施錠

図 1.59 のように，2 番ピンの P2A 欄と 3 番ピンの P2B 欄を施錠します．右上のパッケージの 2 番ピンに P2A, 3 番ピンに P2B の文字が現れることを確認してください．

1.5.5　タイマ 2 の設定

図 1.60 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．

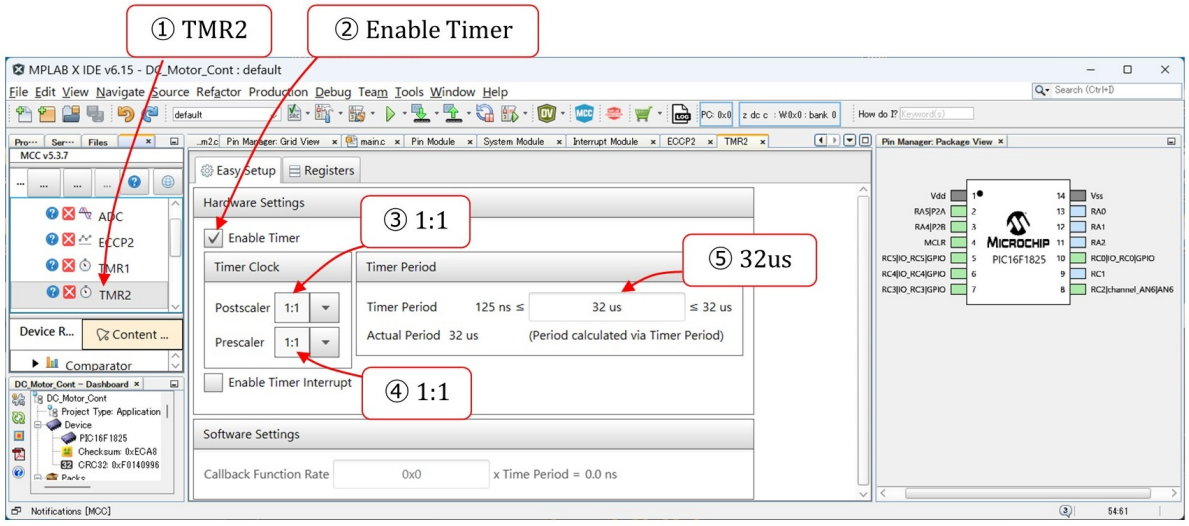


図 1.60: タイマ 2 モジュールの設定

(4) 1:1 を選択

クロックを 1/1 に分周します．データシートによると，TMR2 レジスタのクロックは FOSC/4 = 8 [MHz] です．図 1.56 の PWM モジュールのブロック図にて述べたとおり，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，そのクロックは 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 PWM 周期 T_{PWM} となります．

以上で周辺モジュール設定が終了です．Project Resources エリアの Generate ボタンを左クリックすることで，epwm2.h, tmr2.h, epwm2.c, tmr2.c ファイルが自動生成されます．

1.5.6　タイマ 1 割り込み処理関数の記述

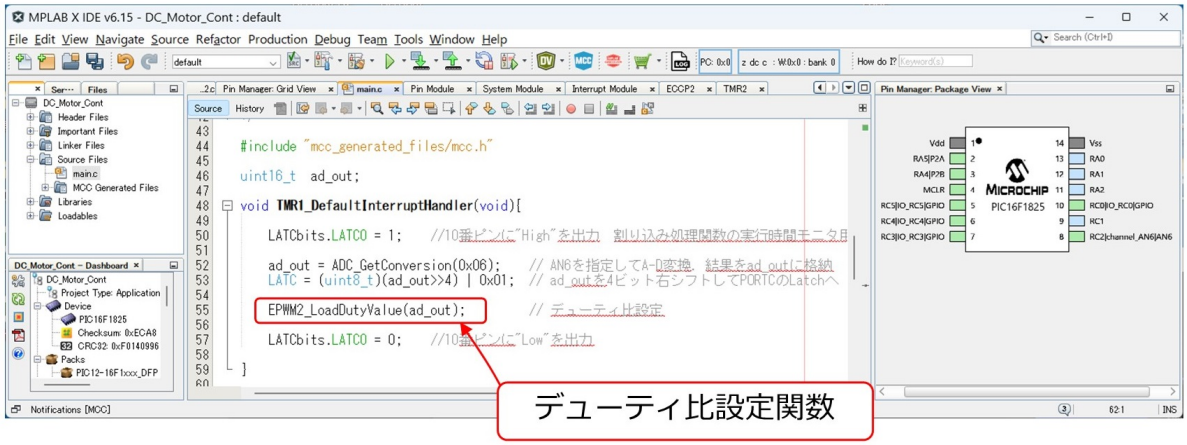


図 1.61: タイマ 1 割り込み処理関数にてデューティ比設定

図 1.61 に示すように，タイマ 1 割り込み処理関数内にデューティ比設定関数を記入しました．この EPWM2_LoadDutyValue() 関数は，epwm2.c ファイル内に自動生成されています．この関数が，図 1.55 の ad_out の第 9〜2 ビットを CCPR2L レジスタに格納し，第 1, 0 ビットを DC2B ビットに格納します．

1.5.7　PWM 制御の実験結果

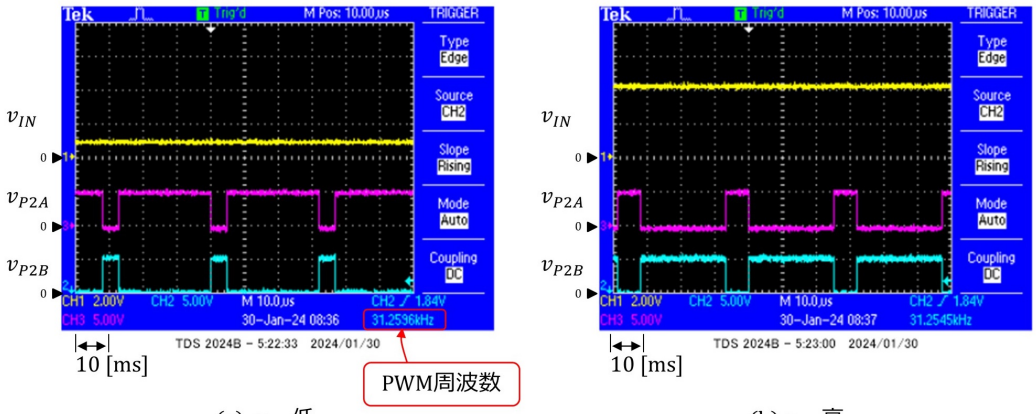


図 1.62: PWM モジュール実行時の入出力波形

図 1.62 は，8 番ピンの入力電圧 v_{IN} と 2, 3 番ピンの PWM モジュール出力電圧 $v_{\mathrm{P2A}}, v_{\mathrm{P2B}}$ の実験波形例です．同図 (a) は入力電圧 v_{IN} が低い場合，(b) は v_{IN} が高い場合です．入力電圧の高低に応じて，$v_{\mathrm{P2A}}, v_{\mathrm{P2B}}$ のパルス幅が変化している様子が分かります．また，PWM 周波数 $f_{PWM} \approx 31.26$ [kHz] でした

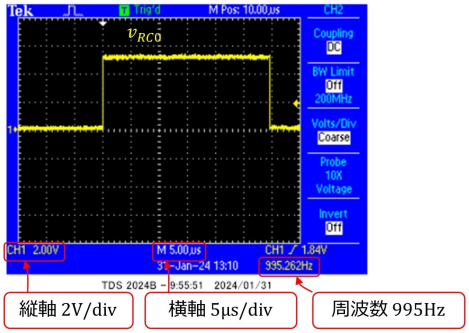


図 1.63: デューティ比設定関数を含むタイマ 1 割り込み処理関数の処理時間

図 1.63 はデューティ比設定関数を含むタイマ 1 割り込み処理関数の処理時間を示します．図の横軸は 5 [μs/div] なので，処理時間は約 29 [μs] です．

1.6　DC モータの回転数制御

1.6.1　回路図

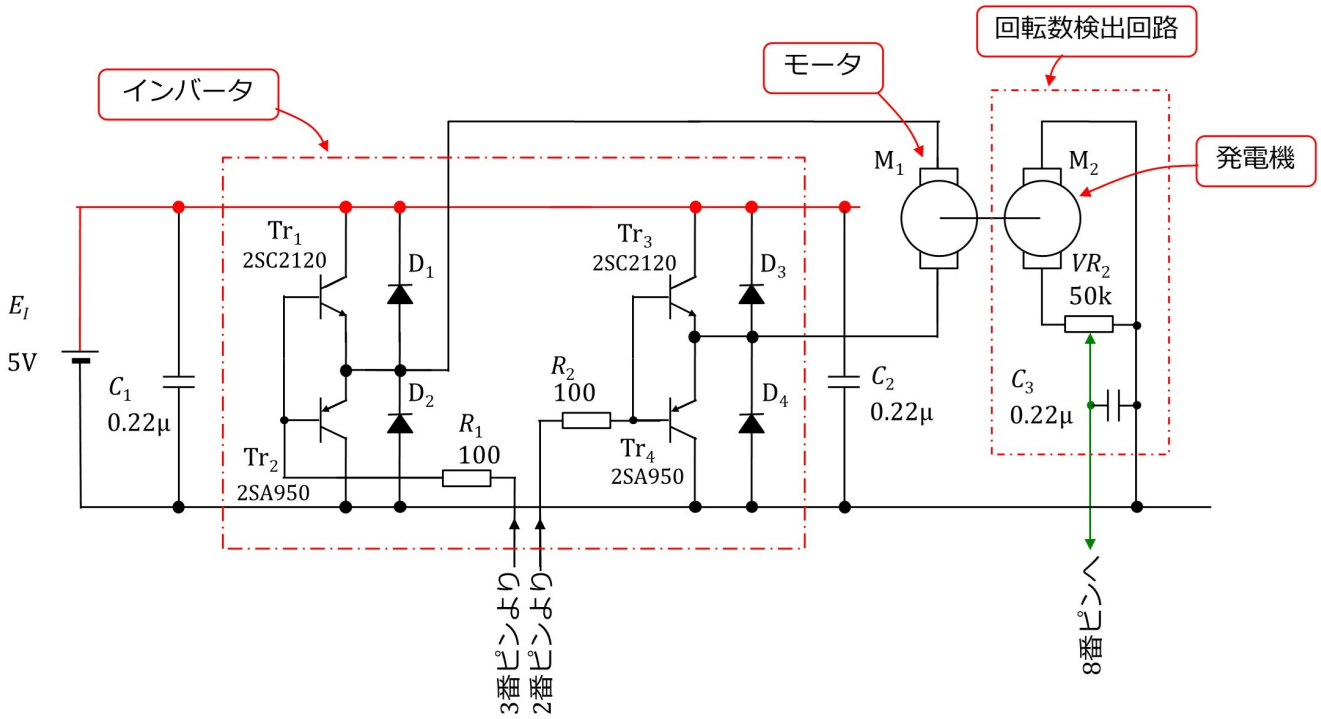


図 1.64: インバータ，モータと回転数検出回路

図 1.54 の回路に図 1.64 のインバータ，モータと回転数検出回路を接続することで図 1.4 の回路を構成できます．前節までで A/D 変換モジュールと PWM モジュールを使えるようになったので，いよいよ DC モータの回転数制御を実現する段階になりました．

1.6.2　部品

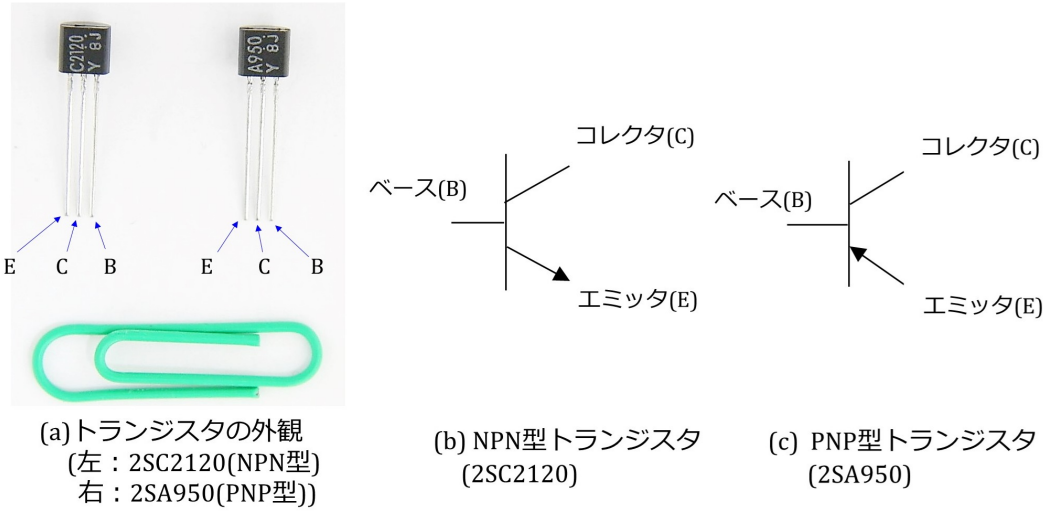
表 1.3: 部品表（追加分）

(2024年2月時点)

品名	型式	個数	単価	値段	入手先の例
トランジスタ	2SC2120 (20個入り)	1	150	150	秋月電子通商
	2SA950 （20個入り）	1	150	150	〃
整流用ショットキーダイオード	1S4 40V, 1A	4	20	80	〃
積層セラミックコンデンサ	0.22μF 50V	3	20	60	〃
電解コンデンサ	10μF 16V	1	10	10	〃
半固定ボリューム	50kΩ,	1	50	50	〃
電池ボックス	単3×4本 フタ付きプラスチック・スイッチ付き	1	160	160	〃
抵抗	100Ω,, 1/4W（100個入り）	1	100	100	〃
DCモータ	マブチ RE-280RA ﾓｰﾀｰﾍﾞｰｽ付き	2	420	840	アマゾン，ヨドバシ，共立エレショップ
			総計	1600	円

図 1.64 の回路を製作するには，表 1.3 の部品を必要とします．価格，入手先は 2024 年 2 月時点のものです．

トランジスタ



(a)トランジスタの外観
(左：2SC2120(NPN型)
右：2SA950(PNP型))

(b) NPN型トランジスタ
(2SC2120)

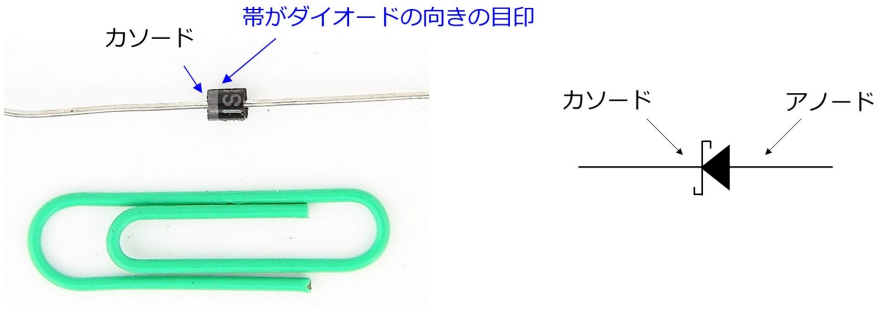
(c) PNP型トランジスタ
(2SA950)

図 1.65: トランジスタの外観と記号

図 1.65 はトランジスタ Tr_1～Tr_4 の外観と記号です．同図 (a) の左側が 2SC2120 であり，NPN 型トランジスタです．右側は 2SA950 であり，PNP 型トランジスタです．これらはバ

イポーラトランジスタと呼ばれます．3 つの電極には呼び名があり，写真のようにラベル面を上にしたときに，左からエミッタ (E; Emitter)，コレクタ (C: Collector)，ベース (B: Base) です．E, C, B を「えくぼ」と憶えておくと忘れないでしょう．同図 (b), (c) はそれぞれ NPN 型, PNP 型トランジスタの記号です．両者はエミッタ電極の矢印の向きで区別されています．これはエミッタ電流の流れる向きを示しています．

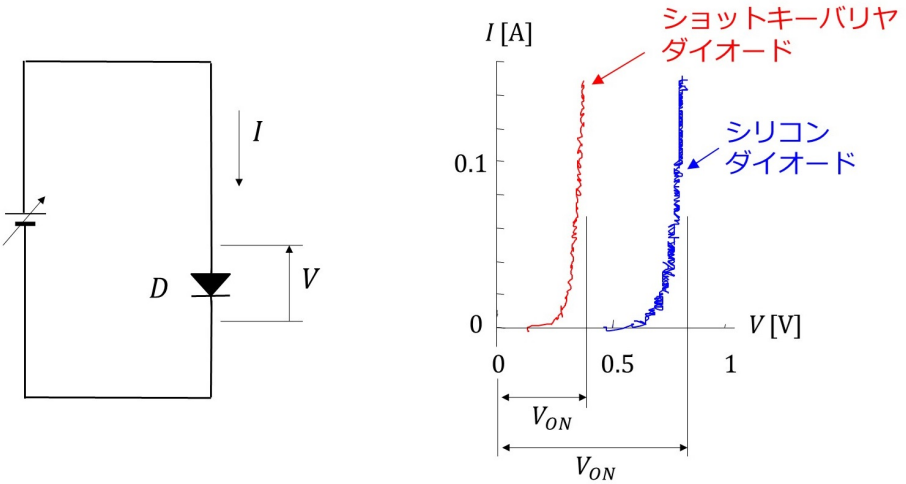
ダイオード



(a) 外観　　(b) ダイオードの記号

図 1.66: ショットキーバリヤダイオードの外観と記号

図 1.66 は整流用ショットキーバリヤダイオードの外観と記号です．ダイオードにはアノード電極とカソード電極があります．ダイオードの円筒部分に帯がある側がカソード電極です．記号には，ショットキーバリヤダイオードであることを示すために，カソード側に S の文字が使われています．



(a) ダイオードの特性測定回路　　(b) 電圧・電流特性

図 1.67: ショットキーバリヤダイオードの特性

ショットキーバリヤダイオードの特徴はオン電圧が低いことです．図 1.67(a) はダイオード D の両端電圧 V と電流 I の向きの定義を示し，同図 (b) はその測定結果の例を示します．シリコンダイオードは 100[V], 1[A] の定格のものです．ショットキーバリヤダイオードの定格は 40[V], 1[A] です．オシロスコープの測定結果であるためノイズが乗っていますが，それぞれのオン電圧 V_{ON} に差があることが見て取れます．ショットキーバリヤダイオードはオン電圧が低いことから，シリコンダイオードより損失が少なくなります．

積層セラミックコンデンサ

(a) 積層セラミックコンデンサの外観　　(b) コンデンサの記号

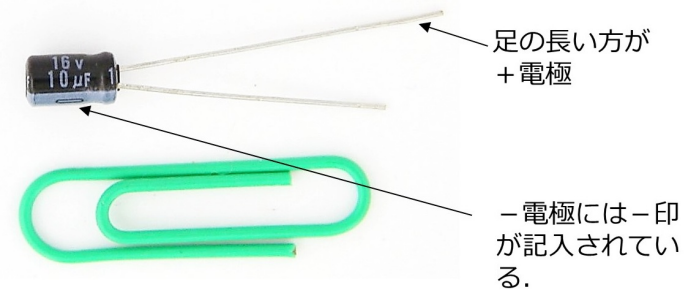
図 1.68: 積層セラミックコンデンサの外観と記号

図 1.68 は積層セラミックコンデンサの外観と記号です．セラミックコンデンサはノイズ対策用のコンデンサとしてよく用いられます．積層タイプは，セラミックコンデンサの静電容量を大きくしたもので，10[μF] の積層セラミックコンデンサもあります．図は 0.22[μF] のものです．表面に印字された 224 は

$$\begin{aligned} 224 &= 22 \times 10^4 [\mathrm{pF}] \\ &= 0.22 [\mu \mathrm{F}] \end{aligned} \tag{1.12}$$

を意味します．図 1.64 の回路では，インバータのオン・オフ（スイッチングと呼ばれる）によるノイズ（スイッチングノイズと呼ばれる）の除去用 (C_1, C_2)，および，DC モータを発電機として用いた場合の発電電圧に含まれるノイズ除去用 (C_3) として用いられています．

電解コンデンサ

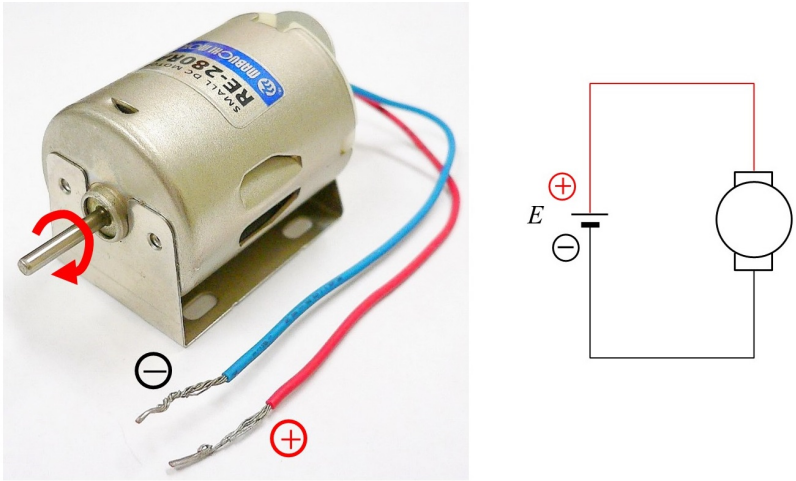


(a) 電解コンデンサ（10μF)の外観　　(b) 電解コンデンサの記号

図 1.69: 電解コンデンサの外観と記号

図 1.69 は電解コンデンサの外観と記号です．大容量コンデンサの代表であり，電源電圧の安定化などに良く用いられます．図 1.4 の回路においてマイコン用電源の＋の線（電源ラインと呼ばれる）と－の線（これも電源ラインと呼ばれる）の間に入れてあります．電解コンデンサとセラミックコンデンサを並列にして電源ライン間に入れることで，負荷変動などに対して電源を強化します．

DC モータ



(a) 外観と回転方向　　(b) 回路図

図 1.70: DC モータの外観

図 1.70 は本章で用いる DC モータ（マブチモータ, RE-280RA）の外観と回路図です．外観には，印加電圧の極性と回転方向の関係を示してあります．図示の⊕，⊖の向きに電

圧を印加すると，モータは矢印のように時計方向に回転します．最大効率運転時で 3 [V], 0.87 [A]，出力 1.61 [W] のモータです．

1.6.3　インバータ回路

●基本回路

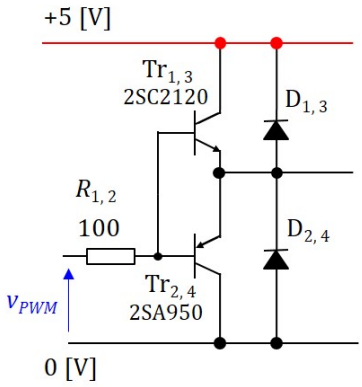


図 1.71: インバータの基本回路

PNP 型と NPN 型のトランジスタをペアとし，それぞれにダイオードを接続することで，図 1.71 のインバータの基本回路を構成できます．図中の電圧 v_{PWM} はマイコンの PWM モジュールの出力電圧です．図 1.62 の v_{P2A} もしくは v_{P2B} です．マイコンの電源電圧 V_{E_I} = 5 [V] とすると，v_{PWM} = 0 or 5 [V] の 2 値をとります．

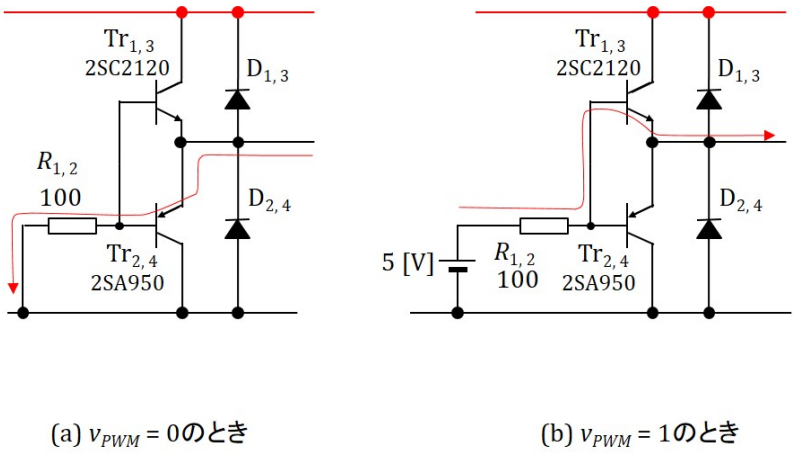


図 1.72: トランジスタの駆動

トランジスタの駆動の様子を図 1.72 に示します．同図 (a) は v_{PWM} = 0 のとき，(b) は v_{PWM} = 5 [V] のときです．v_{PWM} = 0 のときは図の経路で下側のトランジスタにベース

電流が流れ，このトランジスタがオンとなります．$v_{PWM} = 5$ [V] のときには上側のトランジスタがオンとなります．

●インバータの実験（抵抗負荷）

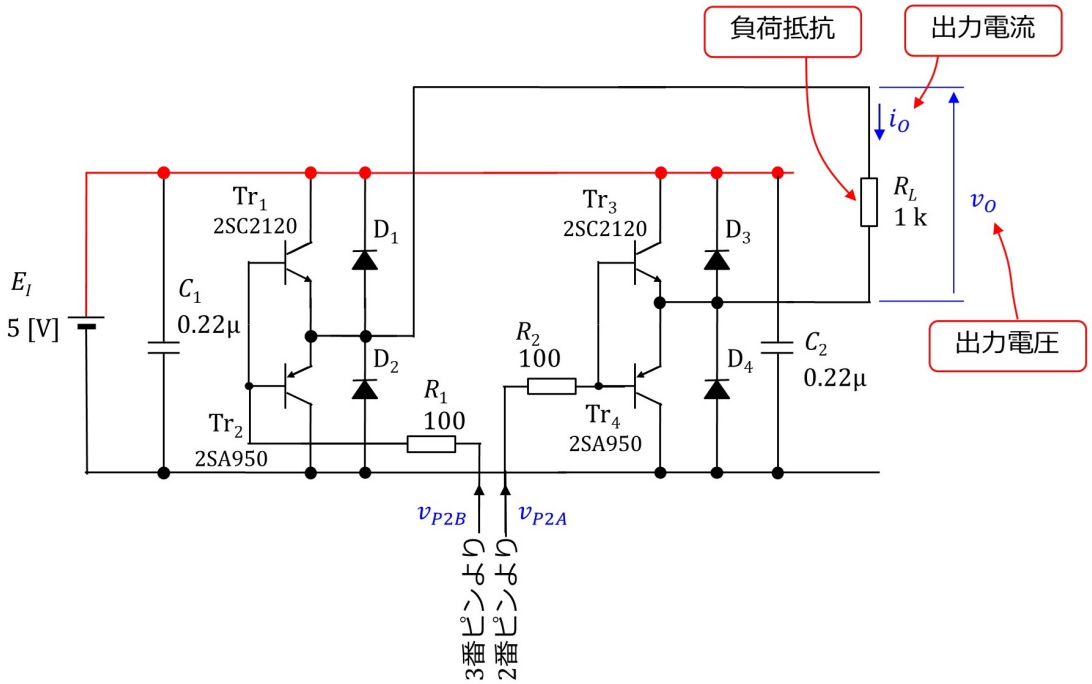


図 1.73: インバータの実験回路

図 1.73 はインバータの実験回路です．インバータの出力に負荷抵抗 R_L を接続してあります．出力電圧 v_O および出力電流 i_O の向きを図のように定義します．図 1.82 のプログラムにより，マイコンの 3 番ピンに v_{P2B} を出力させて，これにより $\mathrm{Tr}_1, \mathrm{Tr}_2$ を駆動させます．2 番ピンには v_{P2A} を出力させて，$\mathrm{Tr}_3, \mathrm{Tr}_4$ を駆動させます．

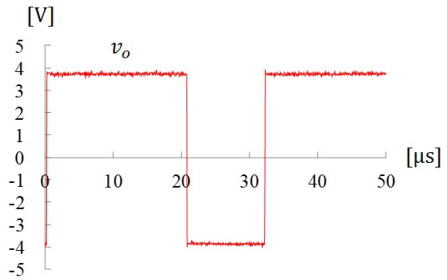


図 1.74: インバータの実験 _ 出力電圧波形（抵抗負荷）

得られた電圧波形を図 1.74 に示します．PWM 周期は式 (1.9) より 32 [μs] であり，図からも読み取ることができます．また，電圧振幅は約 3.8 [V] です．この値は，$v_{\mathrm{P2A}} = 5$ [V],

$v_{\mathrm{P2B}} = 0$ [V] のとき Tr_3 と Tr_2 がオンとなりますが，Tr_3 のベース・エミッタ間電圧を v_{BE}，Tr_2 のエミッタ・ベース間電圧を v_{EB} とすると

$$\begin{aligned} v_O &= v_{\mathrm{P2A}} - v_{BE} - v_{EB} - v_{\mathrm{P2B}} \\ &= 5 - v_{BE} - v_{EB} \\ &\approx 3.8[V] \end{aligned} \tag{1.13}$$

により与えられます．v_{BE} と v_{EB} の和は約 1.2[V] であったことが分かります．

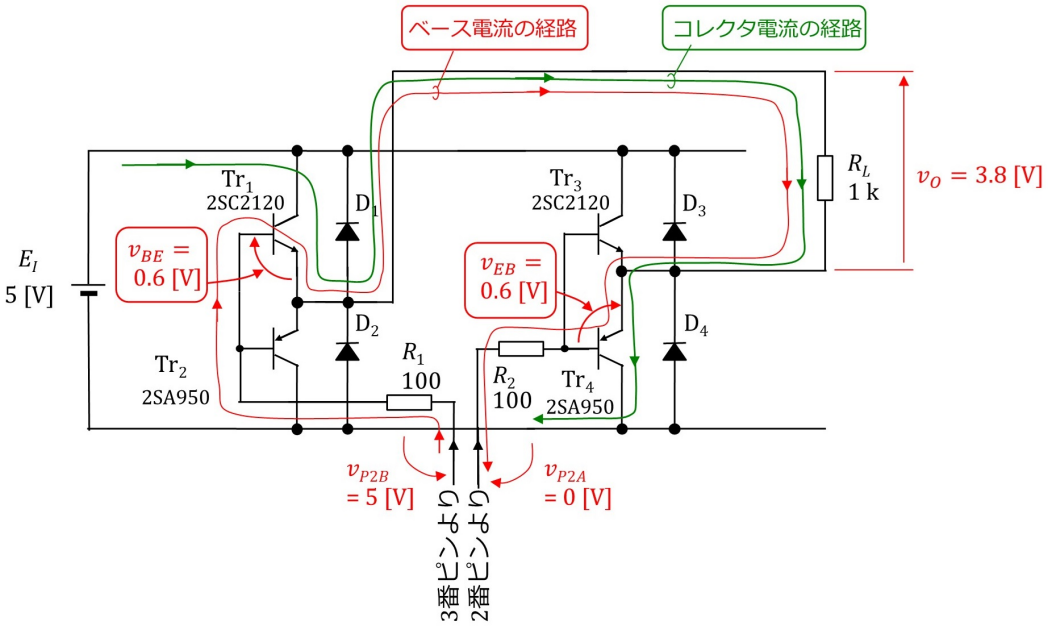


図 1.75: インバータの実験（v_{BE} の出力電圧への影響）

ベース電流の経路を図 1.75 に赤線で示します．3 番ピンから 2 番ピンまでの経路には Tr_1 の v_{BE}，出力電圧 v_O，Tr_4 の v_{EB} $(= -v_{BE})$ があります．トランジスタのベース・エミッタ間の特性は図 1.67 のシリコンダイオードの特性と同じです．同図は NPN 型トランジスタのベース・エミッタ間電圧対ベース電流特性に相当します．PNP 型トランジスタの場合は電圧，電流の極性が反対となる点を除けば，ほぼ同じ特性です．Tr_1 のベース・エミッタ間に 0.6 [V] 程度の電圧がかかることで，ベース電流が数十 [μA] 流れます．それによってコレクタ電流 I_C が

$$\begin{aligned} I_C &= \frac{3.8[V]}{1[\mathrm{k}\Omega]} \\ &= 3.8[\mathrm{mA}] \end{aligned} \tag{1.14}$$

流れます．コレクタ電流の経路を図 1.75 に緑色で示します．

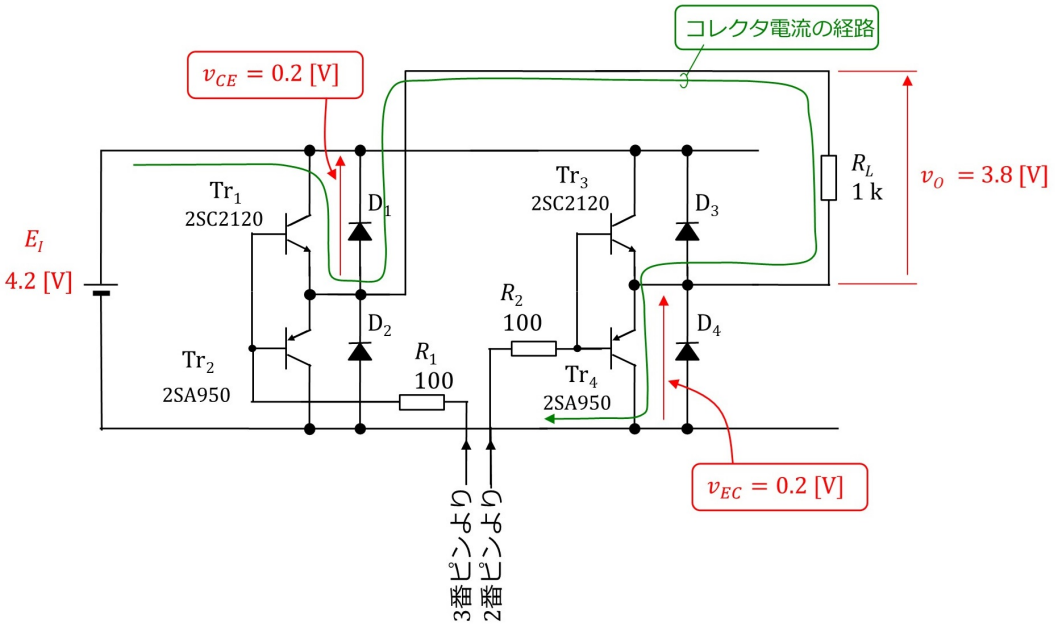


図 1.76: インバータの実験（v_{CE} の出力電圧への影響）

このインバータの駆動方式の場合，インバータの電源電圧 V_{E_I} を約 4.2 [V] 程度にまで下げても，インバータの出力電圧 v_O はほとんど変わらず約 3.8 [V] を維持します．その様子を図 1.76 に示します．この場合も式 (1.13) の関係は変わりません．変わるのはコレクタ電流の流れる経路における Tr_1 のコレクタ・エミッタ間電圧 v_{CE} と Tr_4 のエミッタ・コレクタ間電圧 v_{EC} の値です．コレクタ電流の経路上で

$$V_{E_I} = v_{EC} + v_O + v_{CE} \tag{1.15}$$

の関係が成立し，2SC2120 と 2SA950 では，

$$v_{EC} \approx v_{CE} \approx 0.2[V] \tag{1.16}$$

です．V_{E_I} をさらに低下させると，v_{EC}, v_{CE} はほぼこの値を保ったまま，出力電圧 v_O が低下します．逆に V_{E_I} を図 1.75 の 5 [V] よりも増大させた場合は，出力電圧の 3.8 [V] はほとんど変わらず，v_{EC}, v_{CE} が増大します．

●インバータの実験（DC モータ負荷）

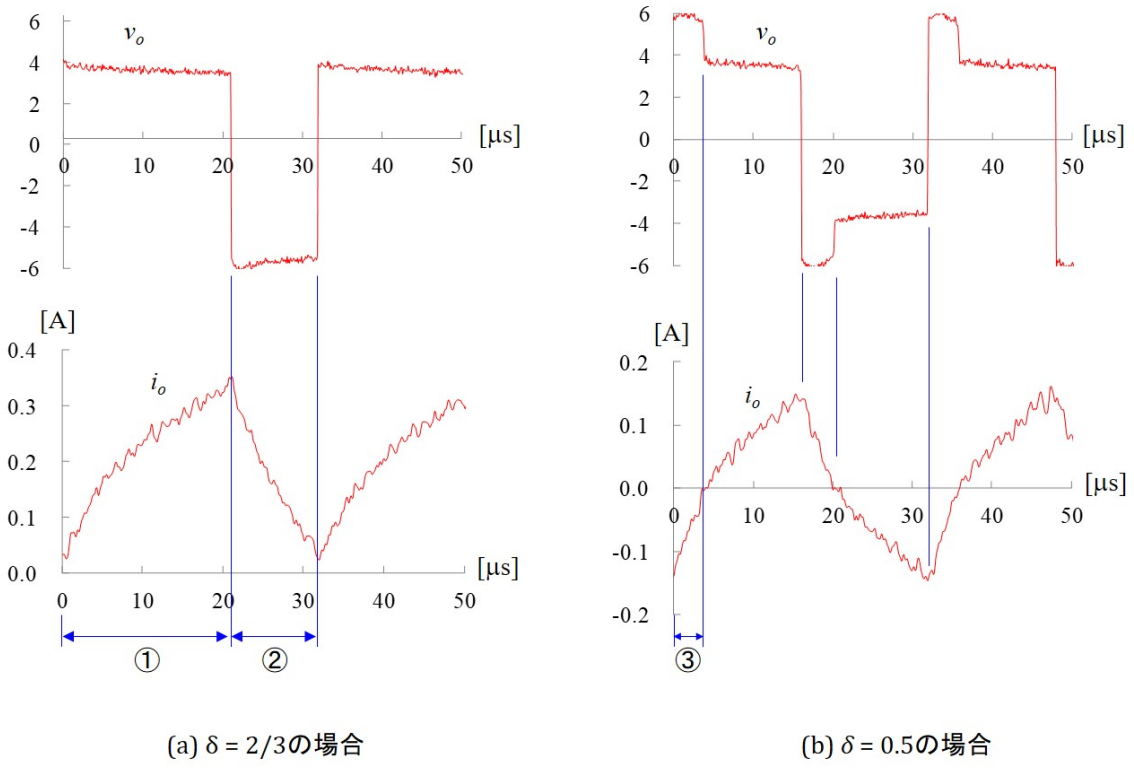


図 1.77: インバータの実験 _ 出力電圧・電流波形（DC モータ負荷）

図 1.77 は，図 1.73 の負荷抵抗 R_L の替わりに，DC モータを接続した場合の出力電圧 v_O，出力電流 i_O の測定例です．モータが回転しないように軸を指先でつまんで固定して，測定しました．また，波形はオシロスコープにより測定したため，ノイズが目立っています．同図 (a) はデューティ比 $\delta \approx 2/3$ のときであり，(b) は $\delta \approx 0.5$ のときです．(a) の①の区間は図 1.75 と同じ経路でコレクタ電流が流れ，出力電圧 $v_O \approx 3.8$ [V] となっています．②の区間では出力電流 i_O は正のままですが，出力電圧 v_O が負になっています．

図 1.78 は，図 1.77 の②の区間における出力電流 i_O の経路です．DC モータは電機子抵抗 R_a，電機子インダクタンス L_a の等価回路で表してあります．i_O は電源 E_I からダイオード D_2，DC モータ，ダイオード D_4 を通って E_I に戻っています．DC モータの両端電圧 v_O は，電源電圧 V_{E_I}= 5[V] に二つのショットキーバリヤダイオードのオン電圧 $V_D \approx 0.45$ [V] を加えた値になります．極性は負で，$v_O \approx -5.9$[V] です．②の区間の i_O は L_a が流します．この間，L_a の磁気エネルギは電源 E_I に電気エネルギとして戻されます．これはエネルギ回生と呼ばれます．エネルギ回生を行っている回路モードは回生モードと呼ばれます．

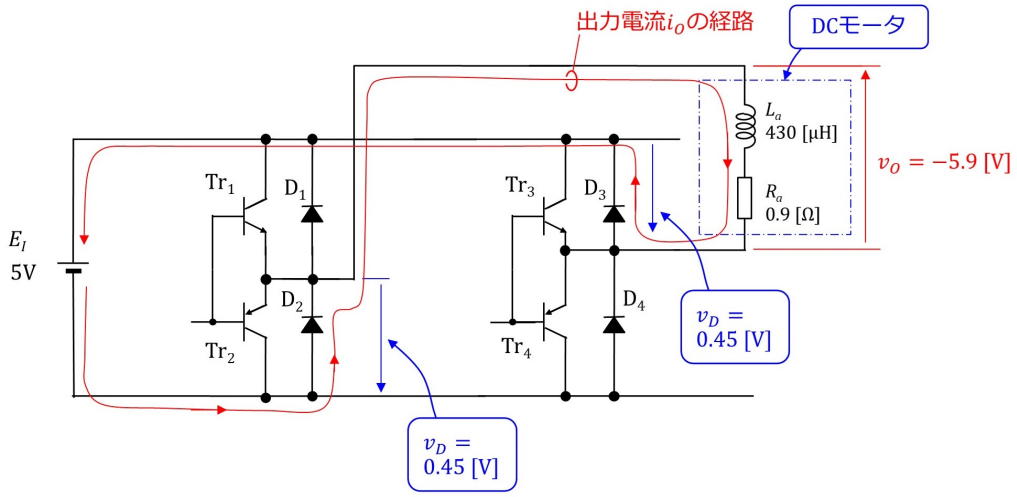


図 1.78: インバータの実験 _ 回生モード（区間②，DC モータ負荷）

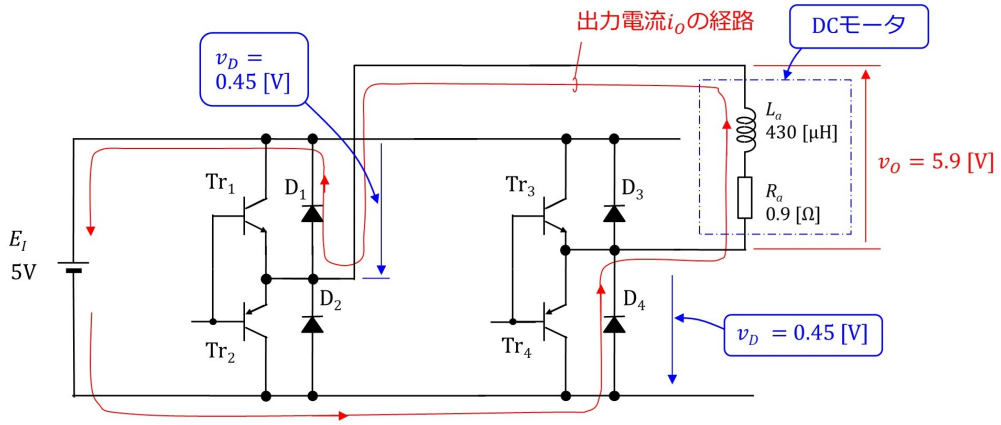


図 1.79: インバータの実験 _ 回生モード（区間③，DC モータ負荷）

図 1.79 は図 1.77 の区間③における出力電流 i_O の経路を示します．この区間では，i_O が負であり，v_O が正です．この区間も，L_a が i_O を流す回生モードです．

インバータ回路とその動作の詳細は拙著 [5] を参照してください．

1.6.4　フィルタ回路

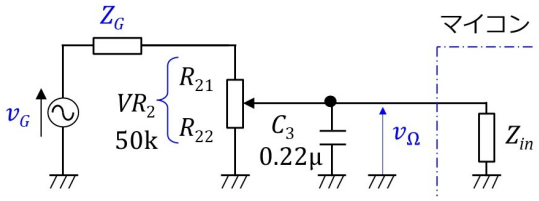


図 1.80: フィルタ回路

図 1.64 のモータ M_2，可変抵抗 VR_2 とコンデンサ C_3 からなる回転数検出回路において，M_2 は DC モータを発電機として用いています．後述の式 (2.10) より，発電電圧（モータの逆起電力）E_a はモータの回転数に比例するので，E_a により回転数を検出できます．ただし，これも後述するように，モータの逆起電力には図 2.28 のように大きな脈動成分（リップルと呼ばれます.）が含まれています．VR_2 と C_3 からなるフィルタ回路はこのリップルをとるための回路です．

図 1.80 はフィルタ回路の等価回路です．v_G は発電機の発電電圧であり，v_Ω はフィルタ回路の出力電圧です．VR_2 の可動電極の上側の抵抗値を R_{21}，下側のそれを R_{22} とします．発電機の内部インピーダンス Z_G は，電機子抵抗が約 1[Ω]，電機子インダクタンスが約 400[μH] なので，$VR_2 = 50$ [kΩ] と比べて小さく，無視できます．マイコン側の入力インピーダンス Z_{in} は，A/D コンバータの入力です．データシートの ANALOG INPUT MODEL より，静電容量は 15[pF]，漏れ電流は 100[nA] 程度なので，フィルタ回路のインピーダンスに比べて大きく，無視できます．

v_Ω と v_G の関係は

$$\begin{aligned} v_\Omega &= \frac{\frac{R_{22}}{1+j\omega C_3 R_{22}}}{R_{21}+\frac{R_{22}}{1+j\omega C_3 R_{22}}} v_G \\ &= \frac{R_{22}}{R_{21}+R_{22}+j\omega C_3 R_{21} R_{22}} \\ &= \frac{\frac{R_{22}}{R_{21}+R_{22}}}{1+j\frac{\omega C_3 R_{21} R_{22}}{R_{21}+R_{22}}} \end{aligned} \tag{1.17}$$

です．よって，この回路のカットオフ周波数 f_c は

$$f_c = \frac{R_{21}+R_{22}}{2\pi C_3 R_{21} R_{22}} \tag{1.18}$$

と求めらます．$R_{21} = 10[\mathrm{k\Omega}], \mathrm{R}_{22} = 40[\mathrm{k\Omega}]$ とすると，$C_3 = 0.22[\mu\mathrm{F}]$ なので，$f_c \approx 90[\mathrm{Hz}]$ です．

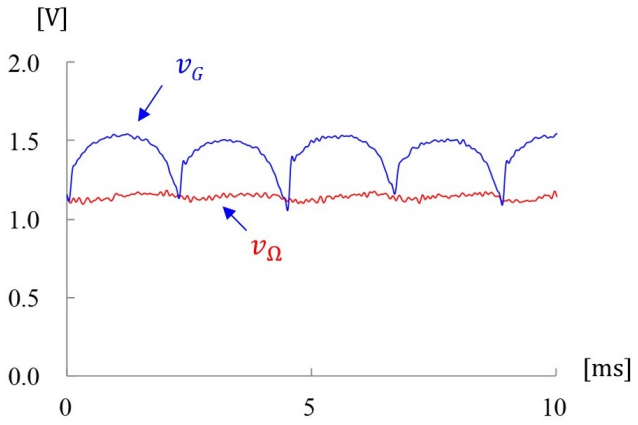


図 1.81: フィルタ回路の入出力電圧

図 1.81 はフィルタ回路の入力電圧 v_G と出力電圧 v_Ω の測定例です．この例ではリップルの繰り返し周波数は約 430 [Hz] です．フィルタ回路はこの成分を大きく低減しています．

1.6.5　回転数制御プログラムのブロック図

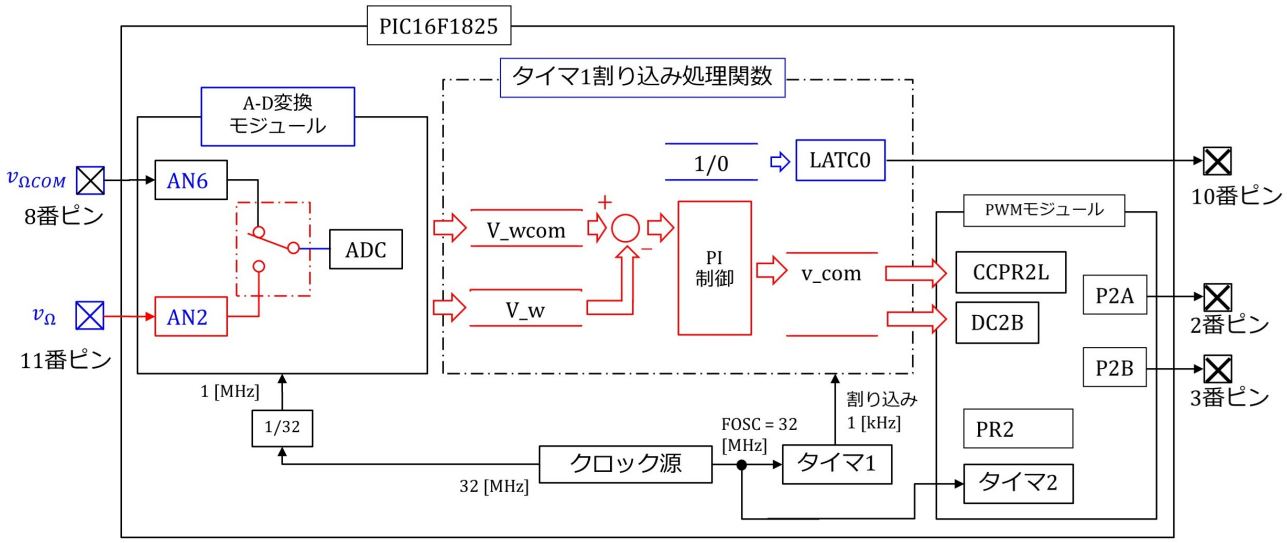


図 1.82: DC モータの回転数制御プログラムのブロック図

以上で 1.1 節の DC モータ回転数制御回路のプログラミングと実験の準備が整いました．図 1.82 は作成したプログラムのブロック図です．図 1.55 の PWM 制御プログラムのブロック図に対して新たに追加した箇所を朱書きで示します．図 1.4 に示すように回転数指令値 $v_{\Omega COM}$ を 8 番ピンに入力し，回転数検出値 v_Ω を 11 番ピンに入力します．PIC16F1825 内には A-D 変換モジュールが 1 つしかないので，入力を切り替えてそれぞれの電圧を検出

します．変換結果は $v_{\Omega COM}$ を v_wcom に，v_{Ω} を v_w にそれぞれ格納します．両者の差を用いて PI 制御演算を行い，出力を電圧指令値 v_com に格納します．

1.6.6　A-D 変換モジュール入力の追加設定

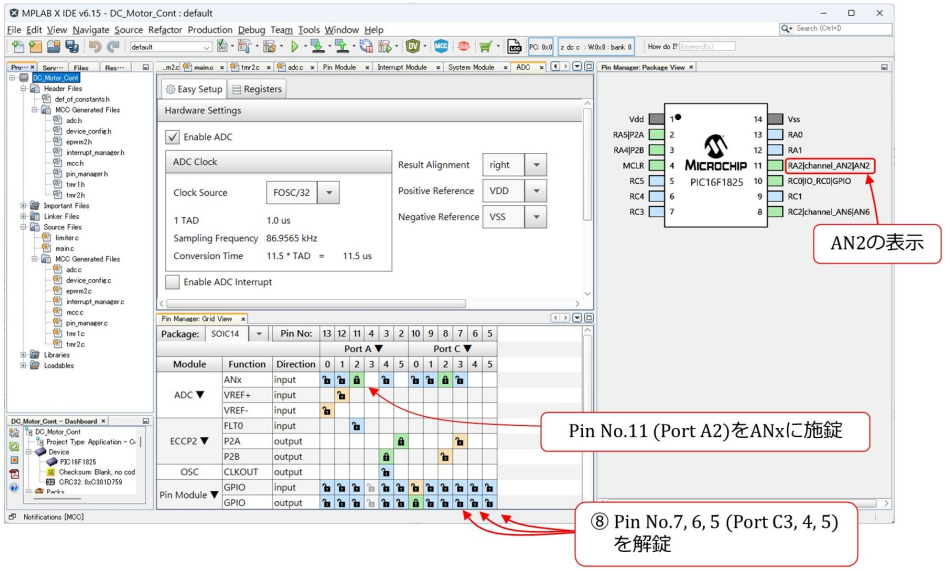


図 1.83: A-D 変換モジュール入力の追加設定

図 1.83 は，A-D 変換モジュール入力を追加設定した画面です．11 番ピン（ポート A2）を ANx に施錠しました．

1.6.7　PI 制御プログラム

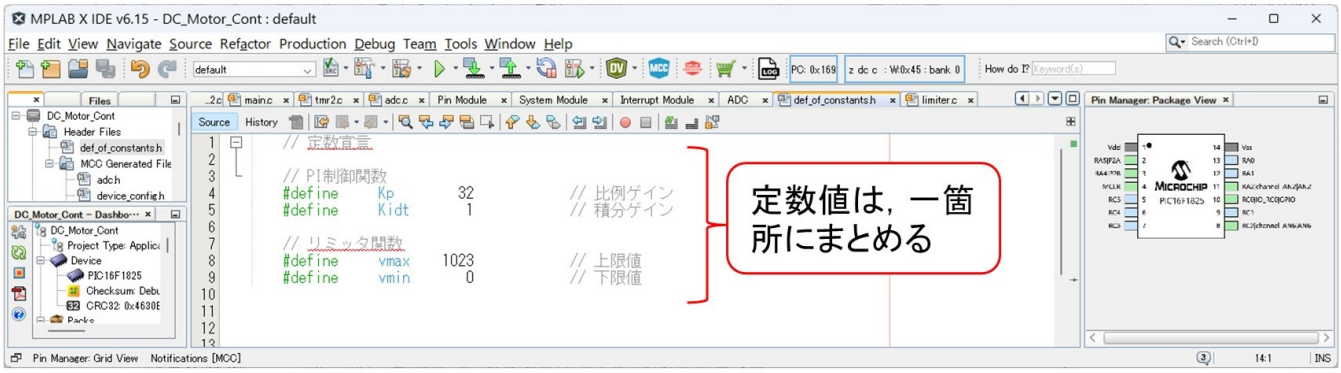


図 1.84: 定数値の定義

図 1.84 は def_of_constants.h ファイルに記述した定数宣言です．PI 制御関数内の比例ゲイン Kp，積分ゲイン Kidt と，リミッタ関数内の上限値 vmax, 下限値 vmin です．上限値の vmax = 1023 は (1.10) 式の CCPR 値です．定数値は，プログラム内に散在させないで，一箇所にまとめておくことで，後の変更／調整の際にプログラム内を探さなくて済みます．

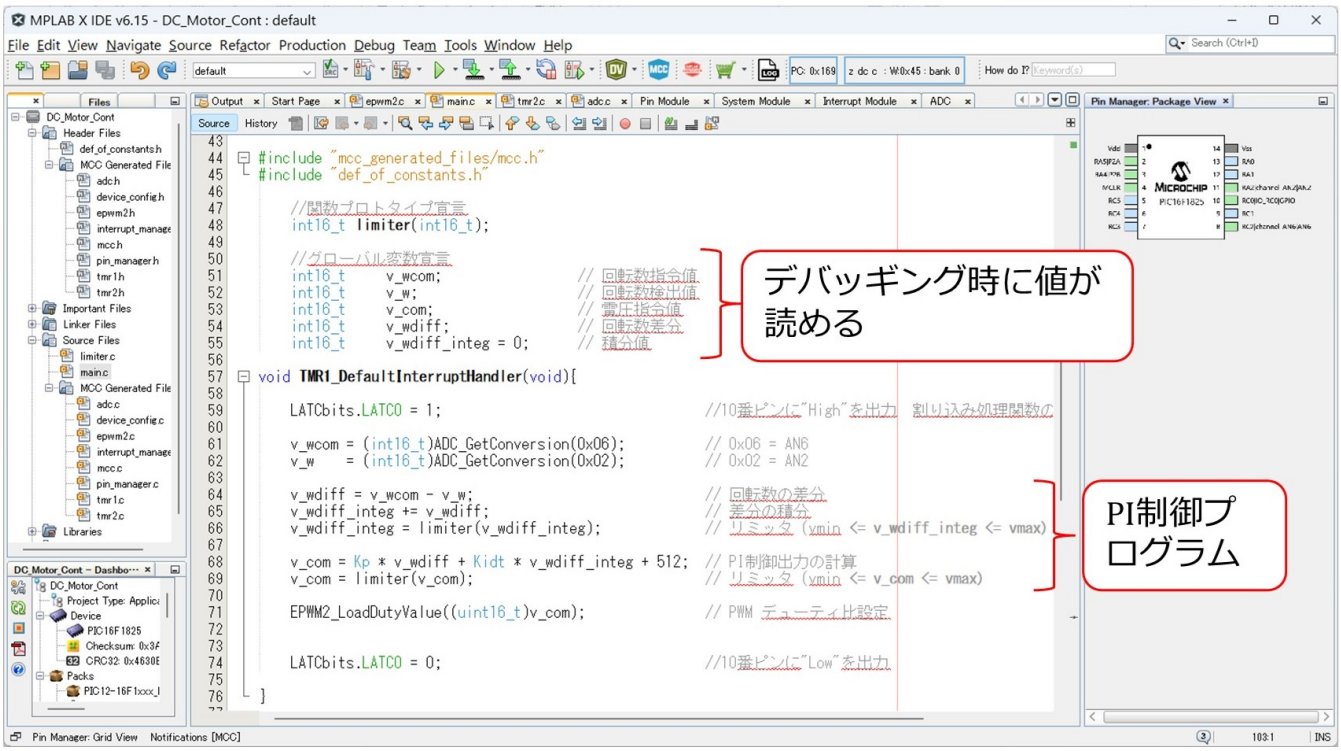


図 1.85: PI 制御プログラム

図 1.85 は，PI 制御プログラムです．関数プロトタイプ宣言とグローバル変数宣言および，タイマ 1 割り込み処理関数からなります．limiter() 関数は，電圧指令値 v_com と積分値 v_wdiff_integ を vmin～vmax の範囲内の整数値となるように制限する関数です．

グローバル変数は，回転数指令値，検出値などです．int16_t は符号あり 16 ビット整数形です．これらの変数をグローバル変数とする理由は，デバッギング時に変数値が読めるからです．割り込み処理関数の中で，局所変数として定義（積分値は static 変数として定義）しても良いのですが，局所関数ではデバッガで読むことができません．プログラム作成・改変時には，変数値のモニタリングが効果的です．

TMR1_DefaultInterruptHandler() 関数内に，PI 制御のコードを記述しました．

$$\text{v\_wcom} = (\text{int16\_t})\text{ADC\_GetConversion}(0\text{x}06);$$
$$\text{v\_w} = (\text{int16\_t})\text{ADC\_GetConversion}(0\text{x}02);$$

により，AN6(8 番ピン) から $v_{\Omega COM}$ を読み込み，AN2(11 番ピン) から v_{Ω} を読み込みます．

右辺先頭の int16_t は，符号あり 16 ビット整数型への型変換です．

```
v_wdiff = v_wcom - v_w;
   ...
v_com = limiter(v_com);
```

では，時刻 k における電圧指令値を v_{COMk} とすると

$$v_{COMk} = K_p(v_{\Omega COMk} - v_{\Omega k}) + K_{idt}\sum_k (v_{\Omega COMk} - v_{\Omega k}) \tag{1.19}$$

を計算しています．K_p, K_{idt} はそれぞれ比例ゲイン，積分ゲインです．$v_{\Omega COMk}, v_{\Omega k}$ は，それぞれ時刻 k における回転数指令値 $v_{\Omega COM}$ のサンプル値，回転数検出値 v_Ω のサンプル値です．回転数の PI 制御を行っています．

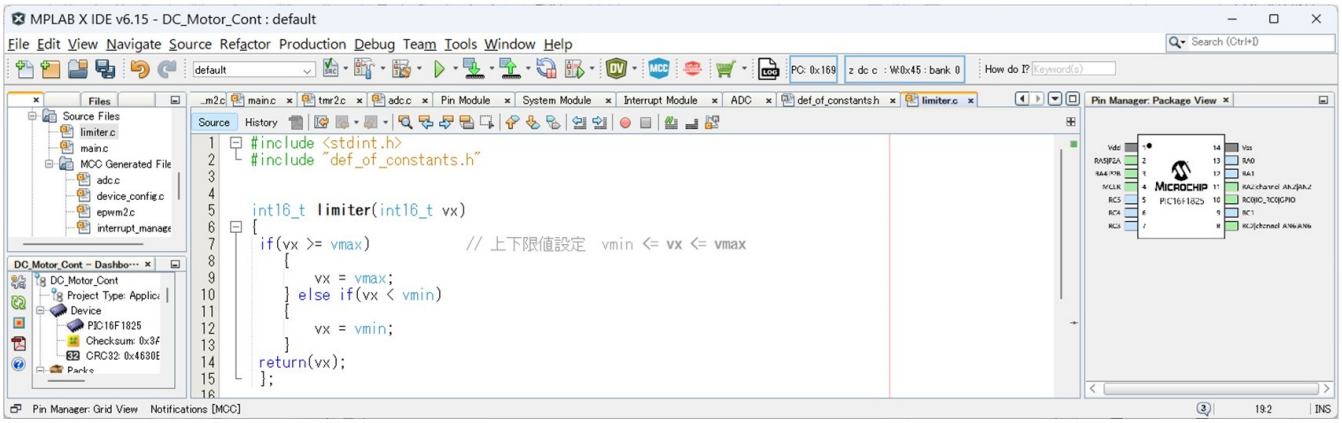


図 1.86:

図 1.86 は limiter() 関数です．

```
#include <stdint.h>
```

により，int16_t の定義を読み込みます．

```
#include “ def_of_constans.h ”
```

により，図 1.84 で宣言された定数値 vmax, vmin を読み込みます．図 1.85 内の 1 番目のリミッタは，(1.19) 式の右辺第 2 項の積分値が上下限値を超えないようにしています．2 番目のリミッタは電圧指令値 v_comk を上下限値内に抑えています．

1.6.8　回転数制御実験結果

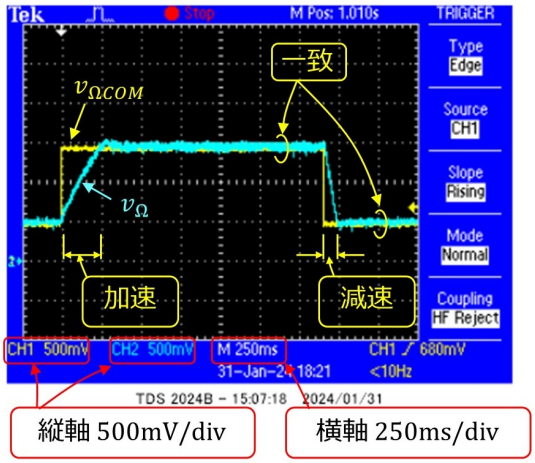


図 1.87: DC モータ回転数制御の実験結果

図 1.87 は図 1.1 の DC モータの回転数制御回路による実験結果を示します．$K_p = 32, K_{idt} = 1$ の場合です．図 1.4 の回路における回転数指令値 $v_{\Omega COM}$ と回転数検出値 v_Ω です．$v_{\Omega COM}$ のステップ的な変化に応じて，v_Ω が追従し，整定後は指令値と検出値が一致しています．ステップ変化直後の v_Ω の加速と減速では，減速の方が早くなりました．これは，加速時にはインバータの加速電流によるトルクとモータの摩擦力が反対方向であるのに対して，減速時には両力が同方向であることによります．

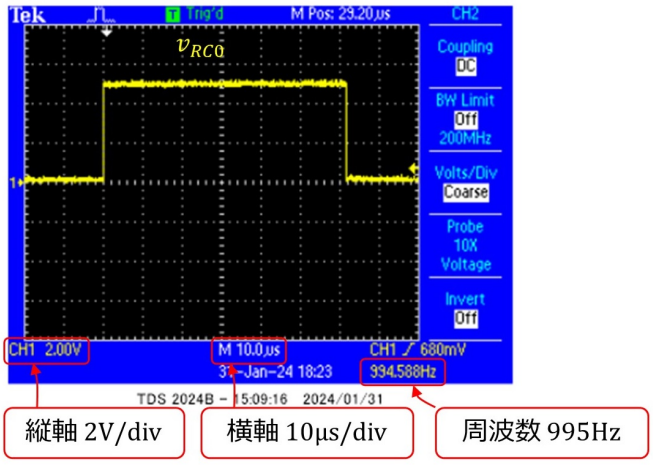


図 1.88: DC モータの回転数制御プログラムの処理時間

図 1.88 は，図 1.85 のタイマ 1 割り込み処理関数の処理時間です．約 62 $[\mu s]$ を要しました．

索 引

関連図書

[1] 後閑哲也「C 言語 & MCC による PIC プログラミング大全」技術評論社，2023.

[2] 古橋武「トラ技 Jr. 特集記事　基本回路 10 選！ブレッドボード実験室」 2022 年冬号（通巻 48 号）

[3] 「ジュニアのためのブレッドボード実験室 (4) 第 4 回はじめての PIC マイコン DC モータ制御」 トランジスタ技術 2022 年 6 月号

[4] モータドライブノート

[5] 古橋武「パワーエレクトロニクスノート」コロナ社，2008.

[6] 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
furuhashi.takeshi*

*に @gmail.com を付けてください．