ものづくりコンテスト電子回路組立
出力回路練習基板・
入力回路練習基板
プログラム解説マニュアル
(R8C/38A 版)

本マニュアルで 説明しているセット内容	・ものづくりコンテスト電子回路組立　出力回路練習基板 ・ものづくりコンテスト電子回路組立　入力回路練習基板
本基板の 対象マイコンボード	RY_R8C38 ボード
本基板の プログラムについての説明	本マニュアルで説明しています。

第 1.00 版
2016.02.03
株式会社日立ドキュメントソリューションズ

注 意 事 項 (rev.6.0H)

著作権

・本マニュアルに関する著作権は株式会社日立ドキュメントソリューションズに帰属します

・本マニュアルは著作権法および、国際著作権条約により保護されています。

禁止事項

ユーザーは以下の内容を行うことはできません。

・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと

・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること

・本マニュアルの一部または全部を改変、除去すること

・本マニュアルを無許可で翻訳すること

・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

転載、複製

本マニュアルの転載、複製については、文書による株式会社日立ドキュメントソリューションズの事前の承諾が必要です。

責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したものですが万一本マニュアルの記述誤りに起因する損害が生じた場合でも、株式会社日立ドキュメントソリューションズはその責任を負いません。

その他

・本マニュアルに記載の情報は本マニュアル発行時点のものであり、株式会社日立ドキュメントソリューションズは、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、最新の内容を確認いただきますようお願いします。

・すべての商標および登録商標は、それぞれの所有者に帰属します。

連絡先

株式会社 日立ドキュメントソリューションズ

〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー

E-mail：himdx.m-carrally.dd@hitachi.com

目　次

1. 概要

本マニュアルは、株式会社日立ドキュメントソリューションズのマイコンカーラリー販売サイトで販売している「**ものづくりコンテスト電子回路組立　出力回路練習基板**」と「**ものづくりコンテスト電子回路組立　入力回路練習基板**」を使用した、プログラム作成について、説明します。マイコンボードは、RY_R8C38 ボードを使用します。
プログラムは、ものづくりコンテストの電子回路組立部門の課題として使えるような内容にしています。

※本マニュアルの内容は、第 11 回高校生ものづくりコンテスト全国大会・電子回路組立部門の資料、課題を参考に作成しています。

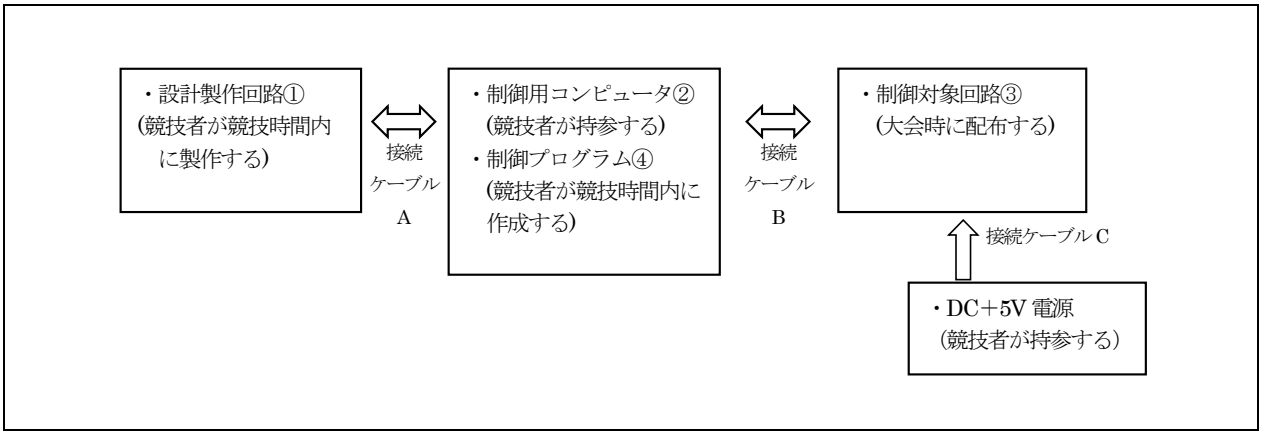
※高校生ものづくりコンテストについては、全国工業高等学校長協会のホームページ（アドレス：http://www.zenkoukyo.or.jp/）を参照してください。

※高校生ものづくりコンテスト・電子回路組立部門については、全国情報技術教育研究会のホームページ（アドレス：http://www.zenjouken.com/?page_id=63）を参照してください。

▲第 11 回高校生ものづくりコンテスト全国大会・電子回路組立部門　競技中の様子

2. 構成

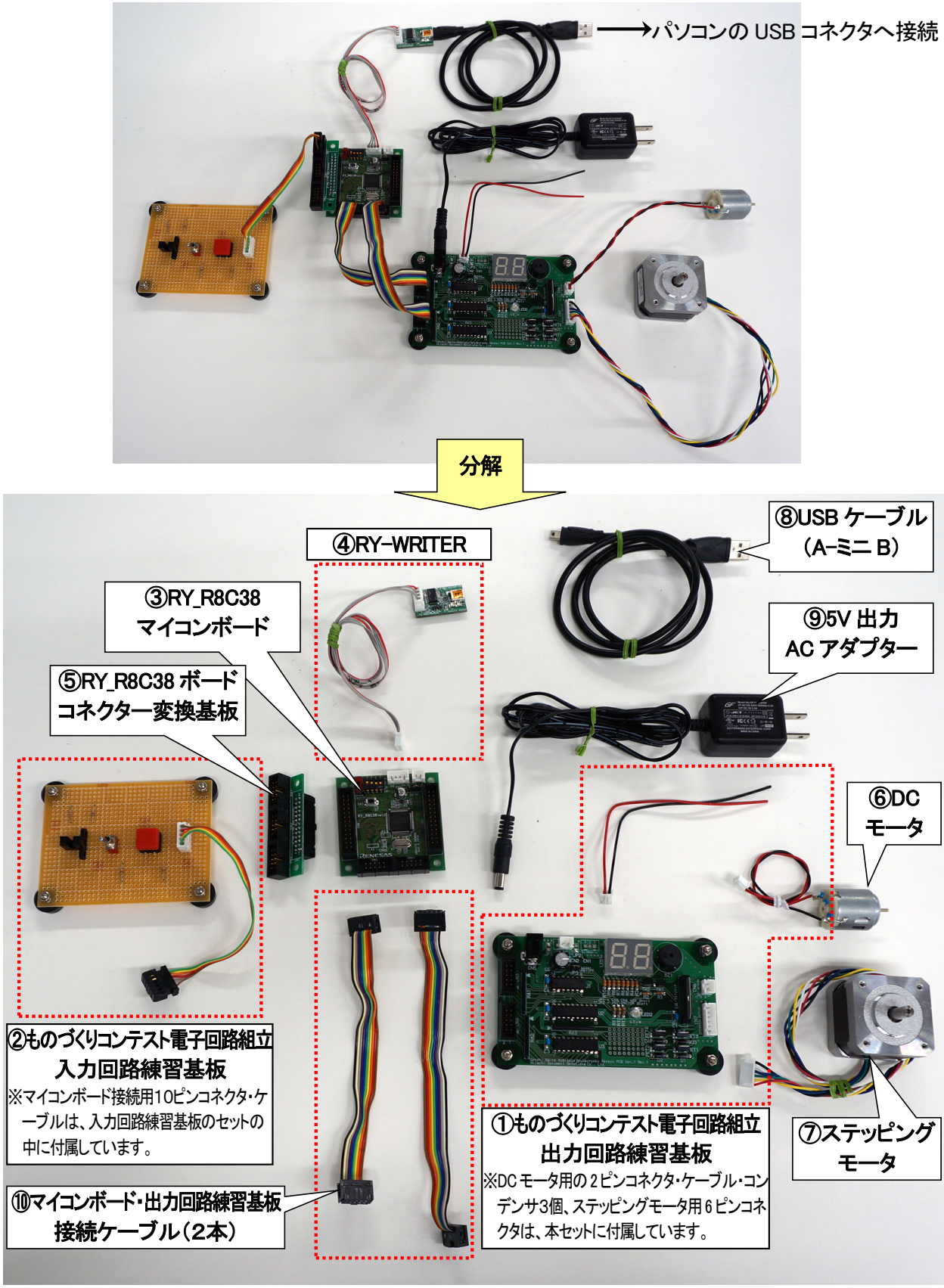
全体の接続構成を下図に示します。



設計製作回路①	今回は、「ものづくりコンテスト電子回路組立　入力回路練習基板」が該当します。
制御用コンピュータ②	制御用コンピュータは競技者が自由に構築することができます。全国大会の事前配付資料には、下記のように記載されています。 **開発環境を含め全て持参する。コンピュータの性能・形状等に制限はない。** 今回の制御用コンピュータ②の構成を、下記に示します。 ・制御用コンピュータ②として、ルネサス エレクトロニクス(株)製の R8C/38A マイコンを搭載した、RY_R8C38 ボードを使用します。 ・R8C/38A マイコンのプログラム開発として、ルネサス統合開発環境を使用し、言語は、C 言語を使用します。 ・ルネサス統合開発環境での開発には、Windows Vista、または Windows7 搭載のパソコンを使用します。
制御対象回路③	大会当日に配布される基板です。全国大会の事前配付資料には、下記のように記載されています。 **大会当日に競技実行委員から配布する。制御対象回路には制御対象駆動回路と制御対象が含まれている。なお、制御対象としては、次のようなものが考えられる。** **①LED** **②7セグメントLED** **③DCモータ** **④ステッピングモータ等** 今回は、「ものづくりコンテスト電子回路組立　出力回路練習基板」が該当します。
制御プログラム④	全国大会の事前配付資料には、下記のように記載されています。 **大会当日に提示する仕様に基づいたプログラムを作成する。使用する言語は自由である。なお、プログラムの仕様例として、次のようなものがある。** **①ストップウォッチのプログラム** **②回転制御のプログラム** 今回は、ルネサス統合開発環境を使い、C 言語でプログラムを行います。

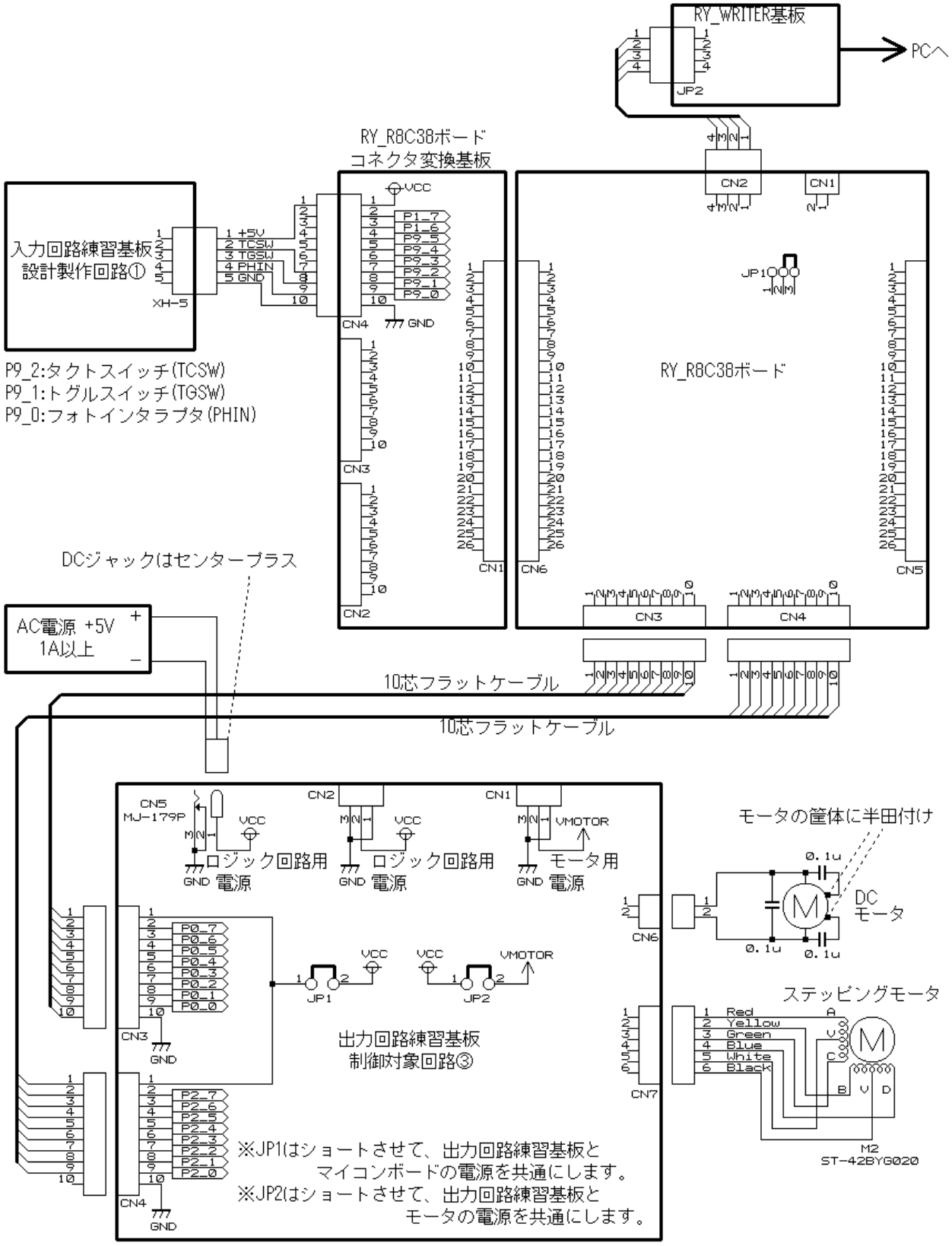
2.1 今回の結線

今回、使用する結線を下記に示します。



番号	マイコンカーラリー販売 型式	マイコンカーラリー販売 製品名	解説
①	M-S323	ものづくりコンテスト電子回路組立 出力回路練習基板	DC モータ用のコード・積層セラミックコンデンサ 3 個・コネクタ、ステッピングモータ用のコネクタを含んだセットです。
②	M-S322	ものづくりコンテスト電子回路組立 入力回路練習基板	
③	M-S181	RY_R8C38	マイコンボードです。 26 ピンコネクタオスは、「RY_R8C38 ボード　コネクター変換基板」に付属しているものを使用しています。「RY_R8C38」のセットには含まれません。
④	M-S183	RY-WRITER	RY_R8C38 ボード用のプログラム書き込み基板です。
⑤	M-S185	RY_R8C38 ボード コネクター変換基板セット	RY_R8C38 ボードの 26 ピンコネクタを 10 ピンコネクタ 3 個に変換する基板です。
⑥	M-S19	モーター (RC-260RA 18130(MCR 刻印入り))	定格 5～12V 程度の DC モータを使ってください。 このモータは、定格 6V です。
⑦	M-S320	ステッピングモータ(ST-42BYG0506H)	定格 5～12V 程度の DC モータを使ってください。 このステッピングモータは、定格 5V です。
⑧	M-S333	USB ケーブル(A-ミニ B　1.5m)	家電量販店で売られている USB ケーブル(A-ミニ B)で構いません。
⑨	M-S206	アダプターセット	5V を出力する AC アダプタです。出力回路練習基板の CN1 から 5V を供給する場合は、AC アダプタは必要ありません。
⑩	M-S43	10P メスコネクタ(PS-10SEN-D4P1-1C)	4 個必要です。
	M-S45	10 色フラットケーブル(1.27mm ピッチ)	約 18～30cm 程度のケーブルが 2 本必要です。

今回の結線を、下図に示します。



※制御対象回路③の回路図にあるステッピングモータについて、コイルの記載は本マニュアルでは C を $\overline{A}$、D を $\overline{B}$としています。

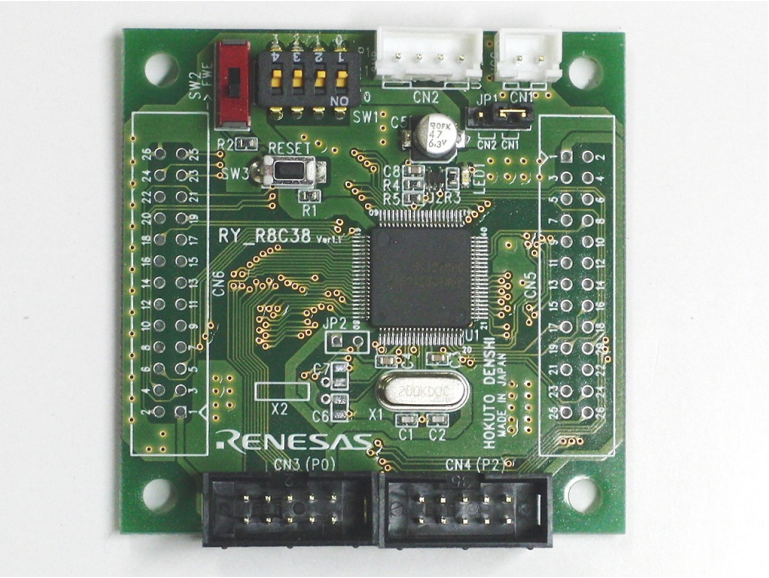
2.2 作業時間

電子回路組立部門は主に、①回路図作成　②基板製作、　③課題（プログラム）作成(7 問)　を制限時間である 2 時間 30 分(150 分)以内で終わらせなければいけません。そのため、時間配分が重要になってきます。生徒にもよりますが、時間配分例を下記に示します。

項目		時間［分］
概要の把握（ざっと読む）		3
回路図作成		7
基板製作		30
プログラム	共通部分ののプログラム	40
	課題 1～7 の main 部分のプログラム	70
合計		150

2.3 制御用コンピュータ②

本マニュアルでは、RY_R8C38 ボードを使用します。本ボードは、ルネサス エレクトロニクス製の R8C/38A マイコンを搭載したボードです。詳しい仕様やサンプルプログラムは、マイコンカーラリー販売サイトのダウンロードページ（https://www2.himdx.net/mcr/product/download.html）にある「マイコン実習マニュアル(R8C/38A 版)」を参照ください。

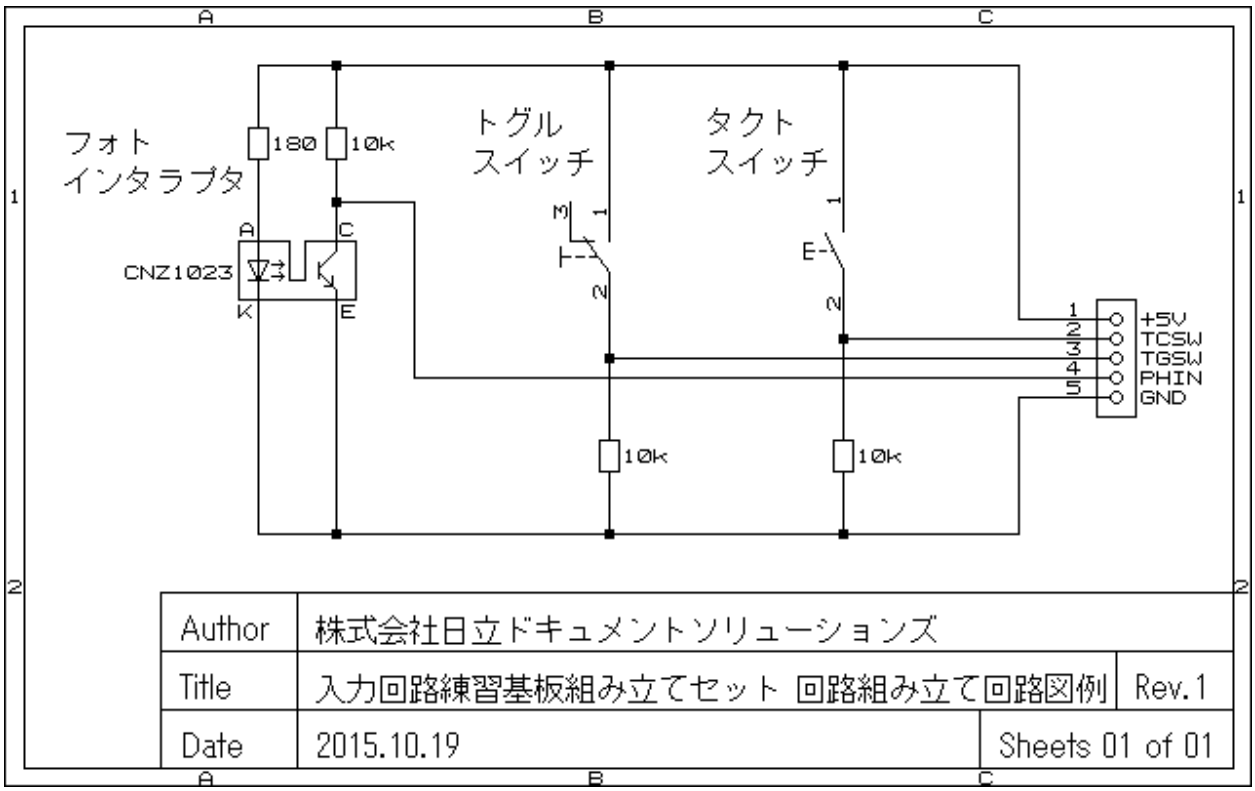


▲RY_R8C38 ボード（マイコンはルネサス エレクトロニクス製の R8C/38A）

2.4 設計製作回路①

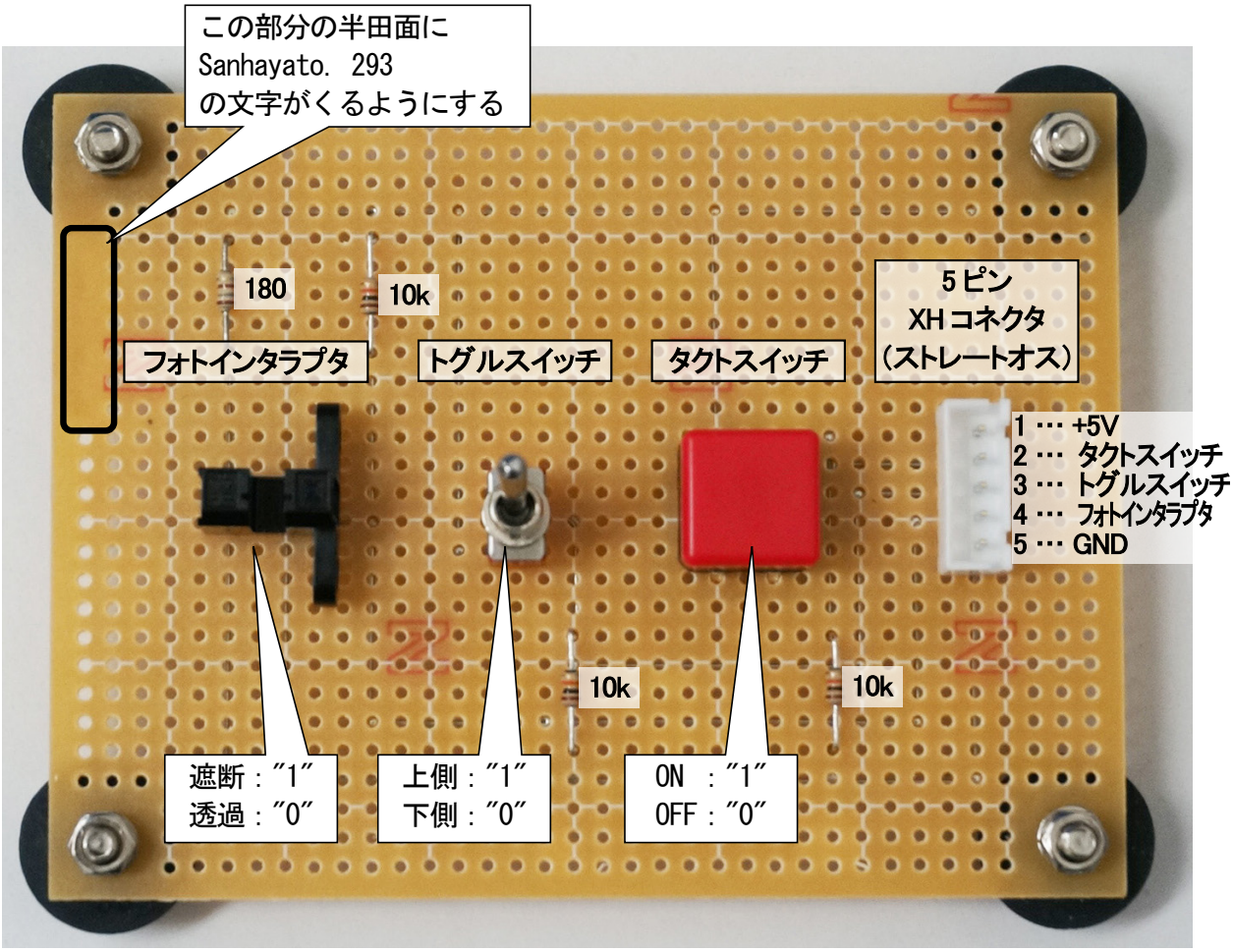
設計製作回路①は部品が支給され、競技時間内に競技者が製作します。
今回は、「ものづくりコンテスト電子回路組立　入力回路練習基板」を使用します。

今回、製作する回路図を、下記に示します。



▲回路図

今回、製作する部品実装位置を、下記に示します。

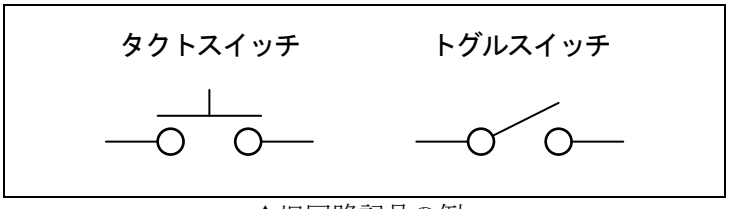


▲ものづくりコンテスト電子回路組立　入力回路練習基板　実装位置

■回路図についてポイント

課題に対応するために、回路を組み立てる必要がありますが、まずその回路図を正確に書くことが重要です。採点基準にも書かれていますが、回路図の配置は適当か？回路記号はあっているか？数値は記入されているか？配線は正しいか？端子にピン番号やピンの名称が書かれているか？そのまま作れば正しく動作するか？などが重要です。

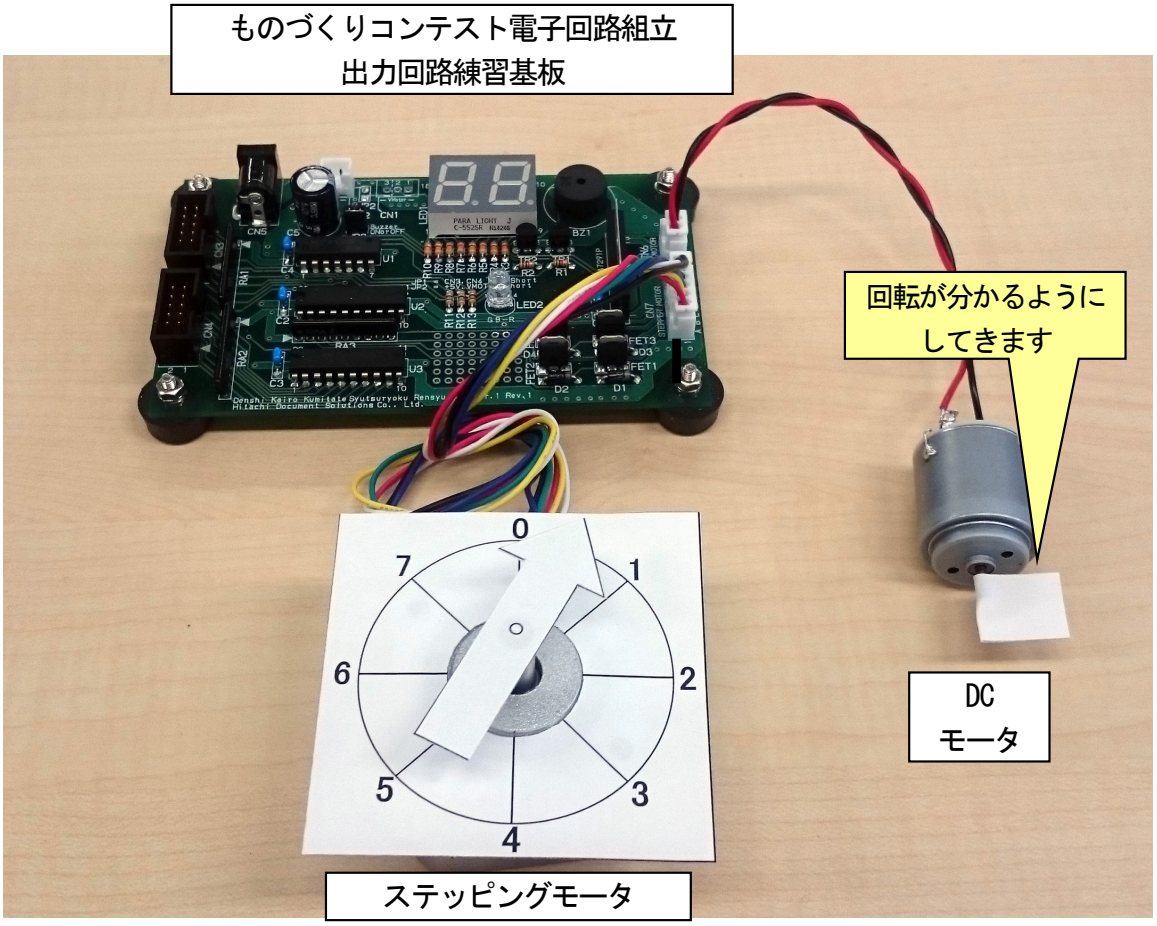
配置は、基本的に信号の流れは左から右に、電圧は高いほうが上、低いほうが下です。入出力一体回路などは必ずしも左から右に書けないこともありますが、今回は入力回路ですのでスイッチやフォトカプラは左側にあって、入力回路の出力端子が一番右側にあるのがいいと思います。回路記号は新記号（JIS C 0617、IEC 60617）に準拠しますが、まだ書籍や部品表などに旧記号（JIS C 0301）が使われている場合があるので、当面旧記号も認められています。抵抗などの数値は必須です。端子はピンの丸とそれらを囲む四角で書かれますが、ピン番号(1,2,3・・・)とピン名称（5V,GND,TCSW など）も記入しましょう。図面の大きさと紙の大きさのバランスも考慮して見やすい回路図を書きましょう。そして回路図ができたら、それを見ながら実際の回路を組み立てていきます。



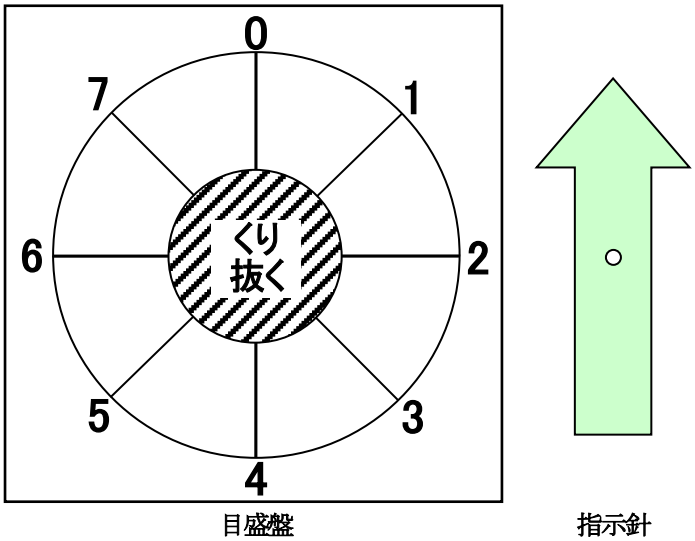
▲旧回路記号の例

2.5 制御対象回路③

制御対象回路③は、大会で配布される基板です。今回は、「ものづくりコンテスト電子回路組立　出力回路練習基板」を使用します。ステッピングモータ、DC モータの使用例を下記に示します。



ステッピングモータに貼る、目盛盤と指示針を下記に示します。そのまま使う場合は、1 倍で印刷して、切り取って使ってください。



2.6 出力信号のポート

各出力先とR8C/38Aマイコンの接続を、下記に示します。

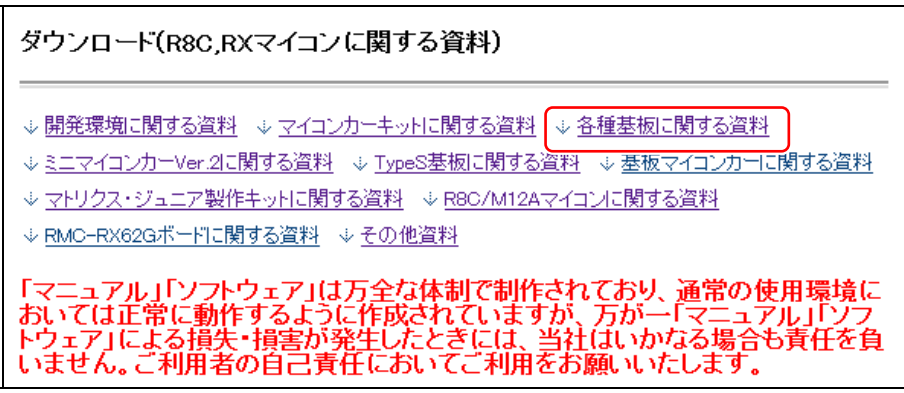
出力先		マイコンのポート	"1"を出力したときの状態	"0"を出力したときの状態	備考
LED	緑	p0_7	点灯	消灯	JP3 をショートさせると圧電サウンダーも同時に接続されます。同時に接続されるだけで、LED には影響はありません。
	青	p0_6	点灯	消灯	
	赤	p0_5	点灯	消灯	
圧電サウンダー		p0_7	"0"と"1"のパルスを送ることにより音を鳴らします。		JP3 をショートさせると、p0_7 と圧電サウンダーが接続されます。使用するときは、JP3 をショートさせてください。ただし、p0_7 は緑色 LED と兼用になっています。圧電サウンダーを制御するとき、緑色の LED も同時に点灯します。
左側7セグメントLED	カソードコモン端子の選択	p0_1	p2_7～p2_0 の値に従って点灯	全消灯	7セグメントLEDはカソードコモンで、このコモン端子を ON / OFF するポートです。 p0_1="0"なら各セグメントLEDは p2_7～p2_0 の値に関係なく消灯します。
	各セグメントのLED制御	p2_7(dp セグメント)	点灯	消灯	p0_1="1"（点灯）のとき、p2_7～p2_0 でどの LED を点灯させるか制御します。 右側 7 セグメント LED と共通なので、p0_1="1"のときだけ、左側 7 セグメント LED に出力したい値を設定します。
		p2_6(g セグメント)	点灯	消灯	
		p2_5(f セグメント)	点灯	消灯	
		p2_4(e セグメント)	点灯	消灯	
		p2_3(d セグメント)	点灯	消灯	
		p2_2(c セグメント)	点灯	消灯	
		p2_1(b セグメント)	点灯	消灯	
		p2_0(a セグメント)	点灯	消灯	
右側7セグメントLED	カソードコモン端子の選択	p0_2	p2_7～p2_0 の値に従って点灯	全消灯	7セグメントLEDはカソードコモンで、このコモン端子を ON/OFF するポートです。 p0_2="0"なら各セグメント LED は p2_7～p2_0 の値に関係なく消灯します。
	各セグメントのLED制御	p2_7(dp セグメント)	点灯	消灯	p0_2="1"（点灯）のとき、p2_7～p2_0 でどの LED を点灯させるか制御します。 左側 7 セグメント LED と共通なので、p0_2="1"のときだけ、右側 7 セグメント LED に出力したい値を設定します。
		p2_6(g セグメント)	点灯	消灯	
		p2_5(f セグメント)	点灯	消灯	
		p2_4(e セグメント)	点灯	消灯	
		p2_3(d セグメント)	点灯	消灯	
		p2_2(c セグメント)	点灯	消灯	
		p2_1(b セグメント)	点灯	消灯	
		p2_0(a セグメント)	点灯	消灯	

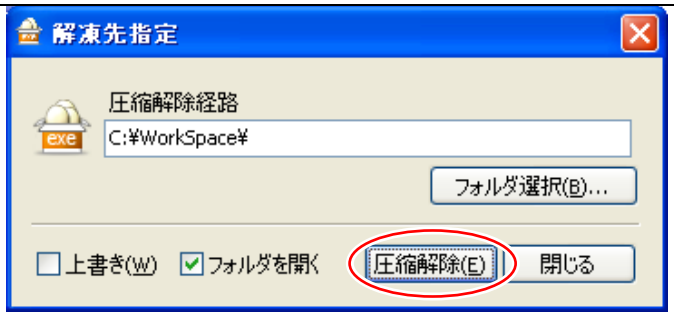
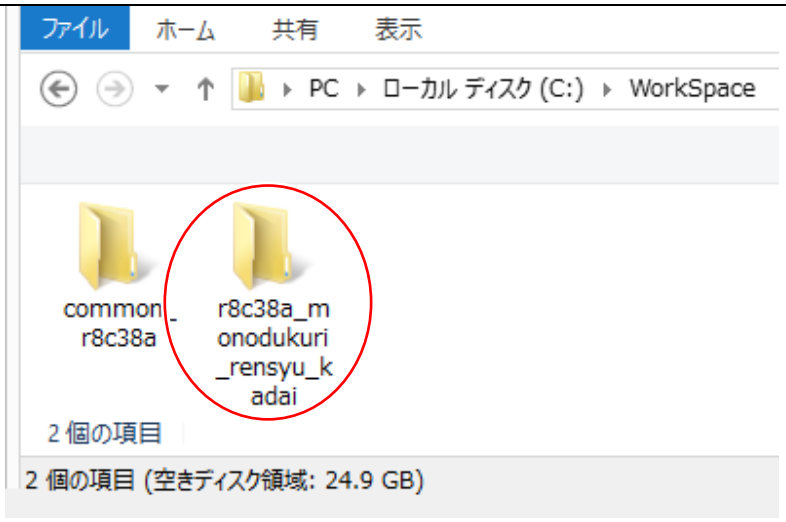
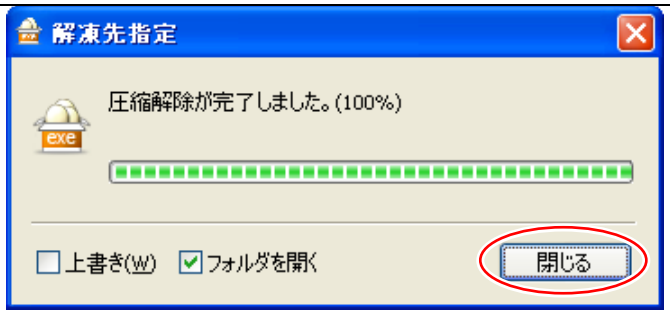
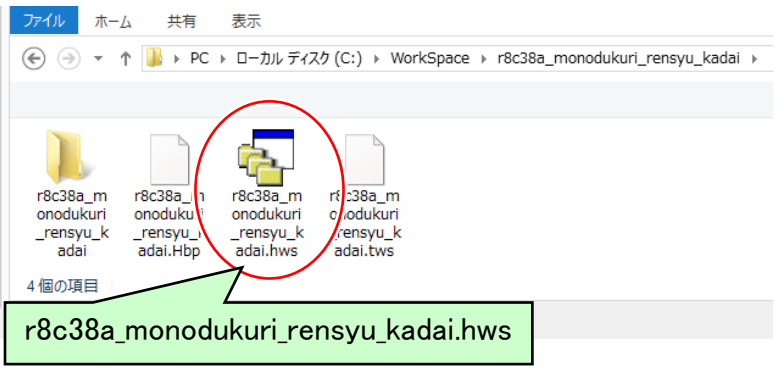
（次のページへ続く）

<table>
<tr><th colspan="2">出力先</th><th>マイコンの
ポート</th><th>″1″を出力した
ときの状態</th><th>″0″を出力した
ときの状態</th><th>備考</th></tr>
<tr><td rowspan="5">ステッピングモータ</td><td>74HC574
出力端子</td><td>p0_0</td><td>″0″→″1″になった瞬間に p2_3～p2_0 の信号が、ステッピングモータに出力される</td><td>変化なし</td><td>″1″にした瞬間に 74HC574 の出力端子から p2_3～p2_0 の値が出力され、ステッピングモータが動作します。″1″にした後は、74HC574 によって出力値が保持されるので p2_3～p2_0 の値を変化させてもモータの動作は変わりません。</td></tr>
<tr><td rowspan="4">励磁コイルの選択</td><td>p2_3($\overline{B}$)</td><td>$\overline{B}$コイルを
励磁する</td><td>$\overline{B}$コイルを
励磁しない</td><td rowspan="4">A→B→$\overline{A}$→$\overline{B}$ の順に励磁するとステッピングモータは反時計回りし、その逆の順番に励磁すると時計回りします。1 回の励磁で 1.8 度動きます。1 周は 360 度なので、1 周させるために励磁する回数は、
360÷1.8＝400 パルスです。</td></tr>
<tr><td>p2_2($\overline{A}$)</td><td>$\overline{A}$コイルを
励磁する</td><td>$\overline{A}$コイルを
励磁しない</td></tr>
<tr><td>p2_1(B)</td><td>B コイルを
励磁する</td><td>B コイルを
励磁しない</td></tr>
<tr><td>p2_0(A)</td><td>A コイルを
励磁する</td><td>A コイルを
励磁しない</td></tr>
<tr><td colspan="2">DC モータの
動作選択</td><td>p0_4、p0_3</td><td colspan="2">″00″:ストップ
″01″:時計回りに回転
″10″:反時計回りに回転
″11″:ブレーキ</td><td>ストップは DC モータの端子間を解放、ブレーキは DC モータの端子間をショートさせます。</td></tr>
</table>

3. ワークスペース(プログラム)のダウンロード、インストール

3.1 ワークスペース(プログラム)のダウンロード

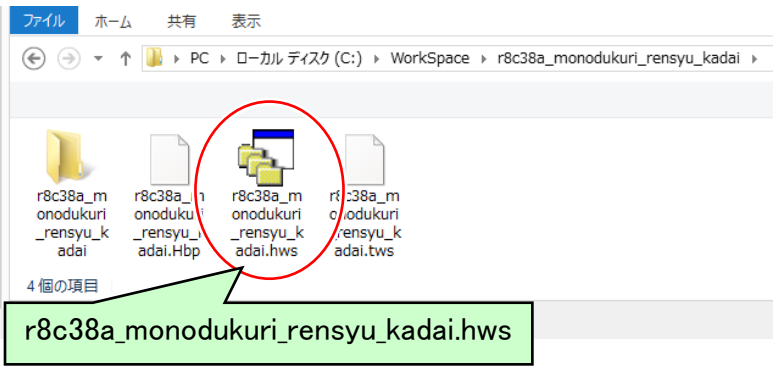
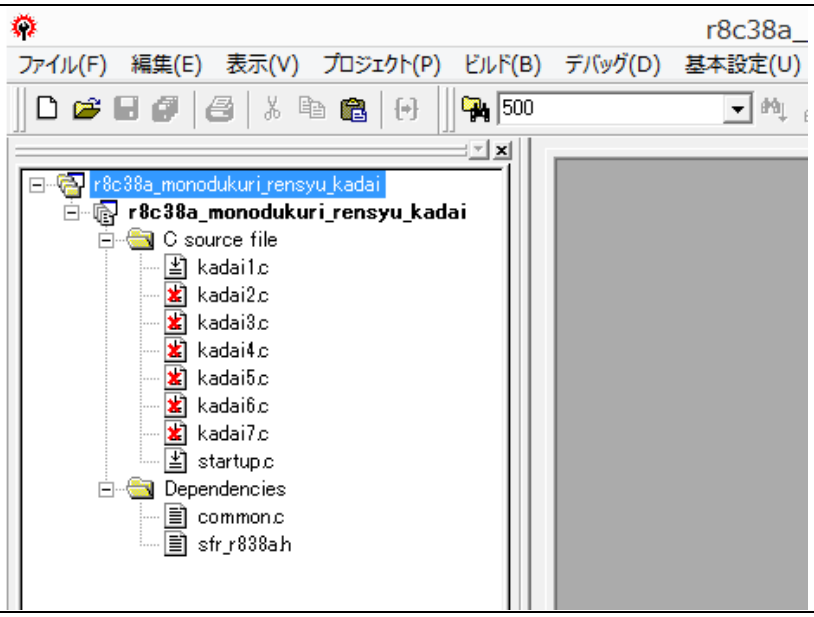
1		マイコンカーラリー販売サイト https://www2.himdx.net/mcr/ にアクセスします。 「ダウンロード」をクリックします。
2		「各種基板に関する資料」をクリックします。
3		「r8c38a_monodukuri_rensyu_kadai.zip」をダウンロードして、解凍します。

4		「r8c38a_monodukuri_rensyu_kadai.exe」を実行して、圧縮解除をクリックします。 **※フォルダは変更できません。変更した場合は、ルネサス統合開発環境の設定を変更する場合がでてきます。**
5		解凍が終わったら、自動的に「C ドライブ→Workspace」フォルダが開かれます。今回使用するのは、「r8c38a_monodukuri_rensyu_kadai」です。
6		閉じるをクリックして終了です。
7		「C ドライブ→ Workspace→ r8c38a_monodukuri_rensyu_kadai→ r8c38a_monodukuri_rensyu_kadai.hws をダブルクリックすると、ルネサス統合開発環境が立ち上がります。

※本マニュアルでは開発環境として、ルネサス統合開発環境(無償評価版)を使用します。ルネサス統合開発環境やその他ファイルの入手、インストール、操作方法については、マイコンカーラリー販売　ダウンロードページ(https://www2.himdx.net/mcr/product/download.html)にある「ルネサス統合開発環境　操作マニュアル(R8C/38A 版)」を参照してください。

3.2 プログラムの内容

ルネサス統合開発環境でのファイルの開き方、操作方法を説明します。

1		「C ドライブ→ Workspace→ r8c38a_monodukuri_rensyu_kadai→ r8c38a_monodukuri_rensyu_kadai.hws をダブルクリックすると、ルネサス統合開発環境が立ち上がります。
2		左側にあるリストが、プログラムファイルになります。

登録されているプログラムの内容を、下記に示します。

ファイル名	※	詳細
sfr_r838a.h	○	R8C/38A マイコンのレジスタを定義しているファイルです。全課題共通で使用します。 ファイルの場所:C:¥Workspace¥common_r8c38a¥sfr_r838a.h
startup.c	○	マイコン起動時のプログラム(スタートアップルーチン)が記載されているファイルです。全課題共通で使用します。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥startup.c
common.c	×	課題プログラムを作成する上で、ポートの入出力設定、入力回路部分のプログラム、出力回路部分のプログラムなど、全課題共通のプログラムをこのファイル内に入れておきます。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥common.c
kadai1.c	×	課題 1 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai1.c
kadai2.c	×	課題 2 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai2.c
kadai3.c	×	課題 3 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai3.c
kadai4.c	×	課題 4 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai4.c
kadai5.c	×	課題 5 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai5.c
kadai6.c	×	課題 6 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai6.c
kadai7.c	×	課題 7 の回答例が入っているファイルです。 ファイルの場所:C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥kadai7.c

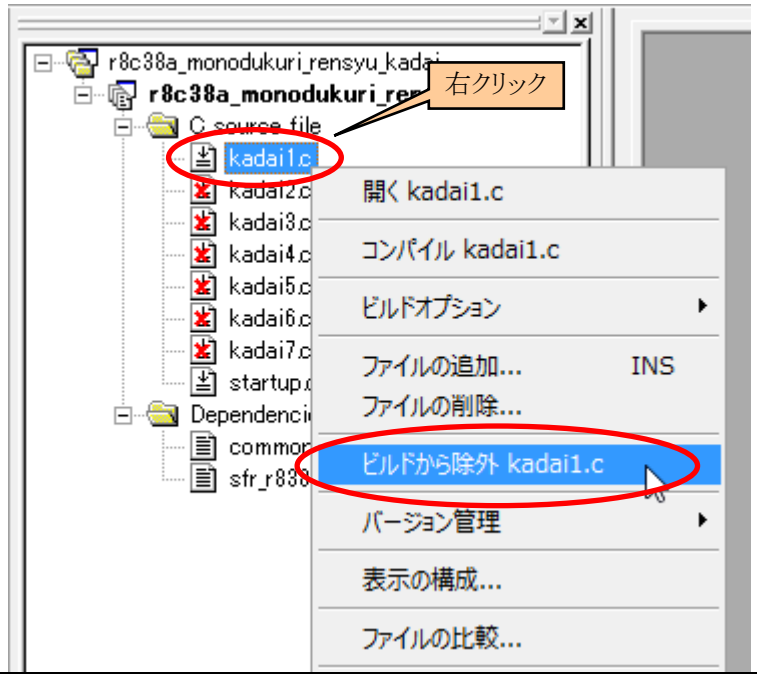
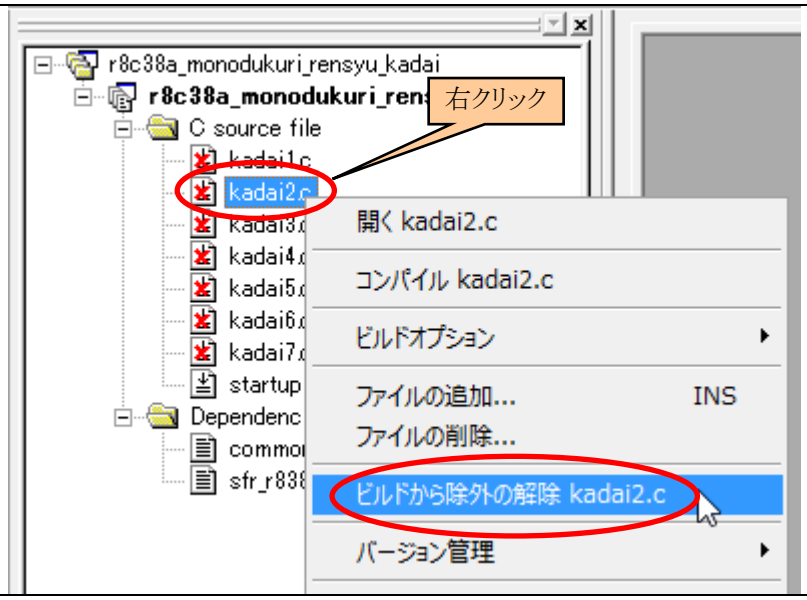
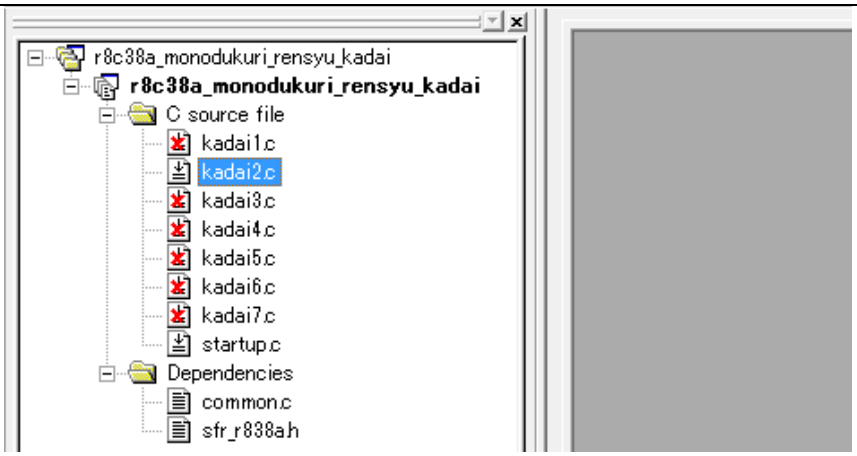
※　○は持ち込み可能　×はコンテスト時に空から作成するファイルです。

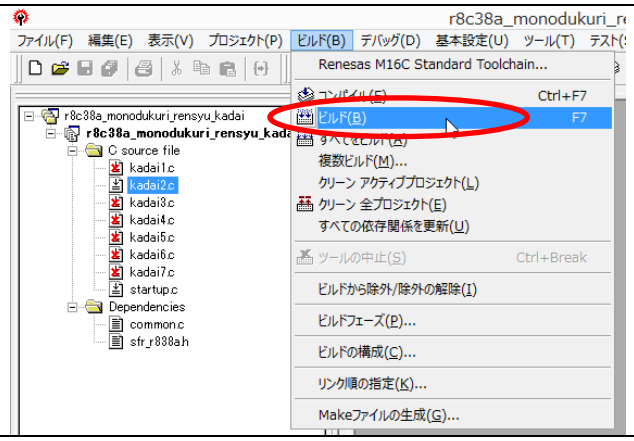
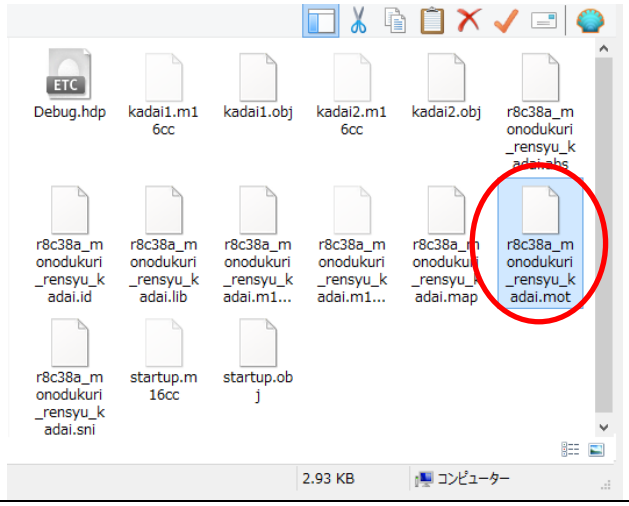
本プロジェクトには、kadai1.c～kadai7.c の課題ファイルが登録されていますが、この中で有効にできるのは 1 つだけです。例えば、課題 1 のときは、「kadai1.c」のみ有効にして、「kadai2.c～kadai7.c」はビルドから除外(ファイル左の赤い×マーク)にしておきます。

各課題のプログラムをビルドするときに、ビルドから除外するファイルを下記に示します(ファイルの詳しい構成は後述します)。

ファイル名	課題 1	課題 2	課題 3	課題 4	課題 5	課題 6	課題 7
startup.c	○	○	○	○	○	○	○
common.c	○	○	○	○	○	○	○
kadai1.c	○	×	×	×	×	×	×
kadai2.c	×	○	×	×	×	×	×
kadai3.c	×	×	○	×	×	×	×
kadai4.c	×	×	×	○	×	×	×
kadai5.c	×	×	×	×	○	×	×
kadai6.c	×	×	×	×	×	○	×
kadai7.c	×	×	×	×	×	×	○

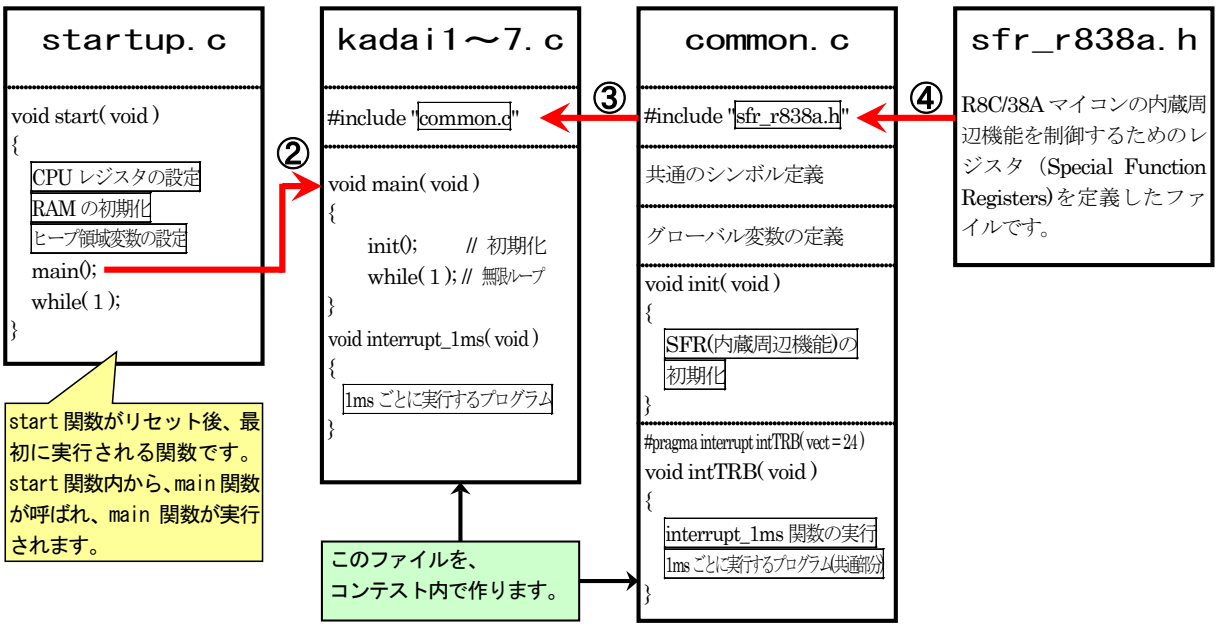
○:有効　×:ビルドから除外するファイル

3		例えば、課題 2 のファイルである、「kadai2.c」をビルド(MOT ファイルの作成)したい場合、次の操作を行います。 「kadai1.c」の上で右クリックし、「ビルドから除外」をクリックします。
4		「kadai2.c」の上で右クリックし、「ビルドから除外の解除」をクリックします。
5		リストが、左画面のようになれば完了です。

6		「ビルド→ビルド」で、kadai2.c などの登録されているファイルがビルド(アセンブル、コンパイル、リンク)され最終ファイル(MOT ファイル)ができあがります。
7		MOT ファイルは、 「C:¥Workspace¥r8c38a_monodukuri_rensyu_kadai¥r8c38a_monodukuri_rensyu_kadai¥Debug」フォルダ内にできます。
8		R8C Writer などの書き込みソフトを使って、プログラムを書き込んでください。

4. ファイルの構成

今回のファイル構成を下図に示します。



プログラムの動きを、下記に示します。

※「kadai○.c」の○は、1～7 の数字が入ります。

①	マイコンの電源が入ると、start 関数が実行されます。start 関数では、CPU レジスタの設定など、マイコンを動かすための設定を行います。
②	①が終わると、main 関数を実行します。
③	kadai○.c は、common.c ファイルをインクルードしてファイルを取り込みます。 common.c は、kadai○.c のファイルで共通で使う変数や関数を記載しているファイルです。
④	common.c は、sfr_r838a.h ファイルをインクルードしてファイルを取り込みます。 このファイルは、R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ（Special Function Registers）を定義したファイルです。

5.「startup.c」ファイル

このファイルは、マイコン固有の設定をまとめたファイルです。このまま使用します。変更する部分はありません。

```
/************************************************************************/
/* 対象マイコン R8C/38A                                                 */
/* ﾌｧｲﾙ内容     スタートアッププログラム(C言語版)                       */
/* バージョン   Ver.1.22                                                */
/* Date         2012.03.08                                              */
/* Copyright    ジャパンマイコンカーラリー実行委員会                    */
/*              ルネサスマイコンカーラリー事務局                        */
/*              日立インターメディックス株式会社                        */
/************************************************************************/
/*--------------------------------------*/
/* プロトタイプ宣言(ローカル)           */
/*--------------------------------------*/
void main( void );

/*--------------------------------------*/
/* スタックサイズの設定                 */
/*--------------------------------------*/
#pragma STACKSIZE       0x200
#pragma ISTACKSIZE      0x100

/*--------------------------------------*/
/* CPUレジスタの宣言                    */
/*--------------------------------------*/
#pragma CREG    _flg_   flg
#pragma CREG    _isp_   isp
#pragma CREG    _sp_    sp
#pragma CREG    _sb_    sb
#pragma CREG    _fb_    fb
#pragma CREG    _intbh_ intbh
#pragma CREG    _intbl_ intbl

unsigned int _flg_;
unsigned int _sb_;
unsigned int _fb_;
unsigned int *_sp_;
unsigned int *_isp_;
unsigned int *_intbh_;
unsigned int *_intbl_;

/*--------------------------------------*/
/* SBの値をコンパイラに設定             */
/*--------------------------------------*/
    _asm("      .glb    __SB__¥n"
         "__SB__ .equ    0400H   ");

/*--------------------------------------*/
/* オプション機能選択レジスタの設定     */
/*--------------------------------------*/
    _asm("  .ofsreg 0BFH ");            /* OFS = 0xbf (ﾊﾟﾜｰｵﾝﾘｾｯﾄ使用)  */

/*--------------------------------------*/
/* IDコードの設定                       */
/*--------------------------------------*/
    _asm("  .id ""¥""#FFFFFFFFFFFFFF¥"" ");

/*--------------------------------------*/
/* RAMを初期化する関数の定義            */
/*--------------------------------------*/
#define scopy(X,Y,Z)    _asm("  .initsct "X","Y","Z"¥n"¥
                "   .initsct "X"I,rom"Y",noalign¥n"¥
                "   mov.w   #(topof "X"I)&0ffffH,A0¥n"¥
                "   mov.b   #(topof "X"I)>>16,R1H¥n"¥
                "   mov.w   #(topof "X")&0ffffH,A1¥n"¥
                "   mov.w   #sizeof "X",R3¥n"¥
                "   smovf.b")

#define sclear(X,Y,Z)   _asm("  .initsct "X","Y","Z"¥n"¥
                "   mov.b   #00H,R0L¥n"¥
                "   mov.w   #(topof "X") ,A1¥n"¥
                "   mov.w   #sizeof "X",R3¥n"¥
                "   sstr.b")

/*--------------------------------------*/
/* セクションの先頭アドレスの型定義     */
/*--------------------------------------*/
extern unsigned int _stack_top;
extern unsigned int _istack_top;
extern unsigned int _vector_top;

/*--------------------------------------*/
/* ヒープ領域の設定                     */
/*--------------------------------------*/
#pragma sectaddress heap_NE,DATA
#define __HEAPSIZE__    0x100

unsigned char heap_area[ __HEAPSIZE__ ]; /* ヒープ領域確保              */
extern unsigned char  _far *_mnext;     /* 次に使用できるﾒﾓﾘの先頭ｱﾄﾞﾚｽ */
extern unsigned long        _msize;     /* 残りのバイト数              */

/*--------------------------------------*/
/* 固定割り込みベクタアドレスの設定     */
/*--------------------------------------*/
#pragma sectaddress fvector,ROMDATA 0xffd8

    _asm("  .addr 0FFFFFFFH");          /* 予約                         */
    _asm("  .byte 0FFH");               /* OFS2                         */
#pragma interrupt/v _dummy_int          /* 未定義命令割り込みベクタ     */
#pragma interrupt/v _dummy_int          /* オーバフロー割り込みベクタ   */
#pragma interrupt/v _dummy_int          /* BRK命令割り込みベクタ        */
#pragma interrupt/v _dummy_int          /* アドレス一致割り込みベクタ   */
#pragma interrupt/v _dummy_int          /* ｼﾝｸﾞﾙｽﾃｯﾌﾟ割り込みベクタ     */
#pragma interrupt/v _dummy_int          /* ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏなどの割り込みﾍﾞｸﾀ*/
#pragma interrupt/v _dummy_int          /* ｱﾄﾞﾚｽﾌﾞﾚｲｸ割り込みベクタ     */
#pragma interrupt/v _dummy_int          /* 予約                         */
#pragma interrupt/v start               /* リセットベクタ               */

/*--------------------------------------*/
/* 固定割り込みプログラム               */
/*--------------------------------------*/
#pragma sectaddress interrupt,CODE
#pragma interrupt _dummy_int()
void _dummy_int(void)
{
    /* ダミー関数 */
}

/*--------------------------------------*/
/* 可変割り込みベクタの設定             */
/*--------------------------------------*/
#pragma sectaddress vector,ROMDATA

    /* ここではセクション名の設定のみ行う */

/*--------------------------------------*/
/* RAMの初期化                          */
/*--------------------------------------*/
#pragma sectaddress program,CODE
void initsct(void)
{
    sclear("bss_SE","data","align");
    sclear("bss_SO","data","noalign");
    sclear("bss_NE","data","align");
    sclear("bss_NO","data","noalign");

    scopy("data_SE","data","align");
    scopy("data_SO","data","noalign");
    scopy("data_NE","data","align");
    scopy("data_NO","data","noalign");
}

/*--------------------------------------*/
/* スタートアッププログラム             */
/*--------------------------------------*/
#pragma entry start
void start( void )
{
    _isp_   = &_istack_top;             /* ISPに割り込みｽﾀｯｸのｱﾄﾞﾚｽ設定 */
    _flg_   = 0x0080;                   /* FLGのU="1"                   */
    _sp_    = &_stack_top;              /* USPにﾕｰｻﾞｽﾀｯｸのｱﾄﾞﾚｽ設定     */
    _sb_    = 0x0400U;                  /* SB相対アドレッシングの設定   */
    _intbh_ = (unsigned int*)0x00;      /* INTBH = vector(上位)に設定   */
    _intbl_ = &_vector_top;             /* INTBL = vector(下位)に設定   */
    initsct();                          /* RAMの初期化                  */
    _mnext = &heap_area[0];             /* ヒープ領域変数の設定         */
    _msize = (unsigned long)__HEAPSIZE__;
    _fb_    = 0U;                       /* FB = 0                       */

    main();                             /* main関数実行                 */

    while(1);
}

/************************************************************************/
/* end of file                                                          */
/************************************************************************/
```

6.「common.c」ファイル

このファイルは、課題 1～7 のプログラムを作成するとき、共通で使用する定義、グローバル変数、関数などを記載します。コンテストでは、このファイルは 0 から作成します。回答例のプログラム内容を説明します。

6.1 init 関数

init 関数は、R8C/38A マイコンの内蔵周辺機能に関わる設定を行います。init は、「initialize（イニシャライズ）」の略です。

```
void init( void )
{
    // R8C/38A マイコンの内蔵周辺機能の初期化
    ①CPU の動作クロックの切り替え
    ②ポートの入出力設定
    ③タイマ RB の設定
    ④割り込みの許可
}
```

6.1.1 CPU の動作クロックの切り替え

R8C/38A マイコンは起動時、内蔵の低速オンチップオシレータというクロックで動作しています。このクロックは 125kHz と遅いので、外付けしている 20MHz のクリスタルに切り替えます。

```
 99 :      int i;
100 :
101 :      // クロックを XIN クロック(20MHz)に変更
102 :      prc0  = 1;                          // プロテクト解除
103 :      cm13  = 1;                          // P4_6,P4_7 を XIN-XOUT 端子にする
104 :      cm05  = 0;                          // XIN クロック発振
105 :      for(i=0; i<50; i++ );               // 安定するまで少し待つ(約 10ms)
106 :      ocd2  = 0;                          // システムクロックを XIN にする
107 :      prc0  = 0;                          // プロテクト ON
```

レジスタの設定について、詳しくは「マイコン実習マニュアル(R8C/38A 版)」を参照してください。

6.1.2 ポートの入出力設定

R8C/38A マイコンは、ポートが 0～9 まで 10 個あります。ポートの端子を入力端子にするか出力端子にするか設定します。リセット後は、全端子が入力端子です。

(1) ポートの接続

ポートの接続を下記に示します。記入の無いビットは未接続、斜線の欄は端子が無いビットです。

ポート	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**	**制御対象 回路③ CN3 （出力）**
1			RxD0 入力	TxD0 出力	RY_R8C38 ボード上の SW3 入力	RY_R8C38 ボード上の SW2 入力	RY_R8C38 ボード上の SW1 入力	RY_R8C38 ボード上の SW0 入力
2	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**	**制御対象 回路③ CN4 （出力）**
3								
4	クリスタル 出力	クリスタル 入力	RY_R8C38 ボード上の LED 出力	時計用 クリスタル ※未接続 出力	時計用 クリスタル ※未接続 出力	Vcc 入力		
5								
6								
7								
8								
9						**設計製作 回路① タクト スイッチ （入力）**	**設計製作 回路① トグル スイッチ （入力）**	**設計製作 回路① フォト インタラプタ （入力）**

※表の斜線の bit は、端子がない bit です。

※リセット後は、全て入力ポートです。

(2) プログラム

ポートの入出力設定を行う前に、出力にする端子の出力値を設定しておきます。今回ポート 0 は、LED 消灯、DC モータ停止、7 セグメント LED 消灯にするため、0xe0 を設定していきます(102 行)。

ポートの入出力設定は、pd レジスタで行います。pd0～pd9 まであります。R8C/38A マイコンは「pd0」を設定する直前に必ず、「prc2 = 1」を入れなければいけないという決まりがあるので 105 行を入れます(pd0 以外は、決まりはありません)。

リセット直後、全端子は入力端子になっています。ポート 0 とポート 2 の端子を出力に設定します。その他は、今回特に使わないので、リセット直後のまま(入力のまま)にしておきます。

※メーカーは、未接続端子を入力のまま未接続にするとノイズが入り端子が壊れることがあるため、必ず出力端子に設定してください、と謳っています。今回は時間が限られているので、ポート 0 とポート 2 以外は操作しません。今回の構成では、大きなノイズが入ってきづらい使い方なので大丈夫だと思われますが、本来は、全端子、入出力設定を行ってください。

```
101 :      // ポートの初期出力
102 :      p0 = 0xe0;                          // LED 消灯、DC モータ停止 7 セグ消灯
103 :
104 :      // ポートの入出力設定
105 :      prc2 = 1;                           // PD0 のプロテクト解除
106 :      pd0 = 0xff;                         // 出力に設定
107 :      pd2 = 0xff;                         // 出力に設定
```

6.1.3 タイマ RB の設定

タイマ RB を使って、1ms ごとに割り込みが発生するよう設定します。

```
109 :      // タイマ RB の設定
110 :      // 割り込み周期 = 1 / 20[MHz]    * (TRBPRE+1) * (TRBPR+1)
111 :      //              = 1 / (20*10^6) * 200        * 100
112 :      //              = 0.001[s] = 1[ms]
113 :      trbmr  = 0x00;                      // 動作モード、分周比設定
114 :      trbpre = 200-1;                     // プリスケーラレジスタ
115 :      trbpr  = 100-1;                     // プライマリレジスタ
116 :      trbic  = 0x07;                      // 割り込み優先レベル設定
117 :      trbcr  = 0x01;                      // カウント開始
```

レジスタの設定について、詳しくは「マイコン実習マニュアル(R8C/38A 版)」を参照してください。

6.1.4 割り込みの許可

タイマ RB を使って、1ms ごとに割り込みを発生させるよう設定しました。R8C マイコンは、個別の割り込み設定の他に、全体の割り込みを許可する設定が必要です。プログラムを下記に示します。

```
119 :      asm(" fset I ");                    // 全体の割り込み許可
```

6.2 intTRB 関数

6.2.1 概要

init 関数でタイマ RB を 1ms ごとに割り込みが発生するように設定しました。intTRB 関数は、1ms ごとに実行される割り込みプログラムになります。

intTRB 関数内では、制御対象回路③基板の左側 7 セグメント LED、右側 7 セグメント LED、ステッピングモータ、DC モータの 4 種類を制御します。

LED はポートを"0"にすれば点灯、"1"にすれば消灯するので、intTRB 関数内では、特に制御しません。

今回のプログラムを、下記に示します。

```
#pragma interrupt intTRB( vect = 24 )
void intTRB( void )
{
    ①interrupt_1ms 関数の実行
    ②DC モータ制御プログラム
    ③ステッピングモータの制御プログラム
    ④7 セグメント LED　左と右の桁を交互に表示させるプログラム

}
```

「#pragma interrupt intTRB(vect=24)」は、『ベクタテーブル 24 番の割り込み関数は intTRB 関数です』、という意味です。24 番は、タイマ RB 割り込みが発生したときに実行される番号です。

タイマ RB は、1ms ごとに割り込みを発生させる設定にしています。また、#pragma interrupt 命令で、intTRB 関数が、タイマ RB 割り込み発生時に実行される関数に設定しているので、intTRB 関数が 1ms ごとに実行されることになります。

①interrupt_1ms 関数の実行

kadai1.c～kadai7.c のそれぞれのファイル内にある、interrupt_1ms 関数を実行します。1ms ごとに実行されます。

②DC モータ制御プログラム

DC モータを制御します。10ms 周期で、正転、逆転、停止を切り替えます。

例えば、2ms 正転、8ms 停止させると 20%で正転することになります。7ms 逆転、3ms 停止させると 70%で逆転することになります。

モータの制御は、p0_4 端子と p0_3 端子で行います。

③ステッピングモータの制御プログラム

ステッピングモータを 1ms ごとに制御します。今回のステッピングモータ「ST-42BYG0506H」は、ステップ角度 1.8 度で、1 周させるには、「360 度÷1.8＝200」回、励磁させれば良いことになります。

例えば、10ms ごとに励磁させると、10ms ごとに励磁×1 周 200 回＝2000ms＝2 秒で軸が一周します。

ステッピングモータの制御は、p2_3～p2_0 端子で行います。ただし、マイコンとステッピングモータの間には 74HC574 があり、p0_0 端子を"1"にした瞬間に p2_3～p2_0 端子の状態が、ステッピングモータに出力されます。ただし、p2_3～p2_0 端子は、7 セグメント LED と兼用の端子のため、ステッピングモータを制御するときは、7 セグメント LED を OFF にしておきます。

④7 セグメント LED　左と右の桁を交互に表示させるプログラム

7 セグメント LED を 1ms ごとに制御します。7 セグメント LED はダイナミック点灯で 1 桁ずつしか表示することができません。左 1ms 表示、右 1ms 表示を交互に繰り返して、人間の目には両方とも点灯しているように表示させます。

7 セグメント LED の制御は、p2_7～p2_0、p0_1、p0_2 端子で行います。ただし、ステッピングモータ制御端子と兼

用の端子があるので、ステッピングモータに情報を出力するときは、7 セグメント LED を左右とも、OFF にします。

①～④の詳しいプログラムについて、これから説明していきます。

6.2.2 制御手順

制御対象回路③は特に難しい回路ではありませんが、p2_7～p2_0 端子が次の回路を兼用しています。

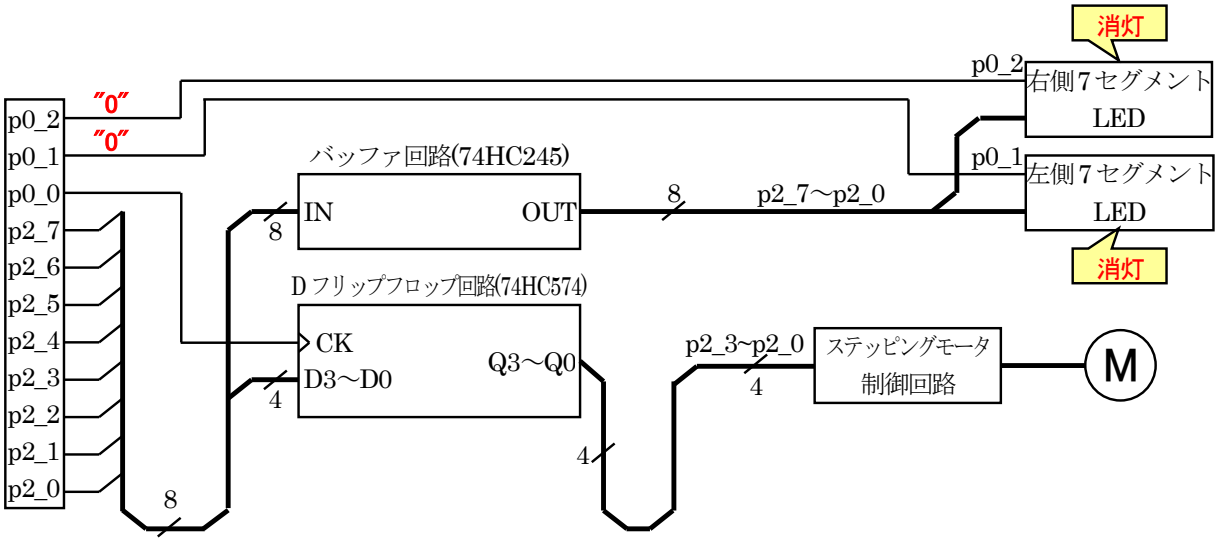
・ステッピングモータの制御
・左側 7 セグメント LED の制御
・右側 7 セグメント LED の制御

よって、これらを混同しないよう制御するのがポイントです。
プログラムでは、1ms ごとの割り込み内でこれらの制御を行います。

回路の簡略図、動作を、下図に示します。

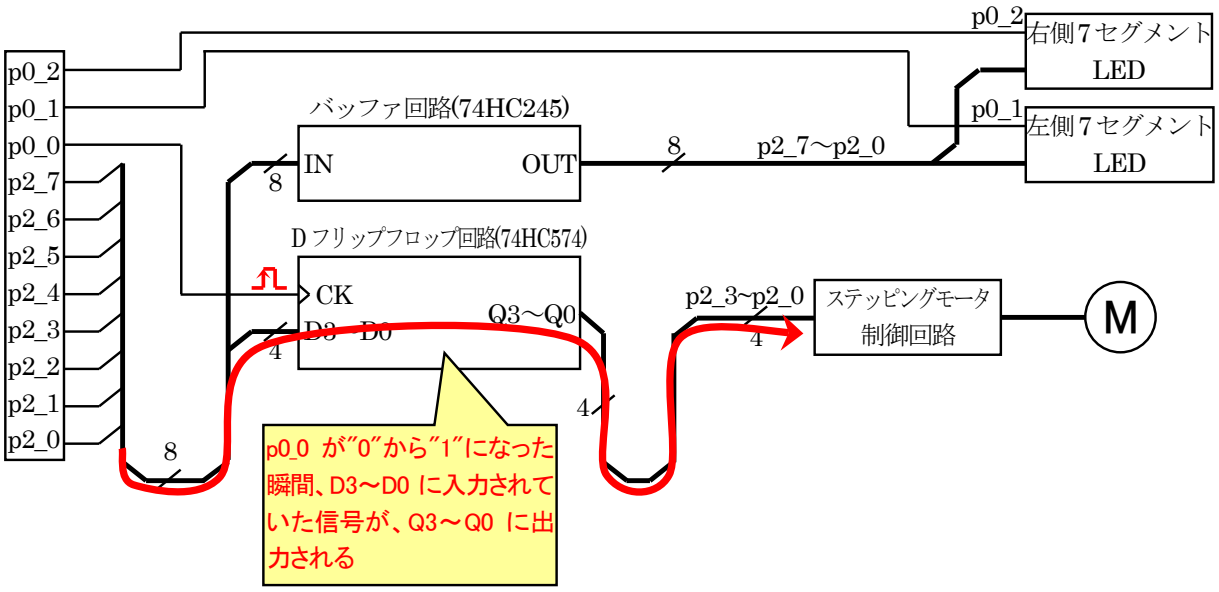
(1) 7 セグメント LED の消灯

まず、p0_2 と p0_1 を"0"にして、7 セグメント LED を消灯させます。



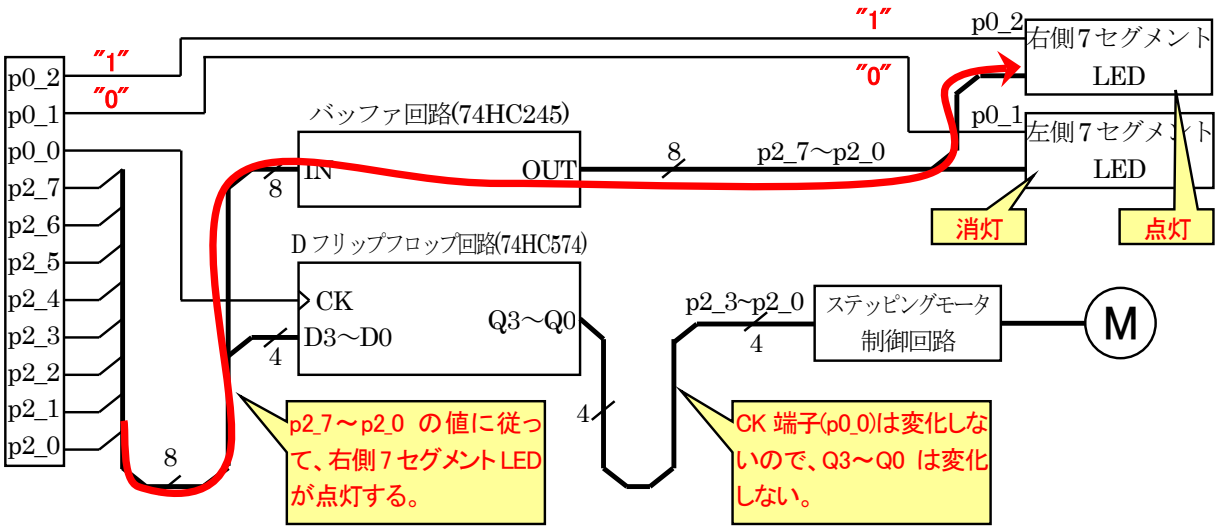
(2) ステッピングモータの制御

ステッピングモータを制御するために p2_3～p2_0 から、ステッピングモータへデータを出力します。まだ、ステッピングモータは前の動作のまま変化しません。p0_0 が"0"から"1"になった瞬間に 74HC574 の入力端子 D3～D0 の値が Q3～Q0 から出力されます。よって、p0_0 を"0"→"1"にした瞬間にステッピングモータの動作が変化します。次に p0_0 端子を"0"から"1"にするまで変化しません。



(3) 7セグメント LED の制御

左側 7 セグメント LED、右側 7 セグメント LED を制御します。p2_7～p2_0 が兼用のため、同時に点灯させることはできません。そのため、1ms ごとに左側 7 セグメント LED と右側 7 セグメント LED の点灯を交互に繰り返します。左と右の 7 セグメント LED は同時に点灯しませんが、1ms ごとに点灯しているので、人間の目で見ると同時に点灯しているように見えます。右側 7 セグメント LED を点灯させるときの信号を、下図に示します。



このように、「ステッピングモータ」→「左側 7 セグメント LED 表示」→「右側 7 セグメント LED 表示」の順番に制御していきます。

6.2.3 DC モータの制御

(1) 回転させる方法

制御対象回路③には、DC モータを制御する専用の IC が取り付けてあり、2bit の信号でモータを制御することができます。今回のスピード制御はマイコンのタイマを使わずに、割り込みプログラムで"1"と"0"にする時間を変えて行います。

ポートと DC モータの動作の関係を、下表に示します。

p0_4	p0_3	DC モータの動作
"0"	"0"	ストップ
"0"	"1"	時計回りに回転
"1"	"0"	反時計回りに回転
"1"	"1"	ブレーキ

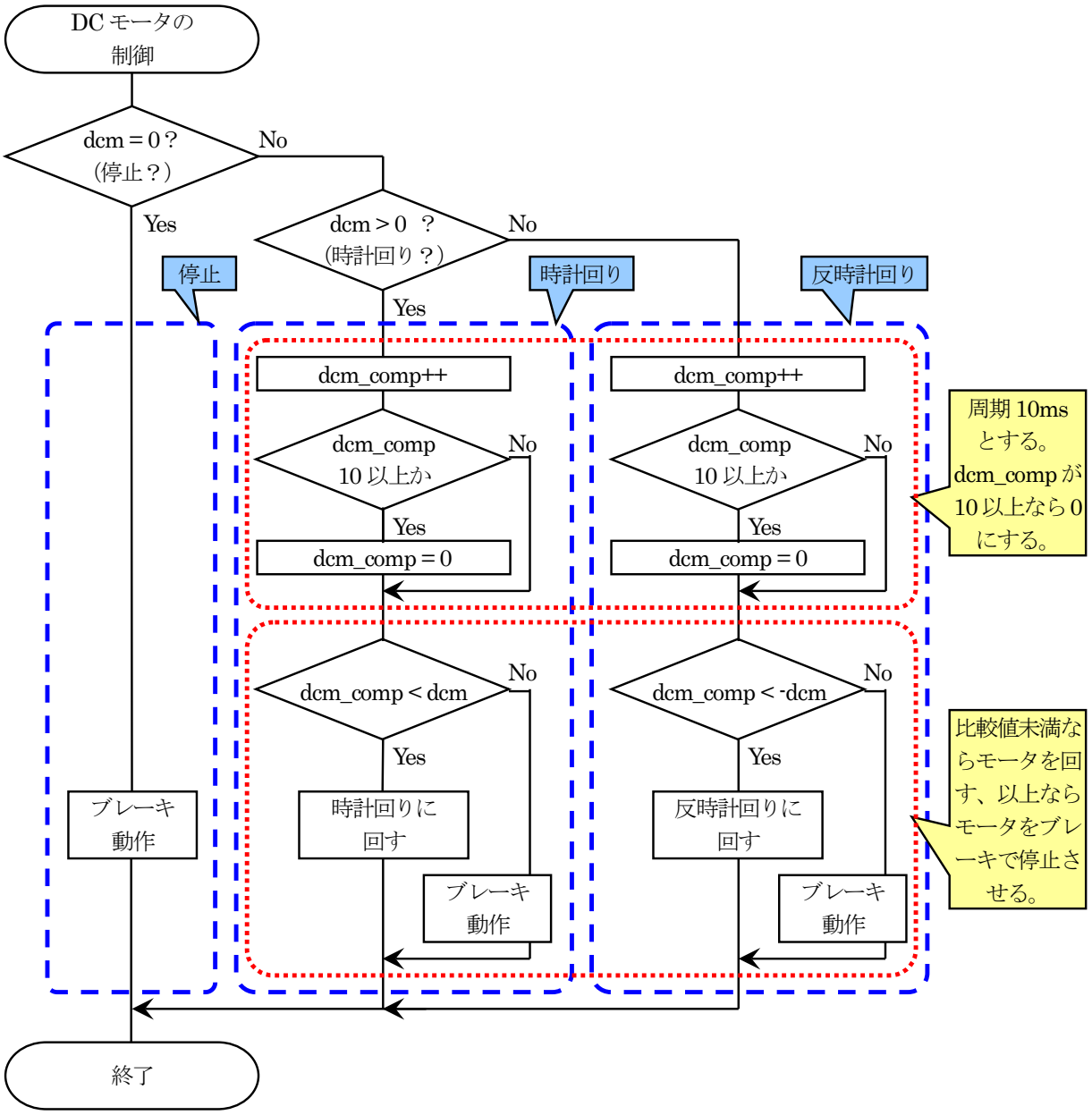
ストップは DC モータの端子間を解放、ブレーキは DC モータの端子間をショートさせます。今回モータを止めるのはストップを使います。

(2) 使用する変数

DC モータ関連で使用する変数を、下記に示します。

変数	内容
dcm	DC モータの回転方向と速度を設定します。 正の数(1～10)　：時計回りで回転(1 が 10%、10 が 100%) 負の数(-1～-10)：反時計回りで回転(-1 が-10%、-10 が-100%) 0 ：停止 例えば、 dcm = 5; とすると、50%で時計回りに回転します。
dcm_comp	DC モータを回転する間隔を比較する変数です。dcm_comp 変数を 1ms ごとに＋1して、dcm 変数と比較します。一致するとモータを停止させます。10 になると dcm_comp を 0 にして、モータを回転させます。 ※この変数は、main 関数では使用しません。

(3) フローチャート


DC モータの
制御
dcm = 0 ?
(停止 ?)
No
Yes
dcm > 0 ?
(時計回り ?)
No
Yes
停止
時計回り
反時計回り
dcm_comp++
dcm_comp
10 以上か
No
Yes
dcm_comp = 0
dcm_comp++
dcm_comp
10 以上か
No
Yes
dcm_comp = 0
周期 10ms
とする。
dcm_comp が
10 以上なら 0
にする。
dcm_comp < dcm
No
Yes
時計回りに
回す
ブレーキ
動作
dcm_comp < -dcm
No
Yes
反時計回りに
回す
ブレーキ
動作
比較値未満な
らモータを回
す、以上なら
モータをブレ
ーキで停止さ
せる。
ブレーキ
動作
終了

(4) プログラム

```
// グローバル変数の宣言

  65 :  ////////////////////////////////////////////////////////////
  66 :  // DC モータ
  67 :  int dcm;                                  // -10～10 0:停止 10:100%
  68 :  int dcm_comp;                             // 回転させるか比較用

// プログラム

 131 :      // DC モータ制御
 132 :      if( dcm == 0 ) {
 133 :          // ブレーキ
 134 :          p0_4 = 0;
 135 :          p0_3 = 0;
 136 :      } else if( dcm > 0 ) {
 137 :          // 正転
 138 :          dcm_comp++;
 139 :          if( dcm_comp >= 10 ) dcm_comp = 0;
 140 :          if( dcm_comp < dcm ) {
 141 :              p0_4 = 0;
 142 :              p0_3 = 1;
 143 :          } else {
 144 :              p0_4 = 0;
 145 :              p0_3 = 0;
 146 :          }
 147 :      } else {
 148 :          // 逆転
 149 :          dcm_comp++;
 150 :          if( dcm_comp >= 10 ) dcm_comp = 0;
 151 :          if( dcm_comp < (-dcm) ) {
 152 :              p0_4 = 1;
 153 :              p0_3 = 0;
 154 :          } else {
 155 :              p0_4 = 0;
 156 :              p0_3 = 0;
 157 :          }
 158 :      }
```

6.2.4 ステッピングモータの制御

(1) 励磁する手順

※励磁(れいじ)とは、ステッピングモータのコイルに電流を流すことです。

ステッピングモータには、A、$\overline{A}$、B、$\overline{B}$の 4 つのコイルがあります。1 相励磁で反時計回りに回転させるには「A → B → $\overline{A}$ → $\overline{B}$」の順番に励磁します。時計回りはその逆です。コイルとポートの関係を、下表に示します。

p2_0	p2_1	p2_2	p2_3	A のコイル	B のコイル	$\overline{A}$ のコイル	$\overline{B}$ のコイル
″1″	″0″	″0″	″0″	**ON**	OFF	OFF	OFF
″0″	**″1″**	″0″	″0″	OFF	**ON**	OFF	OFF
″0″	″0″	**″1″**	″0″	OFF	OFF	**ON**	OFF
″0″	″0″	″0″	**″1″**	OFF	OFF	OFF	**ON**

よって、ステッピングモータを反時計回りに回転させるには、ポート 2(p2_3～p2_0)を次のように設定します。

```
0x01→0x02→0x04→0x08→繰り返し
```

ステッピングモータを時計回りに回転させるには、逆に繰り返します。

(2) 出力データに配列を使う

プログラムでは stm_data という配列を作り、下記のようにプログラムしています。

```
70 :  ////////////////////////////////////////////////////////////
71 :  // ステッピングモータの励磁出力信号 1 相励磁
72 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
73 :  const signed char stm_data[] = {
74 :      0b00000001,                          // 0 A 励磁
75 :      0b00000010,                          // 1 B 励磁
76 :      0b00000100,                          // 2 /A 励磁
77 :      0b00001000,                          // 3 /B 励磁
78 :  };
```

添字([ ] の中に入れる数字のこと)に 0～3 の値をセットすると、0b00000001、0b00000010 ・・・ の値が返ってきます。例えば、添字に 1 をセットしたところを、下記に示します。

```
p2 = stm_data[ 1 ];     // ポート 2(p2)には、0b00000010 が設定される
```

したがって、添字を 0,1,2,3 と増やしていくとステッピングモータが反時計回りに回転して、逆に 3,2,1,0 と減らしていくとステッピングモータが時計回りに回転します。

(3) 回転数

ステッピングモータを回転させるスピードを変えるには、励磁コイルを切り替える間隔を変えます。切り替える間隔が短ければステッピングモータは速く回転し、間隔が長ければステッピングモータはゆっくりと回転します。
※ただし、切り替える時間が短すぎるとステッピングモータの動作が追従できなくなります。脱調（だっちょう）と言います。

今回のステッピングモータは励磁コイルを 1 つ切り替えるたびに、1.8 度回転します。1 回転(360 度)するために切り替える回数は次のようになります。

360÷1 回当たりの角度 1.8＝200

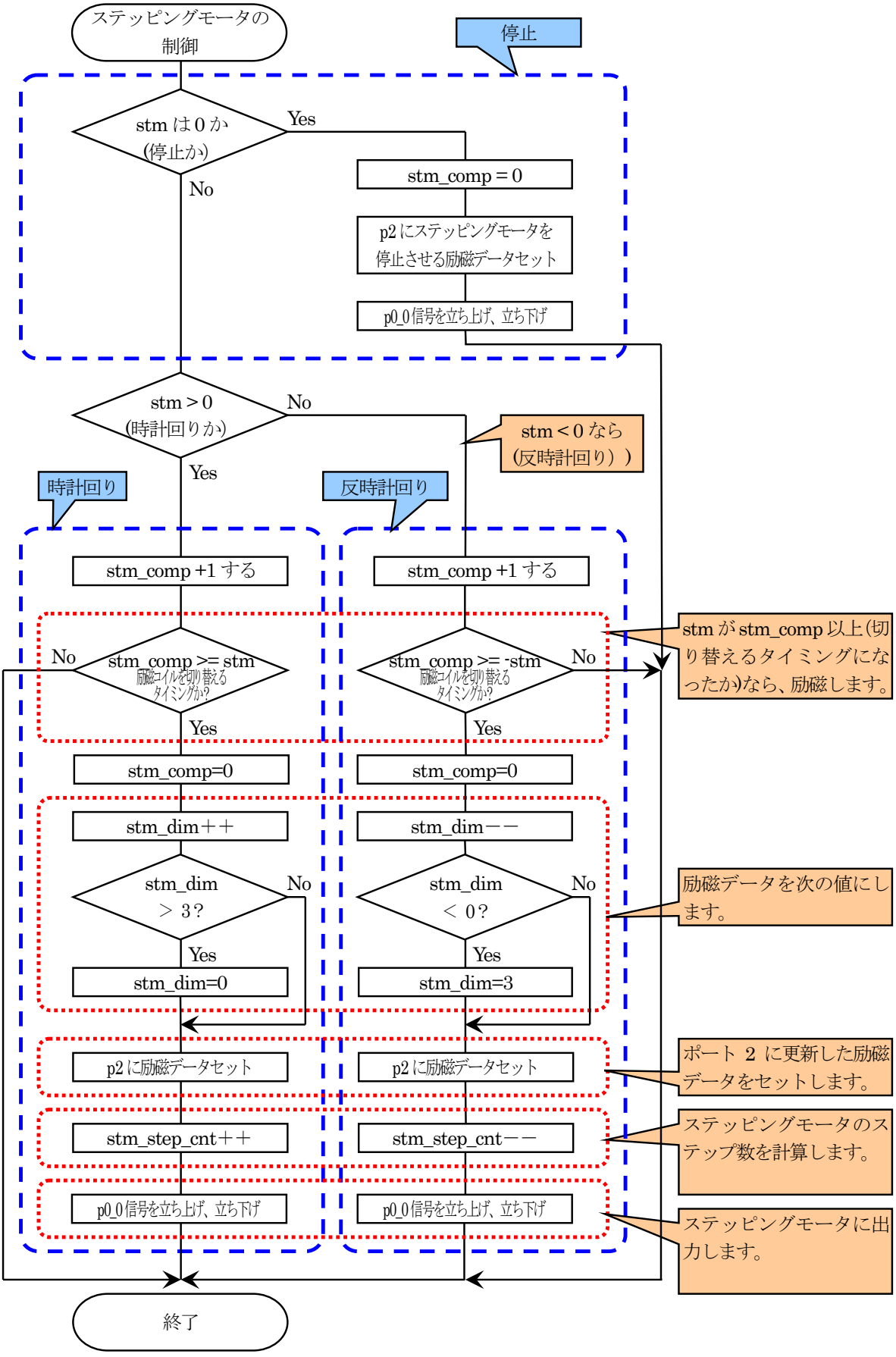
よって、200 回切り替えると、ステッピングモータは 1 回転します。
今回のプログラムは stm_step_cnt という変数を用意して、時計回りに 1 ステップ進んだならこの変数を＋1、反時計回りに 1 ステップ進んだならこの変数を－1して、ステッピングモータの移動ステップ数を検出します。
例えば、stm_step_cnt 変数が 300 なら、時計回りに 1.5 周したということが分かります。

(4) 使用する変数

ステッピングモータ関連で使用する変数を、下記に示します。

変数	内容
stm	ステッピングモータの回転方向と励磁する間隔を設定します。 正の数 ： 時計回りで値[ms]ごとに励磁 負の数 ： 反時計回りで値[ms]ごとに励磁 0 ： 停止 例えば、 stm = 5; とすると、5ms ごとに時計回りに励磁します。200 回で 1 周なので、1 周する時間は、 5ms×200 回＝1000ms＝1 秒 となります。
stm_step_cnt	ステッピングモータの励磁コイルを切り替えた回数（ステップ数）です。 時計回りに励磁コイルを切り替えた場合は＋1、反時計回りに励磁コイルを切り替えた場合は－1します。 1 周 200 回なので、+200 なら正転で 1 周、-200 なら逆転で 1 周したと分かります。
stm_comp	励磁コイルを切り替える間隔を比較する変数です。stm_comp 変数を、1ms ごとに＋1して、stm 変数と比較します。一致すると励磁します。 ※この変数は、main 関数では使用しません。
stm_dim	stm_data 配列の添字です。反時計回りの時は励磁コイルを切り替えるごとに＋1、時計回りの時は励磁コイルを切り替えるごとに－1します。stm_data 配列の添字は、0～3 の範囲なので、4 以上になったら 0 に、-1 以下になったら 3 にします。 ※この変数は、main 関数では使用しません。

(5) フローチャート


ステッピングモータの制御
停止
stm は 0 か (停止か)
Yes
No
stm_comp = 0
p2 にステッピングモータを停止させる励磁データセット
p0_0 信号を立ち上げ、立ち下げ
stm > 0 (時計回りか)
No
Yes
stm < 0 なら (反時計回り）)
時計回り
反時計回り
stm_comp +1 する
stm_comp +1 する
No
stm_comp >= stm
励磁コイルを切り替えるタイミングか?
stm_comp >= -stm
励磁コイルを切り替えるタイミングか?
No
Yes
Yes
stm_comp=0
stm_comp=0
stm_dim＋＋
stm_dim－－
stm_dim ＞ 3?
No
stm_dim ＜ 0?
No
Yes
Yes
stm_dim=0
stm_dim=3
p2 に励磁データセット
p2 に励磁データセット
stm_step_cnt＋＋
stm_step_cnt－－
p0_0 信号を立ち上げ、立ち下げ
p0_0 信号を立ち上げ、立ち下げ
終了
stm が stm_comp 以上(切り替えるタイミングになったか)なら、励磁します。
励磁データを次の値にします。
ポート 2 に更新した励磁データをセットします。
ステッピングモータのステップ数を計算します。
ステッピングモータに出力します。

(6) プログラム

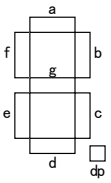
```
// グローバル変数の宣言

  70 :  ////////////////////////////////////////////////////////////
  71 :  // ステッピングモータの励磁出力信号 1 相励磁
  72 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
  73 :  const signed char stm_data[] = {
  74 :      0b00000001,                          // 0 A 励磁
  75 :      0b00000010,                          // 1 B 励磁
  76 :      0b00000100,                          // 2 /A 励磁
  77 :      0b00001000,                          // 3 /B 励磁
  78 :  };
  79 :
  80 :  int stm;                                 // 0:停止 正:時計回りで値[ms]ごとに励磁
  81 :                                           // 負:反時計回りで値[ms]ごとに励磁 ※1 は脱調する
  82 :  int stm_step_cnt;                        // 動作したステップ数 200 で 1 回転
  83 :  int stm_comp;                            // 励磁コイル切り替える間隔
  84 :  int stm_dim;                             // 励磁出力信号の添字

// プログラム

 164 :      // ステッピングモータの制御
 165 :      if( stm == 0 ) {
 166 :          // 停止、何もせず
 167 :          stm_comp = 0;
 168 :          p2 = 0x00;
 169 :          // ステッピングモータへ出力
 170 :          p0_0 = 1;                        // CK HIGH
 171 :          p0_0 = 0;                        // CK LOW
 172 :      } else if( stm > 0 ) {
 173 :          // 時計回り
 174 :          stm_comp++;
 175 :          if( stm_comp >= stm ) {          //励磁ｺｲﾙを替える間隔の確認
 176 :              stm_comp = 0;
 177 :              stm_dim++;                   // 励磁コイルの切り替え
 178 :              if( stm_dim > 3 ) stm_dim = 0;
 179 :              p2 = stm_data[ stm_dim ];    // 励磁データセット
 180 :              stm_step_cnt++;              // ステップ数を増やす
 181 :              // ステッピングモータへ出力
 182 :              p0_0 = 1;                    // CK HIGH
 183 :              p0_0 = 0;                    // CK LOW
 184 :          }
 185 :      } else {
 186 :          // 反時計回り
 187 :          stm_comp++;
 188 :          if( stm_comp >= (-stm) ) {       //励磁ｺｲﾙを替える間隔の確認
 189 :              stm_comp = 0;
 190 :              stm_dim--;                   // 励磁コイルの切り替え
 191 :              if( stm_dim < 0 ) stm_dim = 3;
 192 :              p2 = stm_data[ stm_dim ];    // 励磁データセット
 193 :              stm_step_cnt--;              // ステップ数を減らす
 194 :              // ステッピングモータへ出力
 195 :              p0_0 = 1;                    // CK HIGH
 196 :              p0_0 = 0;                    // CK LOW
 197 :          }
 198 :      }
```

6.2.5 7 セグメント LED の制御

(1) 表示する方法

7 セグメント LED には、名前のとおり 7 個の LED があります。点灯する LED を組み合わせることにより数値や一部のアルファベットを表示させることができます。各 LED には名前が付いており、いちばん上を"a"、時計回りに"b"→"c"…"f"、真ん中を"g"とします。小数点を表示する点"dp"もありますが、今回は表示しません（回路的に繋がっていません）。表示する値と、7 セグメント LED に送るデータを下図に示します。

この表示パターンは課題のプリントで指定されています。指定パターン以外で表示すると、減点の対象となりますので気をつけてください（例えば、7 は"f"を付ける場合と付けない場合があります）。

内容	表示 イメージ 	dp p2_7	g p2_6	f p2_5	e p2_4	d p2_3	c p2_2	b p2_1	a p2_0
0		0	0	1	1	1	1	1	1
1		0	0	0	0	0	1	1	0
2		0	1	0	1	1	0	1	1
3		0	1	0	0	1	1	1	1
4		0	1	1	0	0	1	1	0
5		0	1	1	0	1	1	0	1
6		0	1	1	1	1	1	0	1
7		0	0	1	0	0	1	1	1
8		0	1	1	1	1	1	1	1
9		0	1	1	0	1	1	1	1
A		0	1	1	1	0	1	1	1
B		0	1	1	1	1	1	0	0
C		0	0	1	1	1	0	0	1
D		0	1	0	1	1	1	1	0
E		0	1	1	1	1	0	0	1

F		0	1	1	1	0	0	0	1
OFF		0	1	0	1	0	1	0	0
ON		0	0	0	0	1	0	0	0
-		0	1	0	0	0	0	0	0
dp		1	0	0	0	0	0	0	0

上表のデータをポート 2 から出力し、p0_1 を"1"にすると左側 7 セグメント LED が点灯、p0_2 を"1"にすると右側 7 セグメント LED が点灯します。

ポート 2 は、左側 7 セグメント LED、右側 7 セグメント LED を共用しているので、交互に点灯させます。具体的には 1ms 間は左側 7 セグメント LED を点灯、次の 1ms 間は右側 7 セグメント LED を点灯、これを交互に繰り返します。1ms ごとなので、人間の目には早すぎて同時に点灯しているように見えます。もし 1 秒ごとに交互に点灯させたなら、遅すぎて交互に点灯しているのが分かってしまいます。どれくらいのスピードまで同時に点灯して見えるのか実験するのも良いでしょう。

(2) 表示データに配列を使う

プログラムでは seg_data という配列を作り、下記のようにプログラムしています。

```
 38 :  const unsigned char seg_data[] = {      // 7 セグメント LED の表示データ
 39 :      0b00111111,                         //  0 = 0
 40 :      0b00000110,                         //  1 = 1
 41 :      0b01011011,                         //  2 = 2
 42 :      0b01001111,                         //  3 = 3
 43 :      0b01100110,                         //  4 = 4
 44 :      0b01101101,                         //  5 = 5
 45 :      0b01111101,                         //  6 = 6
 46 :      0b00100111,                         //  7 = 7
 47 :      0b01111111,                         //  8 = 8
 48 :      0b01101111,                         //  9 = 9
 49 :      0b01110111,                         // 10 = A
 50 :      0b01111100,                         // 11 = b
 51 :      0b00111001,                         // 12 = C
 52 :      0b01011110,                         // 13 = d
 53 :      0b01111001,                         // 14 = E
 54 :      0b01110001,                         // 15 = F
 55 :      0b00000000,                         // 16 = 消灯
 56 :      0b01000000,                         // 17 = -(マイナス)
 57 :      0b01010100,                         // 18 = OFF(⌐)
 58 :      0b00001000,                         // 19 = ON (__)
 59 :  };
```

添字([ ] の中に入れる数字のこと)に 0～19 の値をセットすると、7 セグメント LED に出力する値が返ってきます。値は、0～9 が数字、10～15 が"A"～"F"のアルファベット、16 が消灯、17 が"－"、18 が"⌐"(OFF)、19 が"__"(ON)となります。例えば、添字に 10 をセットしたところを、下記に示します。

```
    p2 = seg_data[ 10 ];     // ポート 2(p2)には、0b01110111 が設定されます。
```

このように、添字にそのまま 0～19 の値をセットすればポート 2 に出力する値が取り出せるので、いちいち 0 だから 0b00111111 をセットして･･･ 1 だから 0b00000110 をセットして･･･ というようにプログラムする必要が無くなります。

(3) 使用する変数

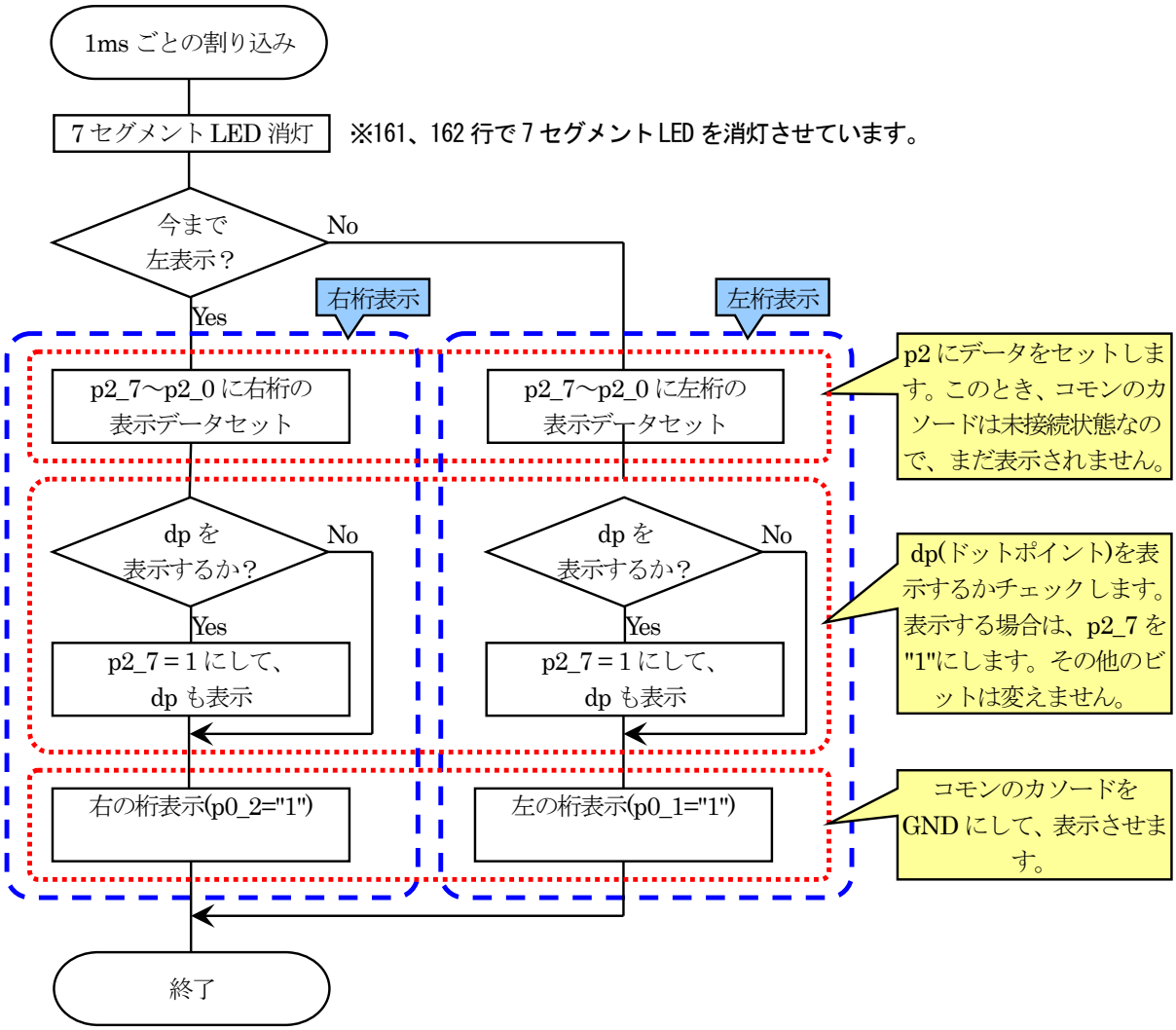
7 セグメント LED 関連で使用する変数を、下記に示します。

変数	内容

seg_left	左側 7 セグメント LED に表示する値を設定します。
	0～9 が数字、10～15 が"A"～"F"のアルファベット、16 が消灯、17～19 がその他です。
	17 以降は、define 文で下記のように定義しています。
	
	22 ： // seg_left と seg_right 変数に入れる値
	23 ： #define SEG_NULL 16 // 7 セグ 消灯
	24 ： #define SEG_MAI 17 // 7 セグ マイナス表示
	25 ： #define SEG_OFF 18 // 7 セグ OFF 表示
	26 ： #define SEG_ON 19 // 7 セグ ON 表示
	27 ： #define SEG_DP 0x80 // 7 セグ DP 表示
	
	使用例を下記に示します。
	
	seg_left = 0; // '0'を表示
	seg_left = 0xb; // 'b'を表示
	seg_left = SEG_OFF // OFF を表示
	seg_left = SEG_NULL; // 消灯

seg_right	右側 7 セグメント LED に表示する値を設定します。表示方法は、seg_left 変数と同様です。
seg_digit	左側、右側のどちらの 7 セグメント LED を表示するか設定します。main 関数内では使いません。 0:左表示　1:右表示

(4) フローチャート


1ms ごとの割り込み
7 セグメント LED 消灯
※161、162 行で 7 セグメント LED を消灯させています。
今まで
左表示？
No
Yes
右桁表示
左桁表示
p2_7～p2_0 に右桁の
表示データセット
p2_7～p2_0 に左桁の
表示データセット
p2 にデータをセットします。このとき、コモンのカソードは未接続状態なので、まだ表示されません。
dp を
表示するか？
No
Yes
p2_7＝1 にして、
dp も表示
dp を
表示するか？
No
Yes
p2_7＝1 にして、
dp も表示
dp(ドットポイント)を表示するかチェックします。表示する場合は、p2_7 を"1"にします。その他のビットは変えません。
右の桁表示(p0_2="1")
左の桁表示(p0_1="1")
コモンのカソードを GND にして、表示させます。
終了

(5) プログラム

```
// シンボル定義
  22 :  // seg_left と seg_right 変数に入れる値
  23 :  #define SEG_NULL            16          // 7セグ 消灯
  24 :  #define SEG_MAI             17          // 7セグ マイナス表示
  25 :  #define SEG_OFF             18          // 7セグ OFF 表示
  26 :  #define SEG_ON              19          // 7セグ ON 表示
  27 :  #define SEG_DP              0x80        // 7セグ DP 表示

// グローバル変数の宣言

  35 :  ////////////////////////////////////////////////////////////
  36 :  // 7セグメント LED
  37 :
  38 :  const unsigned char seg_data[] = {      // 7セグメント LED の表示データ
  39 :      0b00111111,                         //  0 = 0
  40 :      0b00000110,                         //  1 = 1
  41 :      0b01011011,                         //  2 = 2
  42 :      0b01001111,                         //  3 = 3
  43 :      0b01100110,                         //  4 = 4
  44 :      0b01101101,                         //  5 = 5
  45 :      0b01111101,                         //  6 = 6
  46 :      0b00100111,                         //  7 = 7
  47 :      0b01111111,                         //  8 = 8
  48 :      0b01101111,                         //  9 = 9
  49 :      0b01110111,                         // 10 = A
  50 :      0b01111100,                         // 11 = b
  51 :      0b00111001,                         // 12 = C
  52 :      0b01011110,                         // 13 = d
  53 :      0b01111001,                         // 14 = E
  54 :      0b01110001,                         // 15 = F
  55 :      0b00000000,                         // 16 = 消灯
  56 :      0b01000000,                         // 17 = -(マイナス)
  57 :      0b01010100,                         // 18 = OFF(⊓)
  58 :      0b00001000,                         // 19 = ON (‗)
  59 :  };
  60 :
  61 :  int seg_left  = SEG_NULL;               // 左の桁  表示値
  62 :  int seg_right = SEG_NULL;               // 右の桁  表示値
  63 :  int seg_digit;                          // 0:左表示 1:右表示

// プログラム

 160 :      // ステッピングモータを制御するために、共通線の7セグは消灯させる
 161 :      p0_2 = 0;                           // 7セグ右消灯
 162 :      p0_1 = 0;                           // 7セグ左消灯

 200 :      // 7セグメント LED の表示  左と右の桁を交互に表示
 201 :      if( seg_digit == 0 ) {              // 左表示中なら
 202 :          // 右を表示
 203 :          seg_digit = 1;
 204 :          p2 = seg_data[ seg_right & 0x7f ];          // 値のセット
 205 :          if( seg_right & SEG_DP ) p2 |= SEG_DP;      // DP の表示
 206 :          p0_2 = 1;                                   // 右 ON
 207 :      } else {
 208 :          // 左を表示
 209 :          seg_digit = 0;
 210 :          p2 = seg_data[ seg_left & 0x7f ];           // 値のセット
 211 :          if( seg_left  & SEG_DP ) p2 |= SEG_DP;      // DP の表示
 212 :          p0_1 = 1;                                   // 左 ON
 213 :      }
```

6.3「common.c」ファイルーまとめ

今まで説明した R8C/38A マイコンの内蔵周辺機能の初期化、DC モータ、ステッピングモータ、7 セグメント LED の制御プログラムなどを「common.c」としてまとめます。「kadai1.c」～「kadai7.c」プログラムの共通ファイルとしました。下記に、プログラムを示します。

コンテスト時は限られた時間しかありませんが、できるだけコメントをつけてください。見直すときに自分でも見やすいですし、間違いも少なくなります（審査もしやすくなります）。

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　共通ファイル
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////
12 : #include "sfr_r838a.h"                  // R8C/38A SFR の定義ファイル
13 :
14 : ////////////////////////////////////////////////////////////
15 : // シンボル定義
16 : ////////////////////////////////////////////////////////////
17 : // 入力機器のポート
18 : #define         tcsw        (p9_2)      // タクトスイッチの端子
19 : #define         tgsw        (p9_1)      // トグルスイッチの端子
20 : #define         phin        (p9_0)      // フォトインタラプタの端子
21 :
22 : // seg_left と seg_right 変数に入れる値
23 : #define SEG_NULL            16          // 7 セグ 消灯
24 : #define SEG_MAI             17          // 7 セグ マイナス表示
25 : #define SEG_OFF             18          // 7 セグ OFF 表示
26 : #define SEG_ON              19          // 7 セグ ON 表示
27 : #define SEG_DP              0x80        // 7 セグ DP 表示
28 :
29 : ////////////////////////////////////////////////////////////
30 : // プロトタイプ宣言
31 : ////////////////////////////////////////////////////////////
32 : void init( void );
33 : void interrupt_1ms( void );
34 :
35 : ////////////////////////////////////////////////////////////
36 : // 7 セグメント LED
37 :
38 : const unsigned char seg_data[] = {      // 7 セグメント LED の表示データ
39 :     0b00111111,                         //  0 = 0
40 :     0b00000110,                         //  1 = 1
41 :     0b01011011,                         //  2 = 2
42 :     0b01001111,                         //  3 = 3
43 :     0b01100110,                         //  4 = 4
44 :     0b01101101,                         //  5 = 5
45 :     0b01111101,                         //  6 = 6
46 :     0b00100111,                         //  7 = 7
47 :     0b01111111,                         //  8 = 8
48 :     0b01101111,                         //  9 = 9
49 :     0b01110111,                         // 10 = A
50 :     0b01111100,                         // 11 = b
51 :     0b00111001,                         // 12 = C
52 :     0b01011110,                         // 13 = d
53 :     0b01111001,                         // 14 = E
54 :     0b01110001,                         // 15 = F
55 :     0b00000000,                         // 16 = 消灯
56 :     0b01000000,                         // 17 = -(マイナス)
57 :     0b01010100,                         // 18 = OFF(⌐)
58 :     0b00001000,                         // 19 = ON (_)
59 : };
```

```
 60 :
 61 :  int seg_left  = SEG_NULL;              // 左の桁  表示値
 62 :  int seg_right = SEG_NULL;              // 右の桁  表示値
 63 :  int seg_digit;                         // 0:左表示 1:右表示
 64 :
 65 :  ////////////////////////////////////////////////////////////
 66 :  // DC モータ
 67 :  int dcm;                               // -10～10 0:停止 10:100%
 68 :  int dcm_comp;                          // 回転させるか比較用
 69 :
 70 :  ////////////////////////////////////////////////////////////
 71 :  // ステッピングモータの励磁出力信号 1 相励磁
 72 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
 73 :  const signed char stm_data[] = {
 74 :      0b00000001,                         // 0 A 励磁
 75 :      0b00000010,                         // 1 B 励磁
 76 :      0b00000100,                         // 2 /A 励磁
 77 :      0b00001000,                         // 3 /B 励磁
 78 :  };
 79 :
 80 :  int stm;                                // 0:停止 正:時計回りで値[ms]ごとに励磁
 81 :                                          // 負:反時計回りで値[ms]ごとに励磁 ※1 は脱調する
 82 :  int stm_step_cnt;                       // 動作したステップ数 200 で 1 回転
 83 :  int stm_comp;                           // 励磁コイル切り替える間隔
 84 :  int stm_dim;                            // 励磁出力信号の添字
 85 :
 86 :  ////////////////////////////////////////////////////////////////////////
 87 :  // R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
 88 :  ////////////////////////////////////////////////////////////////////////
 89 :  void init( void )
 90 :  {
 91 :      int i;
 92 :
 93 :      // クロックを XIN クロック(20MHz)に変更
 94 :      prc0  = 1;                          // プロテクト解除
 95 :      cm13  = 1;                          // P4_6,P4_7 を XIN-XOUT 端子にする
 96 :      cm05  = 0;                          // XIN クロック発振
 97 :      for(i=0; i<50; i++ );               // 安定するまで少し待つ(約 10ms)
 98 :      ocd2  = 0;                          // システムクロックを XIN にする
 99 :      prc0  = 0;                          // プロテクト ON
100 :
101 :      // ポートの初期出力
102 :      p0 = 0xe0;                          // LED 消灯、DC モータ停止 7 セグ消灯
103 :
104 :      // ポートの入出力設定
105 :      prc2 = 1;                           // PD0 のプロテクト解除
106 :      pd0 = 0xff;                         // 出力に設定
107 :      pd2 = 0xff;                         // 出力に設定
108 :
109 :      // タイマ RB の設定
110 :      // 割り込み周期 = 1 / 20[MHz]    * (TRBPRE+1) * (TRBPR+1)
111 :      //              = 1 / (20*10^-6) * 200        * 100
112 :      //              = 0.001[s] = 1[ms]
113 :      trbmr  = 0x00;                      // 動作モード、分周比設定
114 :      trbpre = 200-1;                     // プリスケーラレジスタ
115 :      trbpr  = 100-1;                     // プライマリレジスタ
116 :      trbic  = 0x07;                      // 割り込み優先レベル設定
117 :      trbcr  = 0x01;                      // カウント開始
118 :
119 :      asm(" fset I ");                    // 全体の割り込み許可
120 :  }
121 :
```

```
122 :  //****************************************************************************
123 :  // タイマ RB 割り込み処理(1ms ごとに実行)
124 :  //****************************************************************************
125 :  #pragma interrupt intTRB(vect=24)
126 :  void intTRB( void )
127 :  {
128 :      // 1ms ごとに実行する関数
129 :      interrupt_1ms();
130 :
131 :      // DC モータ制御
132 :      if( dcm == 0 ) {
133 :          // ブレーキ
134 :          p0_4 = 0;
135 :          p0_3 = 0;
136 :      } else if( dcm > 0 ) {
137 :          // 正転
138 :          dcm_comp++;
139 :          if( dcm_comp >= 10 ) dcm_comp = 0;
140 :          if( dcm_comp < dcm ) {
141 :              p0_4 = 0;
142 :              p0_3 = 1;
143 :          } else {
144 :              p0_4 = 0;
145 :              p0_3 = 0;
146 :          }
147 :      } else {
148 :          // 逆転
149 :          dcm_comp++;
150 :          if( dcm_comp >= 10 ) dcm_comp = 0;
151 :          if( dcm_comp < (-dcm) ) {
152 :              p0_4 = 1;
153 :              p0_3 = 0;
154 :          } else {
155 :              p0_4 = 0;
156 :              p0_3 = 0;
157 :          }
158 :      }
159 :
160 :      // ステッピングモータを制御するために、共通線の 7 セグは消灯させる
161 :      p0_2 = 0;                           // 7 セグ右消灯
162 :      p0_1 = 0;                           // 7 セグ左消灯
163 :
164 :      // ステッピングモータの制御
165 :      if( stm == 0 ) {
166 :          // 停止、何もせず
167 :          stm_comp = 0;
168 :          p2 = 0x00;
169 :          // ステッピングモータへ出力
170 :          p0_0 = 1;                       // CK HIGH
171 :          p0_0 = 0;                       // CK LOW
172 :      } else if( stm > 0 ) {
173 :          // 時計回り
174 :          stm_comp++;
175 :          if( stm_comp >= stm ) {         //励磁ｺｲﾙを替える間隔の確認
176 :              stm_comp = 0;
177 :              stm_dim++;                  // 励磁コイルの切り替え
178 :              if( stm_dim > 3 ) stm_dim = 0;
179 :              p2 = stm_data[ stm_dim ];   // 励磁データセット
180 :              stm_step_cnt++;             // ステップ数を増やす
181 :              // ステッピングモータへ出力
182 :              p0_0 = 1;                   // CK HIGH
183 :              p0_0 = 0;                   // CK LOW
184 :          }
```

```
185 :      } else {
186 :          // 反時計回り
187 :          stm_comp++;
188 :          if( stm_comp >= (-stm) ) {      //励磁ｺｲﾙを替える間隔の確認
189 :              stm_comp = 0;
190 :              stm_dim--;                  // 励磁コイルの切り替え
191 :              if( stm_dim < 0 ) stm_dim = 3;
192 :              p2 = stm_data[ stm_dim ];   // 励磁データセット
193 :              stm_step_cnt--;             // ステップ数を減らす
194 :              // ステッピングモータへ出力
195 :              p0_0 = 1;                   // CK HIGH
196 :              p0_0 = 0;                   // CK LOW
197 :          }
198 :      }
199 :
200 :      // 7セグメントLEDの表示　左と右の桁を交互に表示
201 :      if( seg_digit == 0 ) {              // 左表示中なら
202 :          // 右を表示
203 :          seg_digit = 1;
204 :          p2 = seg_data[ seg_right & 0x7f ];          // 値のセット
205 :          if( seg_right & SEG_DP ) p2 |= SEG_DP;      // DPの表示
206 :          p0_2 = 1;                                   // 右ON
207 :      } else {
208 :          // 左を表示
209 :          seg_digit = 0;
210 :          p2 = seg_data[ seg_left & 0x7f ];           // 値のセット
211 :          if( seg_left  & SEG_DP ) p2 |= SEG_DP;      // DPの表示
212 :          p0_1 = 1;                                   // 左ON
213 :      }
214 :  }
215 :
216 :  ////////////////////////////////////////////////////////////////////////
217 :  // End of File
218 :  ////////////////////////////////////////////////////////////////////////
```

7. 課題1

7.1 課題

下記のプログラムを作成してください。

(1) TCSW ON、OFF で、下記のように 7 セグメント LED を表示する。

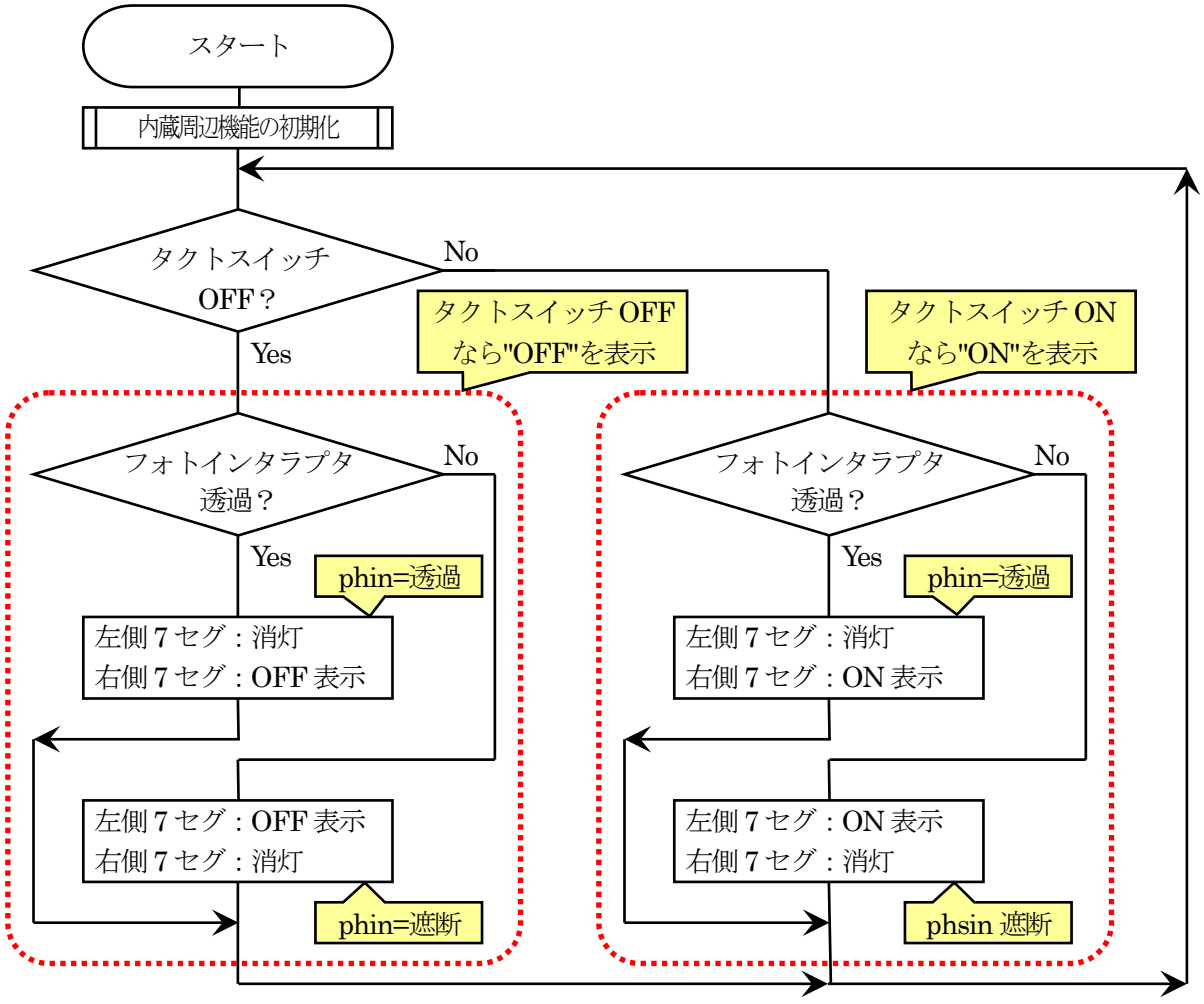
TCSW OFF の表示　　TCSW ON の表示

(2) 表示する 7 セグメント LED の位置は、PHIN の状態で次のように決まる。

(A) PHIN が透過で、右側の 7 セグメント LED に表示する。

(B) PHIN が遮断で、左側の 7 セグメント LED に表示する。

7.2 フローチャート



7.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題１
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
14 : ////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                            // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                        // メインは無限ループ、これ以降は
22 :                                        // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     if( tcsw == 0 ) {
31 :         // タクトスイッチ OFF なら
32 :         if( phin == 0 ) {              // フォトインタラプタが透過なら
33 :             seg_left  = SEG_NULL;
34 :             seg_right = SEG_OFF;
35 :         } else {                       // フォトインタラプタが遮断なら
36 :             seg_left  = SEG_OFF;
37 :             seg_right = SEG_NULL;
38 :         }
39 :     } else {
40 :         // タクトスイッチ ON なら
41 :         if( phin == 0 ) {              // フォトインタラプタが透過なら
42 :             seg_left  = SEG_NULL;
43 :             seg_right = SEG_ON;
44 :         } else {                       // フォトインタラプタが遮断なら
45 :             seg_left  = SEG_ON;
46 :             seg_right = SEG_NULL;
47 :         }
48 :     }
49 : }
50 :
51 : ////////////////////////////////////////////////////////////////////////
52 : // end of file
53 : ////////////////////////////////////////////////////////////////////////
```

7.4 プログラムの解説

行	詳細
12	通常は、common.c と kadai1.c をそれぞれルネサス統合開発環境に登録し、kadai1.c からはヘッダファイル（common.h ファイル）をインクルードします。この場合、common.c と common.h を作りプログラムする必要がありますが、その分プログラム作成に時間がかかってしまいます。 今回は、kadai1.c ファイルから common.c ファイルをインクルードして（取り込んで）、kadai1.c のプログラムの一部とします。コンテストでは限られた時間しかありませんので、手続きを簡略化して、時間短縮します。kadai1.c～kadai7.c で、common.c ファイルを取り込みます。
30	タクトスイッチが OFF かどうかチェックします。OFF なら 31～38 行、ON なら 40～47 行を実行します。
31～38	タクトスイッチが OFF のときの処理です。 フォトインタラプタが透過なら、左側 7 セグメント LED を消灯、右側 7 セグメント LED に OFF を表示します。 フォトインタラプタが遮断なら、左側 7 セグメント LED に OFF を表示、右側 7 セグメント LED を消灯します。
40～47	タクトスイッチが ON のときの処理です。 フォトインタラプタが透過なら、左側 7 セグメント LED を消灯、右側 7 セグメント LED に ON を表示します。 フォトインタラプタが遮断なら、左側 7 セグメント LED に ON を表示、右側 7 セグメント LED を消灯します。

■プログラム作成についてポイント

プログラムは課題どおりに正しく動作することが重要ですが、そのほかにも構造はいいか？書式はあっているか？読みやすいか？ということも評価されます。特に最後の読みやすさについては重要です。作ったばかりの自分はコメントがなくてもわかりますが、将来の自分のため、プログラムを読む他の人のために読みやすいコメントを書くことは大切なことです。面倒な気もしますが、少し大きなプログラムになってプロジェクトなど多くの人がかかわる場合にはわかりやすいコメントは必須です。

8. 課題 2

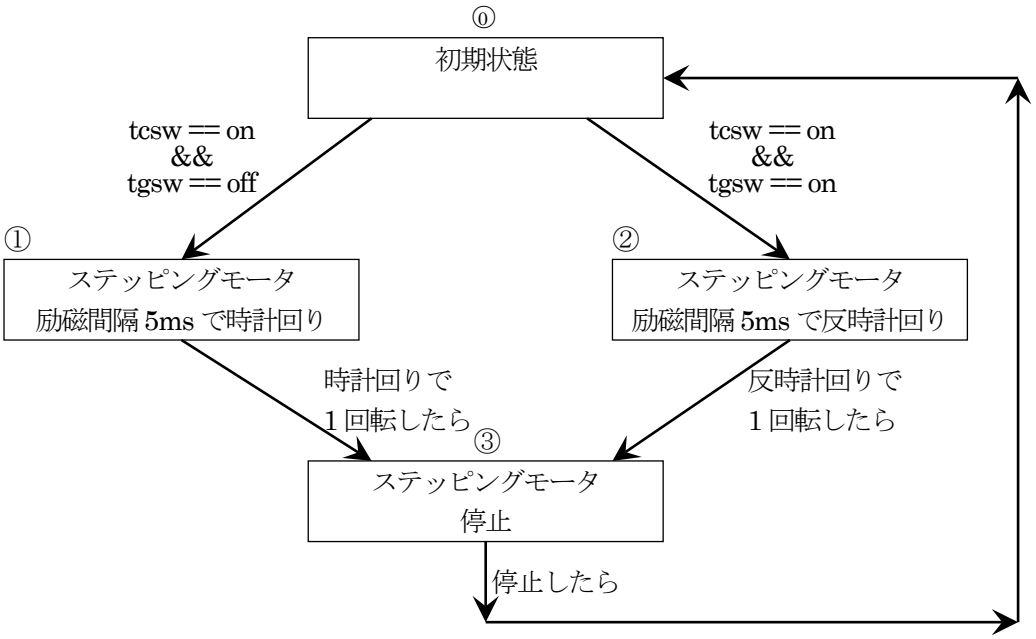
8.1 課題

下記のプログラムを作成してください。

(1) プログラムを開始した後、ステッピングモータの指示針を手で動かし、0 の位置に合わせる。
(2) タクトスイッチを ON すると、ステッピングモータが回転し、1回転して停止する。
(3) 回転方向は、タクトスイッチを ON にしたときの、トグルスイッチの状態により決まる。
　(A) トグルスイッチが OFF のとき、時計回りに回転する。
　(B) トグルスイッチが ON のとき、反時計回りに回転する。
(4) ステッピングモータが1回転し停止した後、(2)以降の動作が再度行えることとする。

8.2 状態遷移図

文章だけでは、なかなか分かりづらいので、複雑な処理の場合は簡単な状態遷移図を書いて、プログラムの構造を確認しましょう。



プログラムは、□ごと分けます。プログラムは switch-case 文を使い、case の数字と○数字の番号に合わせることにします。

8.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題２
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
14 : ////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                            // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                        // メインは無限ループ、これ以降は
22 :                                        // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int mode = 0;               // 状態(必ず static にすること!)
31 :
32 :     switch( mode ) {
33 :         case 0:
34 :             // タクトスイッチとトグルスイッチの確認
35 :             if( tcsw == 1 && tgsw == 0 ) {
36 :                 mode = 1;
37 :                 break;
38 :             } else if( tcsw == 1 && tgsw == 1 ) {
39 :                 mode = 2;
40 :                 break;
41 :             }
42 :             break;
43 :
44 :         case 1:
45 :             // 時計回りで 1 回転するまで待つ(1 回転 200 ステップ)
46 :             stm = 5;                   // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ正転で 5ms ごとに励磁
47 :             if( stm_step_cnt >= 200 ) { // 1 周したか？
48 :                 mode = 3;
49 :                 break;
50 :             }
51 :             break;
52 :
53 :         case 2:
54 :             // 反時計周りで 1 回転するまで待つ(1 回転 200 ステップ)
55 :             stm = -5;                  // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ逆転で 5ms ごとに励磁
56 :             if( stm_step_cnt <= -200 ) {// 反時計回りで 1 周したか？
57 :                 mode = 3;
58 :                 break;
59 :             }
60 :             break;
61 :
62 :         case 3:
63 :             // ステッピングモータ停止
64 :             stm = 0;
65 :             stm_step_cnt = 0;       // ステップ数をクリア
66 :             mode = 0;
67 :             break;
68 :     }
```

```
 69 :  }
 70 :
 71 :  ////////////////////////////////////////////////////////////////////////////
 72 :  // end of file
 73 :  ////////////////////////////////////////////////////////////////////////////
```

8.4 プログラムの解説

行	詳細
33～42	状態遷移図の ⓪部分です。 タクトスイッチ ON でトグルスイッチが OFF の場合は①へ、タクトスイッチ ON でトグルスイッチが ON の場合は②へ、移動します。
44～51	状態変移図の①部分です。 ステッピングモータを時計回りで回します。1 回転（＝200 ステップ）すると、③へ移動します。
53～60	状態変移図の②部分です。 ステッピングモータを反時計回りで回します。1 回転（＝200 ステップ）すると、③へ移動します。 ポイントは、反時計回りなので、stm_step_cnt 変数は 0 から減っていくということです。反時計回りでの 1 回転は-200 となります。-200 以下になったら③へ移動します。
62～67	状態変移図の③部分です。 ステッピングモータを時計回りで回します。1 回転（＝200 ステップ）すると、③へ移動します。

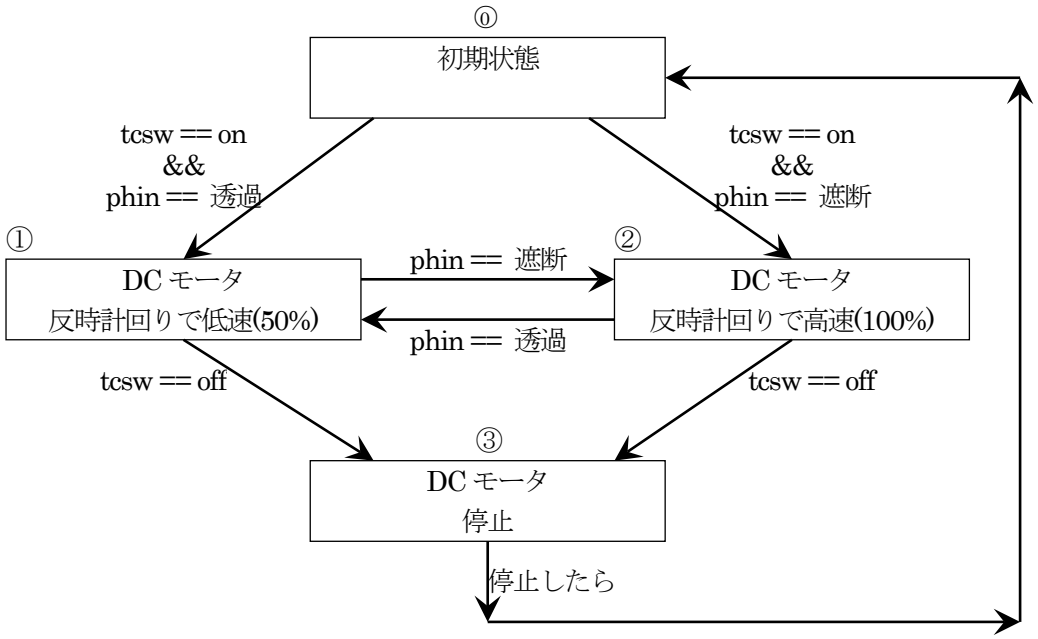
9. 課題 3

9.1 課題

下記のプログラムを作成してください。

(1) DC モータは、タクトスイッチが ON で反時計回りに回転し、OFF で停止する。
(2) 回転速度は、フォトインタラプタの状態により決まる。
　(A) フォトインタラプタが透過で、低速に回転する。
　(B) フォトインタラプタが遮断で、高速に回転する。
　(C) 上記(A)と(B)は、回転中にいつでも変更ができる。

9.2 状態遷移図



9.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題３
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
```

```
14 : ////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                            // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                        // メインは無限ループ、これ以降は
22 :                                        // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int mode = 0;               // 状態(必ず static にすること!)
31 :
32 :     switch( mode ) {
33 :         case 0:
34 :             // タクトスイッチとフォトインタラプタの状態をチェック
35 :             if( tcsw == 1 && phin == 0 ) {
36 :                 mode = 1;
37 :             } else if(  tcsw == 1 && phin == 1 ) {
38 :                 mode = 2;
39 :             }
40 :             break;
41 :
42 :         case 1:
43 :             // DC モータ  50%で反時計回り
44 :             dcm = -5;                  // DC モータ  50%で反時計回り
45 :             if( phin == 1 ) {          // フォトインタラプタが遮断したら 2 へ
46 :                 mode = 2;
47 :                 break;
48 :             }
49 :             if( tcsw == 0 ) {          // タクトスイッチが離されたら 3 へ
50 :                 mode = 3;
51 :                 break;
52 :             }
53 :             break;
54 :
55 :         case 2:
56 :             // DC モータ 100%で反時計回り
57 :             dcm = -10;                  // DC モータ 100%で反時計回り
58 :             if( phin == 0 ) {          // フォトインタラプタが透過したら 1 へ
59 :                 mode = 1;
60 :                 break;
61 :             }
62 :             if( tcsw == 0 ) {          // タクトスイッチが離されたら 3 へ
63 :                 mode = 3;
64 :                 break;
65 :             }
66 :             break;
67 :
68 :         case 3:
69 :             // DC モータ 停止
70 :             dcm = 0;                   // DC モータ 停止
71 :             mode = 0;
72 :             break;
73 :     }
74 : }
75 :
76 : ////////////////////////////////////////////////////////////////////////
77 : // end of file
78 : ////////////////////////////////////////////////////////////////////////
```

10. 課題 4

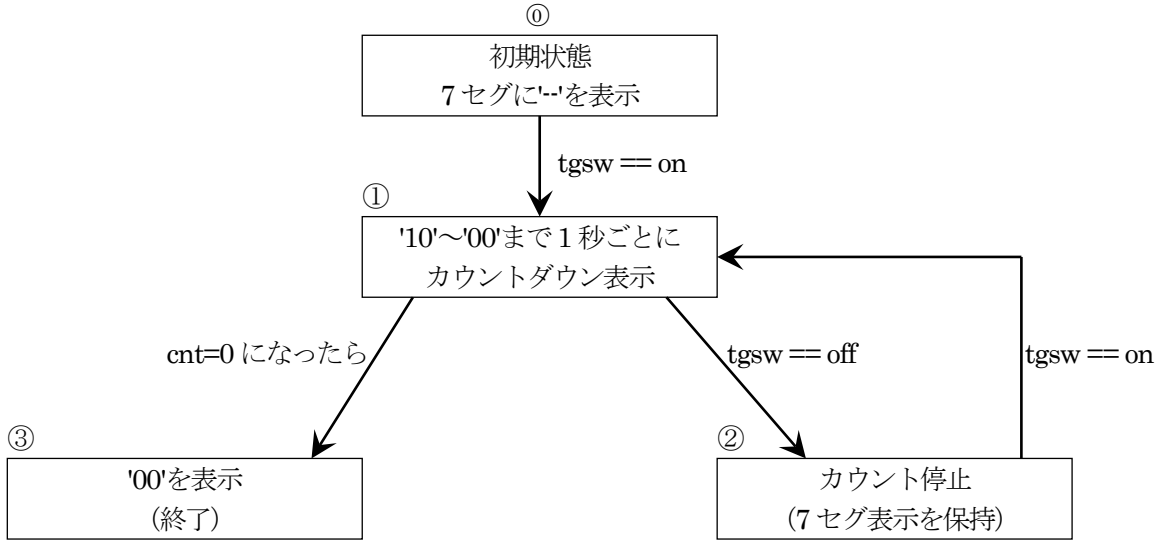
10.1 課題

下記のプログラムを作成してください。

(1) プログラム開始直後、左右の 7 セグメント LED に"--"を表示する。
(2) トグルスイッチが ON になると、"10"を表示し、約 1 秒ごとにカウントダウンし、"00"で停止する。表示中は 2 桁表示すること。
(3) カウントダウン中にトグルスイッチが OFF になると、カウントダウンを停止する。再び ON にするとカウントダウンを再開する。停止中の 7 セグメント LED の表示は、停止したときの表示を維持すること。

10.2 状態遷移図

※7 セグメント LED を「7 セグ」と略します。



10.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題４
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
```

```
14 : ////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                          // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                      // メインは無限ループ、これ以降は
22 :                                      // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int cnt = 0;              // 時間カウント(1ms ごとに+1)用(必ず static にすること!)
31 :     static int mode = 0;             // 状態(必ず static にすること!)
32 :
33 :     switch( mode ) {
34 :         case 0:
35 :             // プログラム開始直後
36 :             seg_left  = SEG_MAI;     // 開始直後の 7 セグ表示
37 :             seg_right = SEG_MAI;
38 :             cnt = 10000;
39 :             if( tgsw == 1 ) {        // トグルスイッチ ON なら
40 :                 mode = 1;
41 :             }
42 :             break;
43 :
44 :         case 1:
45 :             // カウントダウン
46 :             cnt--;                   // 1ms ごとに-1 する
47 :             seg_left  = (cnt / 1000 + 1) / 10; // 10 の桁
48 :             seg_right = (cnt / 1000 + 1) % 10; //  1 の桁
49 :             if( tgsw == 0 ) {        // トグルスイッチ OFF なら
50 :                 mode = 2;
51 :             }
52 :             if( cnt == 0 ) {   // 0 秒になったなら
53 :                 mode = 3;
54 :             }
55 :             break;
56 :
57 :         case 2:
58 :             // トグルスイッチ ON になるまで待機
59 :             if( tgsw == 1 ) {        // トグルスイッチが ON なら
60 :                 mode = 1;
61 :             }
62 :             break;
63 :
64 :         case 3:
65 :             // 0 秒になったなら"00"表示で終了
66 :             seg_left  = 0;
67 :             seg_right = 0;
68 :             break;
69 :     }
70 : }
71 :
72 : ////////////////////////////////////////////////////////////////////////
73 : // end of file
74 : ////////////////////////////////////////////////////////////////////////
```

10.4 プログラムの解説

行	詳細

47、48	47 行、48 行は、カウントダウン中の cnt の値を 7 セグメント LED へ表示している部分です。
	※「%」は、余りを求める式です。例えば「 3 % 2 」は、1 が求まります。
	
	47 :　　seg_left = (cnt / 1000 + 1) / 10; // 10 の桁
	48 :　　seg_right = (cnt / 1000 + 1) % 10; // 1 の桁
	
	cnt 変数は 1ms ごとに-1 するので、1000 で割って、1 秒単位にします。このとき、下記のように「+1」部分が無いとどうなるでしょう。
	
	47 :　　seg_left = (cnt / 1000 ~~+ 1~~) / 10; // 10 の桁
	48 :　　seg_right = (cnt / 1000 ~~+ 1~~) % 10; // 1 の桁
	
	cnt は最初 10000 ですが、1ms 後に 9999 になります。計算結果は整数になるので、
	
	seg_left = (9999/1000) / 10 ＝ 9 / 10 = 0
	seg_rigth = (9999/1000) % 10 ＝ 9 % 10 = 9
	
	となります。課題は「'10'→'09'→・・・・→'01'→'00'」を各 1 秒ごとに表示するよう指示されていますが、'10'は表示されません。よって、＋1して、'10'を 1 秒表示するようにしています。
	このとき、気をつけなければいけないのは、終わり(cnt=0)の時です。次のようになります。
	
	seg_left = (0/1000 + 1) / 10 ＝ (0 + 1) / 10 = 0
	seg_rigth = (0/1000 + 1) % 10 ＝ (0 + 1) % 10 = 1
	
	cnt＝0 のとき'01'表示になりますが、cnt＝0 になった瞬間に③に移り、③で'00'表示をするので、今回のプログラムでは課題通りになります。
	
	必ず初期値(cnt=10000 のとき)と終了値(cnt=0 のとき)に、どのような表示になるか想定しながらプログラムを作っていきましょう。また、できれば、cnt=9999、cnt=1 のときも、想定しておきましょう。

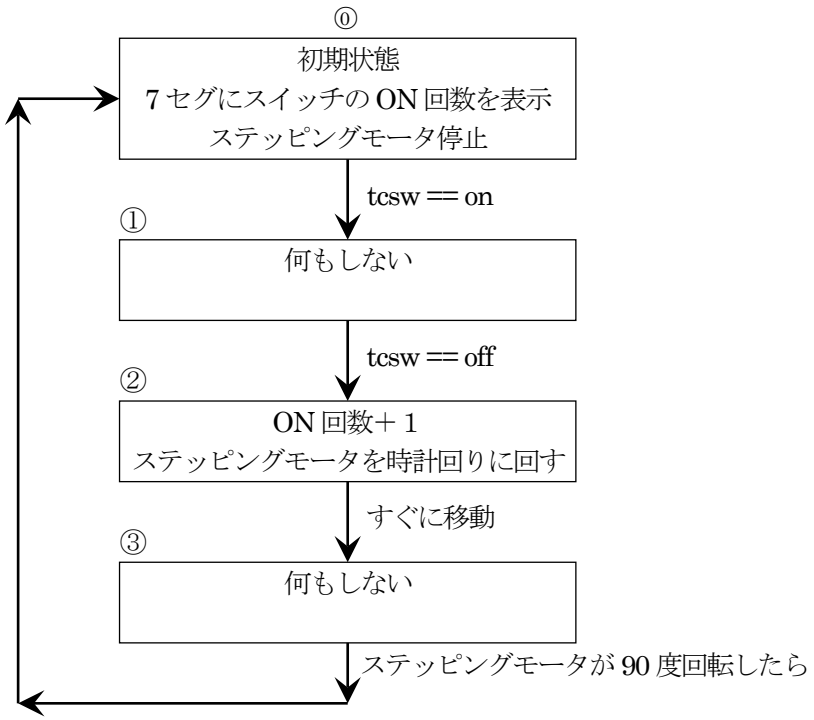
11. 課題 5

11.1 課題

下記のプログラムを作成してください。

(1) プログラム開始直後、右側の 7 セグメント LED に"0"を表示する。その後、ステッピングモータの指示針を手で動かし、0 の位置に合わせる。
(2) タクトスイッチの「ON→OFF」を 12 回行う。
　(A) 1 回のタクトスイッチの「ON→OFF」で、ステッピングモータは時計回りに 90 度回転する。
　(B) 右側の 7 セグメント LED に、タクトスイッチの操作回数を 16 進数で表示する。
　(C) 上記(A)と(B)は、タクトスイッチの「ON→OFF」の、「OFF」になった瞬間に動作する。

11.2 状態遷移図



11.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題５
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
14 : ////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                            // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                        // メインは無限ループ、これ以降は
22 :                                        // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int mode = 0;               // 状態(必ず static にすること!)
31 :     static int on_cnt = 0;             // ON の回数(実際は ON→OFF にした瞬間に+1)
32 :
33 :     switch( mode ) {
34 :         case 0:
35 :             // タクトスイッチ ON になったかチェック
36 :             stm = 0;                   // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ停止
37 :             seg_right = on_cnt;        // 右側 7 セグに ON→OFF した回数を表示
38 :             if( tcsw == 1 ) {
39 :                 mode = 1;
40 :             }
41 :             break;
42 :
43 :         case 1:
44 :             // タクトスイッチ OFF になったかチェック
45 :             if( tcsw == 0 ) {
46 :                 mode = 2;
47 :             }
48 :             break;
49 :
50 :         case 2:
51 :             // カウントを＋１して、ステッピングモータを回す
52 :             on_cnt++;                  // ON 回数を+1 する
53 :             seg_right = on_cnt;        // 右側 7 セグに ON→OFF した回数を表示
54 :             stm = 5;                   // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ正転で 5ms ごとに励磁
55 :             mode = 3;
56 :             break;
57 :
58 :         case 3:
59 :             // ステッピングモータが 90 度進んだら停止(90 度/1.8=50 パルス)
60 :             if( stm_step_cnt >= on_cnt * 50 ) {
61 :                 mode = 0;
62 :             }
63 :             break;
64 :     }
65 : }
66 :
```

```
67 :  ////////////////////////////////////////////////////////////////////////////
68 :  // end of file
69 :  ////////////////////////////////////////////////////////////////////////////
```

11.4 プログラムの解説

行	詳細
31	on_cnt という変数を作り、タクトスイッチが OFF になった瞬間、+1 するようにします。
50～56	②のプログラム部分です。スイッチを離した瞬間、この部分を実行します。 on_cnt 変数を＋1して、seg_right に代入し、7 セグの右に押した回数を表示します。 stm 変数に 5 をセットして、時計回りで回るようにします。
58～63	③のプログラム部分です。 90 度はステップ数で、90÷1 ステップ 1.8 度＝50 です。 よって、stm_step_cnt 変数が ON 回数×50 以上になったら、⓪へ移動するようにします。 0 部分のプログラムでステッピングモータは停止します。

12. 課題 6

12.1 課題

下記のプログラムを作成してください。

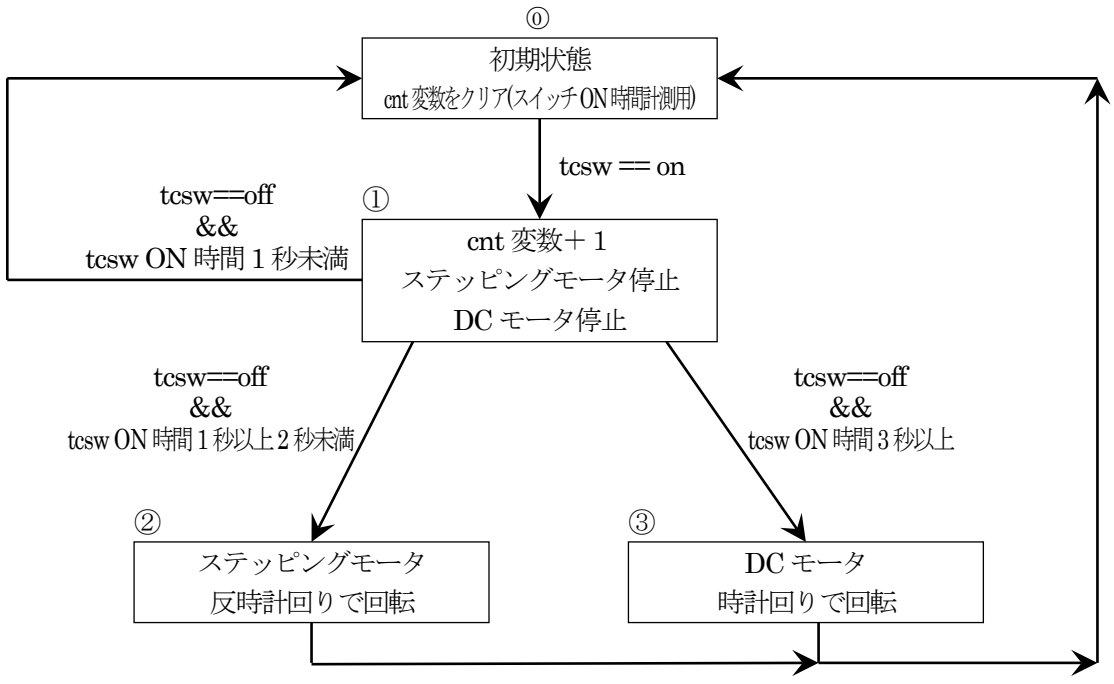
(1) タクトスイッチが ON になった瞬間、回転しているモータがあれば停止させ、ON にしてから OFF になるまでの時間により、次の動作をする。

(A) ON にしてから OFF にするまでの時間が約 1 秒未満なら、ステッピングモータ、DC モータ、どちらも回転しない。

(B) ON にしてから OFF にするまでの時間が約 1 秒以上、約 3 秒未満なら、ステッピングモータのみ反時計回りに回転する。ステッピングモータは、OFF になった瞬間から動作する。

(C) ON にしてから OFF にするまでの時間が約 3 秒以上なら、DC モータのみ時計回りに回転する。DC モータは OFF になった瞬間から動作する。

12.2 状態遷移図



12.3 プログラム例

```
1 : ////////////////////////////////////////////////////////////////////////////
1 : ////////////////////////////////////////////////////////////////////////////
2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題６
3 : // 座席番号：●
4 : // 氏　　名：○○　○○
5 : //
6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
7 : ////////////////////////////////////////////////////////////////////////////
8 :
9 : ////////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                    // 共通ファイルの取り込み
13 :
14 : ////////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                            // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                        // メインは無限ループ、これ以降は
22 :                                        // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int mode = 0;               // 状態(必ず static にすること!)
31 :     static int cnt = 0;                // 時間カウント(1ms ごとに+1)用(必ず static にすること!)
32 :
33 :     switch( mode ) {
34 :         case 0:
35 :             // タクトスイッチ ON を検出
36 :             cnt = 0;                   // タクトスイッチ ON 時間計測変数クリア
37 :             if( tcsw == 1 ) {
38 :                 mode = 1;
39 :             }
40 :             break;
41 :
42 :         case 1:
43 :             // タクトスイッチ OFF を検出
44 :             stm = 0;                   // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
45 :             dcm = 0;                   // DC モータ：停止
46 :             cnt++;                     // ﾀｸﾄｽｲｯﾁが ON の間、ｶｳﾝﾀを 1ms ごとに+1
47 :             if( tcsw == 0 && cnt < 1000 ) {
48 :                 // タクトスイッチ OFF で、1 秒未満なら
49 :                 mode = 0;
50 :                 break;
51 :             } else if( tcsw == 0 && cnt < 3000 ) {
52 :                 // タクトスイッチ OFF で、3 秒未満なら
53 :                 mode = 2;
54 :                 break;
55 :             } else if( tcsw == 0 ) {
56 :                 // タクトスイッチ OFF で 3 秒以上なら
57 :                 mode = 3;
58 :                 break;
59 :             }
60 :             break;
61 :
```

```
62 :          case 2:
63 :              // ステッピングモータ　反時計回り
64 :              stm = -5;                 // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：反時計回り
65 :              mode = 0;
66 :              break;
67 :
68 :          case 3:
69 :              // DC モータ　時計回り
70 :              dcm = 5;                  // DC ﾓｰﾀ：反時計回り
71 :              mode = 0;
72 :              break;
73 :      }
74 :  }
75 :
76 :  ////////////////////////////////////////////////////////////////////////////
77 :  // end of file
78 :  ////////////////////////////////////////////////////////////////////////////
```

12.4 プログラムの解説

行	詳細
31	cnt 変数を用意します。スイッチが ON の間、＋1するようにします。 これで ON している時間を計ることができます。
42～60	①部分です。 cnt 変数の値をチェックすることにより、スイッチを ON していた時間を計ることができます。 47 行で、スイッチ OFF かつ、cnt 変数が 1000 未満なら、何もしません（ステッピングモータ、DC モータ、ともに停止）。 51 行で、スイッチ OFF かつ、cnt 変数が 3000 未満なら、ステッピングモータを反時計回りに回します。 47 行で cnt 変数が 1000 未満の場合をチェックしているので、51 行は必ず cnt 変数が 1000 以上のときになります。 55 行で、スイッチ OFF なら、DC モータを時計回りに回します。47 行と 51 行で cnt 変数が 3000 未満の場合をチェックしているので、55 行は必ず cnt 変数が 3000 以上のときになります。

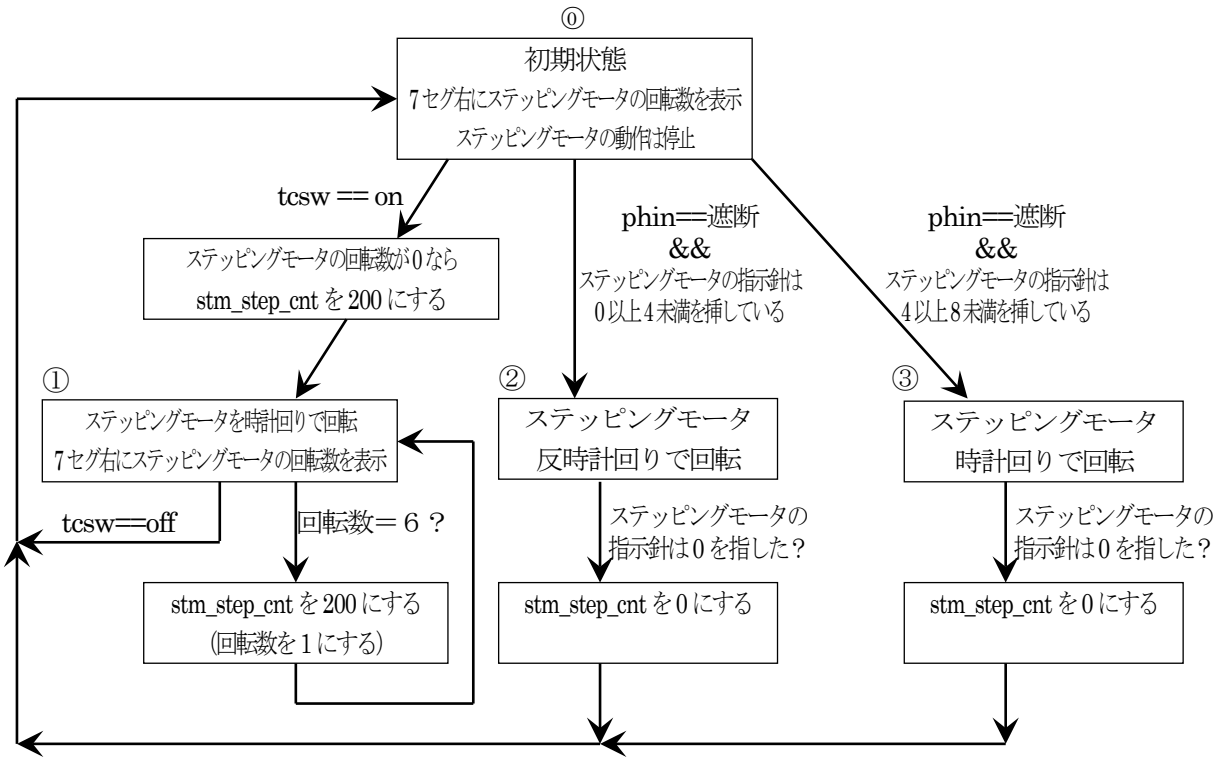
13. 課題 7

13.1 課題

下記のプログラムを作成してください。

(1) プログラム開始直後、右側の 7 セグメント LED に"0"を表示する。その後、ステッピングモータの指示針を手で動かし、0 の位置に合わせる。
(2) ステッピングモータは、タクトスイッチが ON で時計回りに回転し、OFF で停止する。
(3) 右側の 7 セグメント LED にステッピングモータの周回数を次のように表示する。
　(A) ステッピングモータの回転開始と同時に、周回数を加算する。
　(B) ステッピングモータが目盛盤の 0 を通過するたびに周回数を加算する。
　(C) 右側の 7 セグメント LED の表示は、"5"の次は"1"に戻り繰り返す。
(4) ステッピングモータが停止しているとき、フォトインタラプタの光を「遮断→透過」にすると、遮断した瞬間にステッピングモータが次の条件で回転し、指示針が目盛盤の 0 の位置で停止する。
　(A) ステッピングモータの指示針が目盛盤の 0 以上 4 未満の場合、反時計回りに回転する。
　(B) ステッピングモータの指示針が目盛盤の 4 以上 8 未満(0 の位置)の場合、時計回りに回転する。
(5) 上記(4)の動作後、ステッピングモータの指示針が目盛盤の 0 の位置に戻ると、右側の 7 セグメント LED の表示は"0"になり、(2)以降の動作が行える。

13.2 状態遷移図



13.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////////
 2 : // ものづくりコンテスト電子回路組立　入力・出力回路練習基板　課題７
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2016 株式会社日立ドキュメントソリューションズ
 7 : ////////////////////////////////////////////////////////////////////////////
 8 :
 9 : ////////////////////////////////////////////////////////////////////////////
10 : // インクルード
11 : ////////////////////////////////////////////////////////////////////////////
12 : #include "common.c"                      // 共通ファイルの取り込み
13 :
14 : ////////////////////////////////////////////////////////////////////////////
15 : // メイン関数
16 : ////////////////////////////////////////////////////////////////////////////
17 : void main( void )
18 : {
19 :     init();                              // 内蔵周辺機能の初期化
20 :
21 :     while( 1 );                          // メインは無限ループ、これ以降は
22 :                                          // interrupt_1ms 関数で 1ms ごとに実行する
23 : }
24 :
25 : ////////////////////////////////////////////////////////////////////////////
26 : // 1ms ごとに実行する関数
27 : ////////////////////////////////////////////////////////////////////////////
28 : void interrupt_1ms( void )
29 : {
30 :     static int mode = 0;                 // 状態(必ず static にすること!)
31 :
32 :     switch( mode ) {
33 :         case 0:
34 :             // タクトスイッチ、フォトインタラプタ反応待ち
35 :             seg_right = stm_step_cnt / 200;
36 :             stm = 0;
37 :             if( tcsw == 1 ) {
38 :                 // ステッピングモータは回転開始と同時に周回数を加算するので
39 :                 // 0 の場合は 1 回転したことにする
40 :                 if( stm_step_cnt == 0 ) stm_step_cnt = 200;
41 :                 mode = 1;
42 :                 break;
43 :             }
44 :             if( (phin == 1) && (stm_step_cnt % 200 < 100) ) {
45 :                 // フォトインタラプタ遮断かつ、指示針が 4 未満
46 :                 mode = 2;
47 :                 break;
48 :             } else if( phin == 1 ) {
49 :                 // フォトインタラプタ遮断かつ、指示針が 4 以上
50 :                 mode = 3;
51 :                 break;
52 :             }
53 :             break;
54 :
55 :         case 1:
56 :             // ﾀｸﾄｽｲｯﾁを押している間は、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀは時計回りに回転
57 :             stm = 5;                             // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：時計回り
58 :             seg_right = stm_step_cnt / 200;      // 右側 7 セグに回転数を表示
59 :
60 :             if( stm_step_cnt / 200 >= 6 ) {
61 :                 stm_step_cnt = 200;              // 1 回転分とする
62 :             }
63 :             if( tcsw == 0 ) {
64 :                 // タクトスイッチ OFF なら戻る
65 :                 mode = 0;
66 :                 break;
67 :             }
68 :             break;
```

```
 69 :
 70 :          case 2:
 71 :              // 指示針が 0～4 未満なら、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀは反時計回りで 0 に戻る
 72 :              stm = -5;
 73 :              if( stm_step_cnt % 200 == 0 ) {
 74 :                  stm_step_cnt = 0;               // 回転数を 0 にする
 75 :                  mode = 0;
 76 :                  break;
 77 :              }
 78 :              break;
 79 :
 80 :          case 3:
 81 :              // 指示針が 5～8 未満なら、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀは時計回りで 0 に戻る
 82 :              stm = 5;
 83 :              if( stm_step_cnt % 200 == 0 ) {
 84 :                  stm_step_cnt = 0;               // 回転数を 0 にする
 85 :                  mode = 0;
 86 :                  break;
 87 :              }
 88 :              break;
 89 :      }
 90 :  }
 91 :
 92 :  ////////////////////////////////////////////////////////////////////////
 93 :  // end of file
 94 :  ////////////////////////////////////////////////////////////////////////
```

13.4 プログラムの解説

行	詳細
35	stm_step_cnt 変数は、ステッピングモータ 1 回転 200 パルスの変数です。 よって、200 で割ることによって回転数になります。
40	課題は、「(3)(A) ステッピングモータの**回転開始と同時に、周回数を加算する**。」とあるので、0 の場合のみ、1 周からスタートさせます。1 周＝200 パルスなので、stm_step_cnt 変数に 200 を代入します。
44	stm_step_cnt 変数は 1 回転 200 パルスです。「stm_step_cnt % 200」を計算する事によって 200 で割った余りを求めることができます。よって、何周しても 0～199 の値が求められ、0 が 0 度、199 が 360 度の1つ手前の角度となります。 今回、指示針が 4 未満か、という課題です。指示針 4 は 180 度(0.5 周)なので、 1 周のステップ数 200×0.5＝100 未満なら、4 未満ということになります。
48	44 行で指示針が 4 未満か確認していますので、48 行は「フォトインタラプタが遮断、かつ指示針が 4 以上なら」という判定になります。

14. 参考文献

・第 11 回高校生ものづくりコンテスト全国大会・電子回路組立部門の資料、課題
・ルネサス エレクトロニクス(株)
R8C/38C グループ ユーザーズマニュアル ハードウェア編 Rev.1.10
・ルネサス エレクトロニクス(株)
M16C シリーズ,R8C ファミリ用 C/C++コンパイラパッケージ V.6.00
C/C++コンパイラユーザーズマニュアル Rev.1.00
・ルネサス エレクトロニクス(株)
High-performance Embedded Workshop V.4.09 ユーザーズマニュアル Rev.1.00
・ルネサス半導体トレーニングセンター C言語入門コーステキスト 第 1 版
・電波新聞社 マイコン入門講座 大須賀威彦著 第 1 版
・ソフトバンク(株) 新C言語入門シニア編 林晴比古著 初版
・共立出版(株) プログラマのための ANSI C 全書 L.Ammeraal 著
吉田敬一・竹内淑子・吉田恵美子訳 初版

マイコンカーラリー、販売部品についての詳しい情報は、マイコンカーラリー販売サイトをご覧ください。

https://www2.himdx.net/mcr/

R8C マイコンについての詳しい情報は、ルネサス エレクトロニクスのホームページをご覧ください。

http://japan.renesas.com/

の「製品情報」欄→「マイコン」→「R8C」でご覧頂けます